The thesis entitled

KNOWLEDGE DISCOVERY IN DATABASES WITH AN EMPHASIS ON MULTIPLE LARGE DATABASES

Submitted to the

Goa University

For the degree of

Doctor of Philosophy in

Computer Science and Technology

By

005·74
ADI/Kno
T- 428

Mr Animesh Adhikari

Department of Computer Science

S P Chowgule College

Margao, Goa 403 601

Under the guidance of

Dr P R Rao

Department of Computer Science and Technology

Goa University, Goa

May 30, 2008

428

Dedication

The thesis has been divided into the following three parts: Part 1, Part 2, and Part 3.

Part 1 is dedicated to my mother, Manorama Adhikari, and father, Radhaballav Adhikari.
Part 2 is dedicated to my sisters and brothers.
Part 3 is dedicated to my wife, Jhimli, and our son, Sohom.

Acknowledgement

I would like to thank my advisor, Dr P R Rao, for his support and advices during past three years.

I would like to thank State Government of Goa, India for sponsoring me on faculty improvement program with leave to take up full time research.

I am grateful to Dr R V Gaonkar, Principal, S P Chowgule College, Goa, for processing my study leave application at the right time.

I am also grateful to Dr A S Kanade, former Principal, S P Chowgule College, Goa, for his encouragement and support for higher studies.

I would like to thank to my colleagues for their supports during the period of my study leave.

I am grateful to my wife, Jhimli, my sisters and my brothers for their moral supports.

Statement from student

As required under ordinance OB-9.9 of Goa University, I state that the present Ph D thesis entitled "Knowledge discovery in databases with an emphasis on multiple large databases" is my original contribution and the same has not been submitted on any previous occasion. To the best of my knowledge, the present study is the first comprehensive work of its kind in the area of Data Mining and Knowledge Discovery.

The literature related to the problem investigated has been cited. Due acknowledgements have been made whenever facilities and suggestions have been availed of.

[illegible]

Animesh Adhikari
Department of Computer Science
S P Chowgule College
Margao, Goa 403 602
India

Date: 30/05/2008

CERTIFICATE

This is to certify that the thesis entitled "Knowledge Discovery in Databases with an Emphasis on Multiple Large Databases", submitted by Mr.Animesh Adhikari for the award of the degree of Doctor of Philosophy in Computer Science and Technology is based on his original work carried out under my supervision. The thesis or any part thereof has not been previously submitted for any other degree or diploma in any University or Institute.

[illegible]

Place: Goa University

Dr. P. R. Rao
Department of Computer Science and Technology
Goa University, Taleigao Plateau
Goa-403206, India

Certified that all corrections suggested by the examiners have been incorporated in the thesis.

[illegible]
(M. NARASIMHA MURTY)
Feb. 2, 2009

[illegible]
2/2/2009

List of publications

(1 research monograph, 7 journal papers, 1 book chapter and 4 conference papers)

1. List of published papers

1.1. *Journal papers*

[1] Animesh Adhikari, P R Rao, "Enhancing quality of knowledge synthesized from multi-database mining", *Pattern Recognition Letters* 28(16), 2007, pp. 2312 - 2324.

[2] Animesh Adhikari, P R Rao, "Synthesizing heavy association rules from different real data sources", *Pattern Recognition Letters* 29(1), 2008, pp. 59 - 71.

[3] Animesh Adhikari, P R Rao, "Efficient clustering of databases induced by local patterns", *Decision Support Systems* 44(4), 2008, pp. 925 - 943.

[4] Animesh Adhikari, P R Rao, "Mining conditional patterns in a database", *Pattern Recognition Letters* 29(10), 2008, pp. 1515-1523.

[5] Animesh Adhikari, P R Rao, "Capturing association among items in a database", *Data & Knowledge Engineering* 67(3), 2008, pp. 430-443.

1.2. *Book chapter*

[1] Animesh Adhikari, P R Rao, "Association rules induced by item and quantity purchased", J R Haritsa, R Kotagiri, and V Pudi (Eds.): *Proceedings of International Conference on Database Systems for Advanced Applications (DASFAA)*, LNCS 4947, 2008, pp. 478 - 485.

List of publications

(Continued)

1.3. *Conference papers*

[1] Animesh Adhikari, P R Rao, "Study of select items in multiple databases by grouping", *Proceedings of 3rd Indian International Conference on Artificial Intelligence* (*IICAI*), 2007, pp. 1699 - 1718.

[2] Animesh Adhikari, P R Rao, "Synthesizing global exceptional patterns in multiple databases", *Proceedings of 3rd Indian International Conference on Artificial Intelligence* (*IICAI*), 2007, pp. 512 - 531.

[3] Animesh Adhikari, P R Rao, "A framework for synthesizing arbitrary Boolean expressions induced by frequent itemsets", *Proceedings of 3rd Indian International Conference on Artificial Intelligence* (*IICAI*), 2007, pp. 5 - 23.

[4] Animesh Adhikari, P R Rao, Jhimli Adhikari, "Mining multiple large databases", *Proceedings of 10th International Conference on Information Technology* (*ICIT*), 2007, pp. 80 - 84.

2. List of communicated papers / book

2.1. *Research monograph*

[1] Animesh Adhikari, P R Rao, W Pedrycz, *Developing better multi-database mining applications*, Proposal submitted to Springer.

List of publications

(Continued)

2.2. *Journal Papers*

[1] Animesh Adhikari, P R Rao, Bhanu Prasad, Jhimli Adhikari, "Mining multiple large data sources", communicated to International Arab Journal of Information Technology.

[2] Animesh Adhikari, P R Rao, W Pedrycz, "Study of select items in different sources by grouping", communicated to Data & Knowledge Engineering journal.

Different parts of thesis

Synopsis

Discovering and processing of knowledge in databases are two important categories of tasks. Many important themes of activities could fall under these categories of tasks. The work done in this thesis could be divided into three parts. The theme of Part 1 has been entitled as "*Association Analysis and Pattern Recognition in a Database*". Part 2 is based on the theme "*Pattern Recognition in Multiple Databases*". Finally, the theme of Part 3 has been entitled as "*Developing Better Multi-database Mining Applications*". In the following paragraphs, we describe the work performed in different chapters of these three parts.

We have defined the notion of conditional pattern in a transactional database. It helps us to study the association among the items in Y along with negation of items in X-Y at a given itemset X, for all nonempty Y such that $Y \subseteq X$. We have designed an algorithm to mine interesting conditional patterns in a database. Experiments are conducted on three real databases. The results of the experiments show that conditional patterns store significant nuggets of knowledge about a database.

Frequent itemsets determine major characteristics of a transactional database. Thus, it is important to mine arbitrary Boolean expressions induced by frequent itemsets. From a frequent itemset, one could generate Boolean expressions of members of the frequent itemset connecting through Boolean operators. We have established a simple and elegant framework for synthesizing arbitrary Boolean expressions using conditional patterns in a database. It determines the supports of the Boolean expressions generated from a frequent itemset. Experimental results are provided on both real and synthetic databases.

Many data analyses require a suitable metric to capture association among a set of items in a database. Metrics such as correlation, Jaccard distance could be used to measure association between two items in a database. On the other hand, metrics such as support, collective strength, all-confidence have been used to measure interestingness of an itemset. The major concern regarding computation of collective strength of an itemset is that the computation is based on statistical independence of items of the itemset. Moreover, metrics such as support and all-confidence do not consider the frequencies of subsets of an itemset. In general, an existing metric might not be effective to serve as a measure of association among a set of items in a database. We have presented two measures of association, A_1 and A_2, for capturing association among a set of items. Measure A_1 is the proportion of the number of transactions containing all the items of the itemset and the number of transactions containing at least one of the items of the itemset. On the other hand measure A_2 is based on a weighting model. The weight of a transaction is proportion to the number of items of the itemset present in the transaction. For example, if a transaction contains all the items of an itemset, then it has the maximum weight for the given itemset so far as the association among items in the itemset is concerned. Based on measure A_2, we introduce the notion of associative itemset in a database. We express A_1 and A_2 in terms of supports of itemsets. We also provide theoretical foundation of the work. Finally, we present experimental results on both real and synthetic databases to show the effectiveness of A_2.

Most of the real market basket data are non-binary in the sense that an item could be purchased multiple times in the same transaction. In this case, there are two types of occurrences of an itemset in a database: the number of transactions in the database containing the itemset, called the transaction frequency of an itemset, and the number of occurrences of the itemset in the database, called the database frequency of an itemset. Traditional support-confidence framework might not be adequate for extracting association rules in such a database. We have defined following three categories of association rules: (i) Association rules induced by transaction frequency of an itemset,

(ii) Association rules induced by database frequency of an itemset, and (iii) Association rules induced by both transaction frequency and database frequency of an itemset. We have established a framework based on traditional support-confidence framework for mining each category of association rules. We have presented experimental results based on two databases.

Multi-database mining has been recognized recently as an important area of research. The first problem we present here is to identify global exceptional patterns in multiple databases. A global exceptional pattern describes interesting individuality of few branches. Therefore, it is interesting to identify such patterns. We have given a definition of global exceptional frequent itemset in multiple databases. A global exceptional frequent itemset has high support in multiple databases. But, it is reported from a few data sources. Also, we have defined the notion of exceptional sources for a global exceptional frequent itemset. The data sources that support a global exceptional frequent itemset heavily are called exceptional sources for the global exceptional frequent itemset. We have designed an algorithm for synthesizing global exceptional frequent itemsets. Experimental results are presented on both artificial and real databases. We have compared our algorithm with the existing algorithm theoretically and experimentally. The experimental results show that the proposed algorithm is effective and promising.

We have defined two new patterns, called exceptional association rule and heavy association rule in multiple databases. A heavy association rule has high support and high confidence in multiple databases. An exceptional association rule is a heavy association rule that is reported from a few data sources. On the other hand, a high-frequent association rule is extracted from many data sources. We have designed an algorithm to mine high-frequent, exceptional and heavy association rules in multiple databases. In this connection, we have designed an extended model of local pattern analysis. In the extended model, we have included many layers and interfaces. Thus, it helps us to mine multiple databases systematically. In the extended model, a global pattern is synthesized based on both local patterns and suggested local patterns. Thus, it improves the accuracy of multi-database mining. We present experimental results on three real databases. Also, we make a comparative analysis between our algorithm and an existing algorithm.

Frequent items could be considered as the basic ingredients of different types of knowledge in a database. Based on measure A_2 and a multi-database mining technique, we have designed an algorithm for clustering frequent items in multiple databases. We have conducted experiments on three databases to judge effectiveness of the clustering technique.

Effective data analysis with multiple databases requires highly accurate patterns. Due to large size of some local databases, a traditional data mining technique might not be suitable for mining the collection of all local databases. Also, local pattern analysis might extract low quality of patterns from multiple databases. Thus, it is necessary to improve mining multiple databases. We have designed a new multi-databases mining technique, called PFM + SPS. It combines pipelined feedback model (PFM) and simple pattern synthesizing (SPS) algorithm for mining multiple large databases. In this technique, each local database is mined using a traditional data mining technique in a particular order for synthesizing global patterns. The technique improves quality of synthesized global patterns significantly. We conduct experiments on both real and synthetic databases to judge effectiveness of our technique.

Multi-database mining using local pattern analysis could be considered as an approximate method of mining multiple large databases. Thus, it might be required to enhance the quality of knowledge synthesized from multiple databases. Also, many decision-making applications are directly based on the available local patterns in different databases. The quality of synthesized knowledge / decision based on local patterns in different databases could be enhanced by incorporating more local patterns in the knowledge synthesizing / processing activities. Thus, the available local patterns play a crucial role in building efficient multi-database mining applications. We represent patterns in condensed form by employing a coding, called ACP coding. It allows us to store more local patterns in the main memory. Accordingly, one could extract more patterns by lowering further the user inputs, like minimum support and minimum confidence. The proposed coding enables more local patterns participate in the knowledge synthesizing / processing activities and thus, the quality of synthesized knowledge based on local patterns in different databases gets enhanced significantly at a given pattern synthesizing algorithm and computing resource.

For the purpose of mining relevant databases one may need to cluster the given databases. It could help reducing the cost of searching relevant information in multiple large databases. In this regard, we have defined two measures of similarity between a pair of databases, called $simi_1$ and $simi_2$. Measure $simi_1$ is the ratio of the number frequent itemsets common to databases and the total number of distinct frequent itemsets in these databases. But, the similarity measure $simi_2$ is based on the supports of the frequent itemsets in the databases. We have proved the metric properties of corresponding distance measures. We have designed an algorithm for clustering a set of databases. For the purpose of enhancing efficiency of the clustering process, we have presented a new coding technique, called itemset (IS) coding, for representing frequent itemsets in different databases. It allows more local patterns to participate in decision-making measures. Efficiency of the clustering process has been improved using the following strategies: reducing execution time of clustering algorithm, using more appropriate similarity measure, and storing frequent itemsets space efficiently.

Many important decisions are based on a set of specific items, called the select items. Thus, the analysis of select items in multiple databases is an important issue. For purpose of studying select items in multiple databases, a model of mining global patterns of select items in multiple databases has been designed. We have defined a measure of overall association (*OA*) between two items in a database. Measure *OA* is based on both positive and negative association between two items in a database. We have designed an algorithm based on *OA* for the purpose of grouping the frequent items in multiple databases. Each group contains a select item, called the nucleus item of the group and the group grows centring round the nucleus item. Experimental results are presented on real, synthetic and artificial databases.

A multi-database mining application could be developed using a sequence of stages. It might be possible to provide a framework for each stage of the development process. Finally, we provide a framework for developing better multi-database mining applications.

Table of contents

Table of contents

(Continued)

Table of contents

(Continued)

Table of contents

(Continued)

Table of contents

(Continued)

Table of contents

(Continued)

Part 1

Association analysis and pattern recognition in a database

Chapter 1.1

Introduction

Association analysis of items (variables) as well as patterns in a database could play important roles in finding solutions to many problems. In the context of market basket data, one could perform various types of association analyses of items purchased. Also, there may exist new types of pattern useful in solving different problems. We get to know interesting buying patterns of customers by analyzing a large volume of data. Previous work on mining frequent itemsets, association rules, and negative association rules might have not answered all the queries of a data miner, or a decision maker. Let X be a set of items, called an itemset, purchased frequently in a database. We mention below some issues that have not been addressed in the previous work.

- Given an itemset X, we might be interested in the pattern where the items in Y are purchased and the items in X-Y are not purchased, for nonempty $Y \subseteq X$.
- Given an itemset X, we might be interested in mining arbitrary Boolean expressions induced by items in X.
- Given an itemset X, we would like to measure the amount of statistical association among the items in X.
- We might be interested in the association rules in a database where each transaction contains the items and their quantities purchased.

In Part 1, we have addressed the above issues and made the following contributions.

- The notion of conditional pattern in a database has been introduced.
- An algorithm has been designed to extract interesting conditional patterns from a database.

- A simple and elegant framework has been proposed for synthesizing arbitrary Boolean expressions.
- We have proposed two measures of association A_1 and A_2 for capturing statistical association among a set of items.
- The notion of associative itemset is introduced.
- Three categories of association rules have been introduced in a database containing transactions of items and their quantities purchased. A framework based on traditional support-confidence framework has been proposed for mining each category of association rules.

There might be repetition of some pieces of information in different chapters of different parts. There are two reasons behind such repetitions. Firstly, some chapters are correlated in some sense. For example, a multi-database mining technique described is one chapter and has been used to mine multiple databases for finding a solution to a problem in an earlier chapter. Secondly, each chapter except the introductory and concluding chapters, is made complete with respect to the problem discussed, and has been communicated, or proposed to communicate as a paper to either an international journal, or an international conference. Nevertheless, we have made efforts to reduce such repetitions.

Chapter 1.2

Mining conditional patterns in a database

Association analysis of items [11], [17], and selecting right interestingness measures [41], [75] are two significant tasks being at the heart of many data mining problems. An association analysis is generally associated with interesting patterns in a database. A pattern would become interesting if the associated interestingness measures satisfy some conditions. Association rules [11] and negative association rules [17] are examples of two types of patterns that are synthesized from the itemset patterns in a database. An association rule is expressed by a forward implication $X \to Y$, where X and Y are itemsets in the database. Itemsets X and Y are called the antecedent and consequent of the association rule, respectively. The meaning attached to this type of association rules is that if all the items in X are purchased by a customer then it is likely that all the items in Y are purchased by the same customer at the same time. On the other hand, a negative association rule is expressed by one of the following three forward implications: $X \to \neg Y$, $\neg X \to Y$, and $\neg X \to \neg Y$, where X and Y are itemsets in the given database. Let us consider a negative association rule of the form $X \to \neg Y$. The meaning attached to the negative association rule of the form $X \to \neg Y$ is that if all the items in X are purchased by a customer then it is unlikely that all the items in Y are purchased by the same customer at the same time. Though an association rule expresses interesting association among items in a frequent itemset, it might not be sufficient for all kinds of association analysis among items in the itemset.

The importance of an itemset could be judged by its support [11]. *Support* (*supp*) of an itemset X in database D is the fraction of transactions in D containing X. Itemset X is *frequent* in D if *supp* $(X, D) \geq \alpha$, where α is user-defined *minimum support level.* Itemset $X = \{x_1, x_2, \ldots, x_m\}$ corresponds to Boolean expression $x_1 \wedge x_2 \wedge \ldots \wedge x_m$. Thus, if the itemset $\{x_1, x_2, \ldots, x_m\}$ contains in a transaction then the Boolean expression $x_1 \wedge x_2 \wedge \ldots \wedge x_m$ is true for that transaction. On the other hand, if the itemset $\{x_1, x_2, \ldots, x_m\}$ does not contain in a transaction then the Boolean expression $x_1 \wedge x_2 \wedge \ldots \wedge x_m$ is false for that transaction. In general, let E be a Boolean expression on the items in D. Then, *supp*(E, D) is the fraction of transactions in D that satisfy E.

Frequent itemset mining has received significant attention in the recent time. Several implementations of mining frequent itemsets [32] have been reported. Frequent itemsets are important patterns in a database, since they determine major characteristics of a database. Wu et al. [80] have proposed a solution of inverse frequent itemset mining. Authors argued that one could efficiently generate a synthetic market basket database from the frequent itemsets and their supports. Let X and Y be two itemsets in database D. The characteristics of database D are revealed more by the pair (X, *supp* (X, D)) than that of (Y, *supp* (Y, D)), if *supp*$(X, D) >$ *supp*(Y, D). Thus, it is important to study frequent itemsets more than infrequent itemsets. Negative association rules are generated from infrequent itemsets. Thus, their applications in different problem domains are limited. The goal of this chapter is to study some kind of association among items which is not immediately available from frequent itemsets and association rules.

If X is frequent in D then every non-null subset of X is also frequent in D. Let us consider the following example.

Example 1.2.1. Let $D = \{\{a, b\}, \{a, b, c, d\}, \{a, b, c, h\}, \{a, b, g\}, \{a, b, h\}, \{a, c\}, \{a, c, d\}, \{b\}, \{b, c, d, h\}, \{b, d, g\}\}$. Let $X(\eta)$ denote frequent itemset X with support η. The frequent itemsets in D at minimum support level 0.2 are given as follows: $\{a\}(0.7)$, $\{b\}(0.8)$, $\{c\}(0.5)$, $\{d\}(0.4)$, $\{g\}(0.2)$, $\{h\}(0.3)$, $\{a, b\}(0.5)$, $\{a, c\}(0.4)$, $\{a, d\}(0.2)$, $\{a, h\}(0.2)$, $\{b, c\}(0.3)$, $\{b, d\}((0.3)$, $\{b, g\}(0.2)$, $\{b, h\}(0.3)$, $\{c, d\}(0.3)$, $\{c, h\}(0.2)$, $\{a, b, c\}(0.2)$, $\{a, b, h\}(0.2)$, $\{a, c, d\}(0.2)$, $\{b, c, d\}(0.2)$, $\{b, c, h\}(0.2)$.

Suppose we wish to study association among items in $\{a, b, c\}$. A frequent itemset mining algorithm could mine the following details about the items in $\{a, b, c\}$.

Table 1.2.1. Frequent itemset $\{a, b, c\}$ and its non-null subsets at $\alpha = 0.2$

Itemset	$\{a\}$	$\{b\}$	$\{c\}$	$\{a, b\}$	$\{a, c\}$	$\{b, c\}$	$\{a, b, c\}$
Support	0.7	0.8	0.5	0.5	0.4	0.3	0.2

Table 1.2.1 provides the information regarding how frequently a non-null subset of $\{a, b, c\}$ occurs in D. Such information might not be sufficient for all types of queries and analyses of items in $\{a, b, c\}$. •

A positive association rule finds positive association between two disjoint non-null itemsets. Positive association rules are generated from frequent itemsets in the database. A positive association rule r: $X \rightarrow Y$ in D is characterized by its support and confidence measures [11]. *Support* of association rule r: $X \rightarrow Y$ in D is the fraction of transactions in D containing both X and Y. *Confidence* (*conf*) of an association rule r in D is the fraction of transactions in D containing Y among the transactions containing X. An association rule r in D is interesting if $supp(r, D) \geq \alpha$, and $conf(r, D) \geq \beta$, where β is the *minimum confidence level*. The parameters α and β are user-defined inputs to an association rule mining algorithm. In Example 1.2.2, we generate interesting association rules from $\{a, b, c\}$.

Example 1.2.2. We continue here the discussion of Example 1.2.1. The interesting association rules generated from $\{a, b, c\}$ are given in Table 1.2.2.

Table 1.2.2. Association rules generated from $\{a, b, c\}$ at $\alpha = 0.2$ and $\beta = 0.5$

Association rule	Support	Confidence
$\{a, c\} \rightarrow \{b\}$	0.2	0.5
$\{b, c\} \rightarrow \{a\}$	0.2	0.66667

•

The rest of the chapter is organized as follows. In Section 1.2.2, we introduce the notion of conditional pattern in a database. We discuss properties of conditional patterns in Section 1.2.3. In Section 1.2.4, we propose an algorithm for extracting conditional patterns from a database. Experimental results are given in Section 1.2.5. Also, we present an application of conditional patterns. We discuss related work in Section 1.2.6.

1.2.2 Conditional pattern

The study of items in $\{a, b, c\}$ might be incomplete if we know the supports of its non-null subsets and the association rules with respect to the non-null subsets of $\{a, b, c\}$. Thus, the information provided in Tables 1.2.1 and 1.2.2 might not be sufficient for all types of queries and analyses related to items in $\{a, b, c\}$. In fact, there are some queries related to items in $\{a, b, c\}$ whose answers are not immediately available from Tables 1.2.1 and 1.2.2. A few examples of such queries are given below.

- Given a frequent itemset $\{a, b, c\}$, find the support of Boolean expression containing item a but not items b and c.
- Given a frequent itemset $\{a, b, c\}$, find the support of Boolean expression containing items a and b but not item c.

The above queries correspond to a specific type of pattern in a database. Some of these patterns could have significant supports, since $\{a, b, c\}$ is a frequent itemset. In general, if we wish to study the association among the items in Y along with negation of items in X-Y, then such analysis is not immediately available from frequent itemsets and positive association rules, given the itemsets X in a database such that $Y \subseteq X$. Such association analyses could be interesting, since the corresponding Boolean expressions could have high supports. Therefore, we need to mine such patterns for an effective analysis of items in frequent itemsets.

Let $\langle Y, X\rangle$ be a pattern that a transaction in a database contains all the items of Y, but not items of X-Y, for itemsets X and Y in the database such that $\phi \neq Y \subseteq X$. Let *supp* $\langle Y, X,$

$D\rangle$ be the fraction of transactions in database D containing all the items of Y, but not the items of X-Y, for itemset X in D such that $Y \subseteq X$. A pattern of type $\langle Y, X\rangle$ is called a *conditional pattern* [9]. A conditional pattern $\langle Y, X\rangle$ has two components: *pattern itemset* (Y) and *reference itemset* (X). Thus, a conditional pattern $\langle Y, X\rangle$ is associated with two values: *supp*$\langle Y, X, D\rangle$ and *supp*(X, D). *supp*$\langle Y, X, D\rangle$ and *supp*(X, D) are called *conditional support* and *reference support* of conditional pattern $\langle Y, X\rangle$ in D, respectively. The conditional support and reference support of conditional pattern $\langle Y, X\rangle$ in D are denoted by *csupp*$\langle Y, X, D\rangle$ and *rsupp*$\langle Y, X, D\rangle$, respectively. In other words, *supp* $\langle Y, X, D\rangle$ and *supp*(X, D) are denoted by *csupp* $\langle Y, X, D\rangle$ and *rsupp* $\langle Y, X, D\rangle$, respectively. A conditional pattern $\langle Y, X\rangle$ in D is *interesting* if *csupp*$\langle Y, X, D\rangle \geq \delta$ and *rsupp*$\langle Y, X, D\rangle \geq \alpha$, where δ is the *minimum conditional support.* The parameters α and δ are user-defined inputs to a conditional pattern mining algorithm.

The following figures provide more information about given queries.

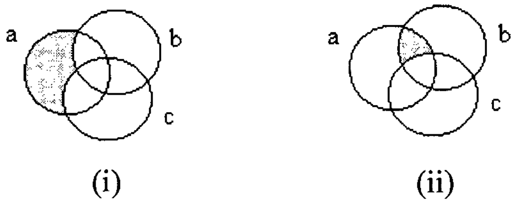


Figure 1.2.1. Shaded regions in (i) and (ii) correspond to conditional supports of $\langle\{a\}, \{a, b, c\}\rangle$ and $\langle\{a, b\}, \{a, b, c\}\rangle$ in D, respectively

The shaded region in Figure 1.2.1(i) is a set of transactions in D such that each transaction contains item a but not items b and c, with respect to $\{a, b, c\}$. The shaded region in Figure 1.2.1(ii) is a set of transactions in D such that each transaction contains items a and b but not item c, with respect to $\{a, b, c\}$. Conditional support of a conditional pattern could be synthesized using supports of relevant itemsets in the database. For example, *csupp*$\langle\{a\}, \{a, b, c\}, D\rangle$ and *csupp*$\langle\{a, b\}, \{a, b, c\}, D\rangle$ could be synthesized as follows.

Table 1.2.4. Non-trivial conditional patterns with respect to $\{a, b, c\}$ at $\delta = 0.2$ and $\alpha = 0.2$

Conditional pattern	*csupp*	*rsupp*	Conditional pattern	*csupp*	*rsupp*
$\langle\{b\}, \{a, b, c\}\rangle$	0.2	0.2	$\langle\{a, c\}, \{a, b, c\}\rangle$	0.2	0.2
$\langle\{a, b\}, \{a, b, c\}\rangle$	0.3	0.2			

●

We observe that $csupp\langle Y, X, D\rangle \leq supp(Y, D)$, for $Y \subset X$. Nonetheless, $csupp\langle Y, X, D\rangle$ could be high, if X is frequent in D. Thus, it is necessary to study such patterns in a database for effective analysis of items in frequent itemsets. The problem could be stated as follows.

We are given a database D of customer transactions. Extract interesting non-trivial conditional patterns from D.

1.2.3 Properties of conditional patterns

In this section, we present some interesting properties of conditional patterns in a database. Before presenting the properties, we introduce some notations. Let $X = \{x_1, x_2, \ldots, x_m\}$ and $Y = \{y_1, y_2, \ldots, y_p\}$. Then, $supp(X \cup Y, D)$ and $supp(X \cap Y, D)$ refer to $supp((x_1 \wedge x_2 \wedge \ldots \wedge x_m) \vee (y_1 \wedge y_2 \wedge \ldots \wedge y_p), D)$ and $supp((x_1 \wedge x_2 \wedge \ldots \wedge x_m) \wedge (y_1 \wedge y_2 \wedge \ldots \wedge y_p), D)$, respectively.

Lemma 1.2.1. *Let E be a Boolean expression that a transaction contains at least one item of itemset X in database D. Then,* $supp(E, D) = \sum_{Y \subseteq X, Y \neq \phi} csupp\langle Y, X, D\rangle$ (1.2.3)

Proof. We re-state the theorem of total probability [31] in terms of supports as follows: For any m Boolean expressions $X_1, X_2, \ldots, X_m$ in database D, we have $supp\left(\bigcup_{i=1}^{m} X_i, D\right) = \sum_{i=1}^{m} supp(X_i, D) - \sum_{i<j; i,j=1}^{m} supp(X_i \cap X_j, D) + \ldots + (-1)^{m-1} supp\left(\bigcap_{i=1}^{m} X_i, D\right)$. The events $\langle Y, X\rangle$ and $\langle Z, X\rangle$ are mutually exclusive, for $Y \neq Z$, $Y \subseteq X$ and $Z \subseteq X$. Thus, $supp(\langle Y, X\rangle \cap \langle Z, X\rangle, D) = 0$, for $Y \neq Z$, $Y \subseteq X$ and $Z \subseteq X$. ●

Let $X = \{a, b, c\}$. With reference to Examples 1.2.1 and 1.2.3, $supp(a \vee b \vee c, D) = 1$ and $supp(a \vee b \vee c, D) = csupp\langle\{a\}, X, D\rangle + csupp\langle\{b\}, X, D\rangle + csupp\langle\{c\}, X, D\rangle + csupp\langle\{a, b\}, X, D\rangle + csupp\langle\{a, c\}, X, D\rangle + csupp\langle\{b, c\}, X, D\rangle + csupp\langle X, X, D\rangle$. It validates Lemma 1.2.1.

Lemma 1.2.2. *$supp(X, D) \leq \sum_{Y \subseteq X, Y \neq \phi} csupp\langle Y, X, D\rangle$, for any two itemsets X and Y in database D such that $Y \subseteq X$.*

Proof. Let $X = \{x_1, x_2, \ldots, x_m\}$. Then X corresponds to Boolean expression $x_1 \wedge x_2 \wedge \ldots \wedge x_m$ in D. Let E be a Boolean expression that a transaction contains at least one item of itemset X in D. Then, $supp(E, D) = \sum_{Y \subseteq X, Y \neq \phi} csupp\langle Y, X, D\rangle$, by Lemma 1.2.1. Therefore, $supp(E, D) = csupp\langle X, X, D\rangle + Q$, where $Q \geq 0$. Then, $supp(E, D) = supp(X, D) + Q$, since $supp(X, D) = csupp\ \langle X, X, D\rangle$. The lemma follows. •

With reference to Examples 1.2.1 and 1.2.3, let $X = \{a, b, c\}$. Then, $supp(X, D) = 0.2$. Now, $csupp\langle\{a\}, X, D\rangle + csupp\langle\{b\}, X, D\rangle + csupp\langle\{c\}, X, D\rangle + csupp\langle\{a, b\}, X, D\rangle + csupp\ \langle\{a, c\}, X, D\rangle + csupp\langle\{b, c\}, X, D\rangle + csupp\langle X, X, D\rangle = 1.0 \geq 0.2$. It validates Lemma 1.2.2.

Lemma 1.2.3. *The conditional supports of $\langle X, Y\rangle$ and $\langle X, Z\rangle$ in a database may not be equal, for any three itemsets X, Y and Z in the database such that $X \subseteq Y$ and $X \subseteq Z$.*

Proof. The itemsets Y-X and Z-X may not be the same. Thus, the lemma follows. •

With reference to Example 1.2.1, we get $csupp\langle\{a, b\}, \{a, b, h\}, D\rangle = 0.3$ and $csupp\langle\{a, b\}, \{a, b, d\}, D\rangle = 0.4$. We observe that $csupp\langle\{a, b\}, \{a, b, h\}, D\rangle \neq csupp\langle\{a, b\}, \{a, b, d\}, D\rangle$. It validates Lemma 1.2.3.

Lemma 1.2.4. *There is no fixed ordered relationship between conditional supports of $\langle Y, X\rangle$ and $\langle Z, X\rangle$ in a database, for any three itemsets X, Y and Z in the database such that $Z \subseteq Y \subseteq X$.*

Proof. Let X, Y and Z be three itemsets in database D such that $csupp\langle Y, X, D\rangle \leq csupp\langle Z, X, D\rangle$, for some $Z \subseteq Y \subseteq X$. Also, there may exist another three itemsets P, Q, and R in database D such that $csupp\langle R, P, D\rangle \le csupp\langle Q, P, D\rangle$, for some $R \subseteq Q \subseteq P$. The proof is based on a counter example. With reference to Example 1.2.1, let $X = \{a, b, c\}$, $Y = \{a, b\}$ and $Z = \{a\}$. Then, $csupp\langle Z, X, D\rangle = csupp\langle \{a\}, \{a, b, c\}, D\rangle = 0$, and $csupp\langle Y, X, D\rangle = csupp\langle \{a, b\}, \{a, b, c\}, D\rangle = 0.3$. In this case, we observe $csupp\langle Z, X, D\rangle \le csupp\langle Y, X, D\rangle$, for $Z \subseteq Y \subseteq X$. Let $A = \{b, d, h\}$, $B = \{b, d\}$ and $C = \{b\}$. Then, $csupp\langle C, A, D\rangle = csupp\langle \{b\}, \{b, d, h\}, D\rangle = 0.3$, and $csupp\langle B, A, D\rangle = csupp\langle \{b, d\}, \{b, d, h\}\rangle = 0.2$. In this case, we observe $csupp\langle B, A, D\rangle \le csupp\langle C, A, D\rangle$, for $C \subseteq B \subseteq A$. •

We could synthesize a set of frequent itemsets from a set of association rules. In particular, let r_1: $X \rightarrow Y$ and r_2: $X \rightarrow Z$ be two positive association rules in D, where X, Y and Z are three frequent itemsets in D. The set of frequent itemsets synthesized from $\{r_1, r_2\}$ is $\{X, XY, XZ\}$. In a similar way, we could synthesize a set of frequent itemsets from a set of conditional patterns. In particular, let cp_1: $\langle\{x_1, x_2\}, \{x_1, x_2, x_3\}\rangle$, and cp_2: $\langle\{x_1, x_3\}, \{x_1, x_2, x_3\}\rangle$ be two conditional patterns in D, where x_i is an item in D, for $i = 1, 2, 3$. The set of frequent itemsets is synthesized from $\{cp_1, cp_2\}$ is $\{\{x_1, x_2\}, \{x_1, x_3\}, \{x_1, x_2, x_3\}\}$.

Example 1.2.4. With reference to Table 1.2.2, the set of frequent itemsets synthesized from the set of positive association rules is given as follows: $\{\{a, c\}(0.4), \{b, c\}(0.3), \{a, b, c\}(0.2)\}$. With reference to Table 1.2.4, the set of frequent itemsets synthesized from the set of conditional patterns is given as follows: $\{\{a, b\}(0.5), \{a, c\}(0.4), \{a, b, c\}(0.2)\}$. •

From Example 1.2.4, one could conclude that the association rules and conditional patterns in a database may not represent the same information about a database, since the amount of information conveyed by association rules in a database is dependent on β, at a given α. Also, the information conveyed by the conditional patterns in a database is dependent on δ, at a given α. Thus, we have the following definition.

Definition 1.2.1. A set of association rules A and a set of conditional patterns C in a database convey the same information about a given database if the set of frequent itemsets synthesized from A is the same as the set of frequent itemsets synthesized from C. •

Lemma 1.2.5. *The set association rules in a database at $\beta = \alpha$ and the set of conditional patterns in the database at $\delta = 0$ represent the same information about the database at a given α.*

Proof. Let S be a set of frequent itemsets in database D. Also, let $CLOSURE(S) = \{s: (s \in S)$, or $(s \neq \phi$ and $s \subseteq p \in S)\}$. Let $FIS(D, i)$ be the set of frequent itemsets in D of size i, for an $i = 1, 2, \ldots$. The set of frequent itemsets synthesized from association rules in D at $\beta = \alpha$ is equal to $CLOSURE\left(\bigcup_{i\geq 2} FIS(D,i)\right)$. Also, the set of frequent itemsets synthesized from the conditional patterns in D at $\delta = 0$ is equal to $CLOSURE\left(\bigcup_{i\geq 2} FIS(D,i)\right)$. •

With reference to Example 1.2.1, the frequent itemsets in D at $\alpha = 0.4$ are given as follows: $\{a, b\}(0.5)$, $\{a, c\}(0.4)$. The association rules in D at $\beta = 0.4$ are given in Table 1.2.5.

Table 1.2.5. Association rules in D at $\alpha = 0.4$ and $\beta = 0.4$

Association rule r	$supp(r, D)$	$conf(r, D)$
$\{a\} \rightarrow \{b\}$	0.5	0.71429
$\{b\} \rightarrow \{a\}$	0.5	0.625
$\{a\} \rightarrow \{c\}$	0.4	0.57143
$\{c\} \rightarrow \{a\}$	0.4	0.8

The set of frequent itemsets synthesized from the above association rules is equal to $\{\{a\}(0.7), \{b\}(0.8), \{c\}(0.5), \{a, b\}(0.5), \{a, c\}(0.4)\}$. The conditional patterns in D at $\delta = 0.4$ are given in Table 1.2.6.

Table 1.2.6. Conditional patterns in D at $\alpha = 0.4$ and $\delta = 0$

Conditional pattern	*csupp*	*rsupp*	Conditional pattern	*csupp*	*rsupp*
$\langle\{a\}, \{a, b\}\rangle$	0.2	0.5	$\langle\{a\}, \{a, c\}\rangle$	0.3	0.4
$\langle\{b\}, \{a, b\}\rangle$	0.3	0.5	$\langle\{c\}, \{a, c\}\rangle$	0.1	0.4

The set of frequent itemsets synthesized from the above conditional patterns is equal to $\{\{a\}(0.7), \{b\}(0.8), \{c\}(0.5), \{a, b\}(0.5), \{a, c\}(0.4)\}$. Thus, the set of frequent itemsets synthesized from the above association rules and the set of frequent itemsets synthesized from the above conditional patterns are the same at $\beta = \alpha$ and $\delta = 0$. Thus, it validates Lemma 1.2.5.

Lemma 1.2.6. *Let the conditional pattern $\langle Y, X\rangle$ in database D be interesting at conditional support level δ and support level α. Then itemset Y is frequent at level $\alpha + \delta$.*

Proof. $csupp\langle Y, X, D\rangle \geq \delta$ and $supp(X, D) \geq \alpha$, since $\langle Y, X\rangle$ is interesting in D at conditional support level δ and support level α. The patterns X and $\langle Y, X\rangle$ in D cannot occur in a transaction simultaneously. $supp(X, D) \geq \alpha$ implies $supp(Y, D) \geq \alpha$, since $Y \subseteq X$. Also, $csupp\langle Y, X, D\rangle \geq \delta$ and thus, $supp(Y, D) \geq (\alpha + \delta)$. •

With reference to Example 1.2.3, $\langle\{b\}, \{a, b, c\}\rangle$ is interesting conditional pattern in D at $\delta = 0.2$ and $\alpha = 0.2$. With reference to Example 1.2.1, $supp(\{b\}, D) = 0.8 \geq 0.2 + 0.2 = 0.4$. Thus, it validates Lemma 1.2.6.

Lemma 1.2.7. *Let $X_1, X_2, ..., X_m$ be itemsets in database D such $X_i \subseteq X_{i+1}$, for i = 1, 2, ..., m-1. Then, $csupp\langle Y, X_i, D\rangle \geq csupp\langle Y, X_{i+1}, D\rangle$, for $Y \subseteq X_i$ at every i = 1, 2, ..., m-1.*

Proof. Let $Y = \{a_1, a_2, \ldots, a_p\}$. For $i = k$, let $Z = X_{k+1}-X_k$. Also let, $X_k = \{b_1, b_2, \ldots, b_q\}$, and $Z = \{c_1, c_2, \ldots, c_r\}$. Consider the following two Boolean expressions: $E_1 = a_1 \wedge a_2 \wedge \ldots \wedge a_p \wedge \neg b_1 \wedge \neg b_2 \wedge \ldots \wedge \neg b_q$ and $E_2 = a_1 \wedge a_2 \wedge \ldots \wedge a_p \wedge \neg b_1 \wedge \neg b_2 \wedge \ldots \wedge \neg b_q \wedge \neg c_1 \wedge \neg c_2 \wedge \ldots \wedge \neg c_r$. The Boolean expressions E_1 and E_2 correspond to conditional patterns

$\langle Y, X_k\rangle$ and $\langle Y, X_{k+1}\rangle$, respectively. The expression E_2 is more restrictive than the expression E_1. Thus, *supp*(E_1, D) $\geq$ *sup*(E_2, D). •

With reference to database D of Example 1.2.1, let $Y = \{b\}$, $X_1 = \{a, b\}$ and $X_2 = \{a, b, c\}$. We have $csupp\langle Y, X_1, D\rangle = 0.3$ and $csupp\langle Y, X_2, D\rangle = 0.2$. We observe that $csupp\langle Y, X_1, D\rangle \geq csupp\langle Y, X_2, D\rangle$.

1.2.4 Mining conditional patterns

For mining conditional patterns in a database, we need to find their conditional supports. We calculate $csupp\langle Y, X, D\rangle$ in terms of supports of relevant frequent itemsets, for $Y \subseteq X$. Let $X = Y \bigcup Z$, where $Z = \{a_1, a_2, \ldots, a_p\}$. The following theorem [9] is useful for synthesizing conditional supports using relevant frequent itemsets in D.

Lemma 1.2.8. *Let X, Y and Z are itemsets in database D such that* $X = Y \bigcup Z$, *where* $Z = \{a_1, a_2, \ldots, a_p\}$. *Then,* $csupp\langle Y, X, D\rangle = supp(Y, D) - \sum_{i=1}^{p} supp(Y \bigcap \{a_i\}, D) + \sum_{i<j;i,j=1}^{p} supp(Y \bigcap \{a_i, a_j\}, D) - \sum_{i<j<k;i,j,k=1}^{p} supp(Y \bigcap \{a_i, a_j, a_k\}, D) + \ldots + (-1)^p \times supp(Y \bigcap \{a_1, a_2, \ldots, a_p\}, D)$ (1.2.4)

Proof. We shall prove the result using the method of induction on p. For $p = 1$, $X = Y \bigcap \{a_1\}$. Then, $csupp\langle Y, X, D\rangle = supp(Y, D) - \text{supp}(Y \bigcap \{a_1\}, D)$. Thus, the result is true for $p = 1$. Let us assume that the result is true for $p = m$.

We shall prove that the result is true for $p = m + 1$. Let $Z = \{a_1, a_2, \ldots, a_{m+1}\}$. Due to the addition of item a_{m+1}, many supports are required to be added to or, subtracted from the expression of $csupp\langle Y, X, D\rangle$ at $p = m$. For example, $supp(Y \bigcap \{a_{m+1}\}, D)$ is required to be subtracted, $supp(Y \bigcap \{a_i, a_{m+1}\}, D)$ is required to be added, for $1 \leq i \leq m$, and so on. Finally, the term $(-1)^{m+1} \times \text{supp}(Y \bigcap \{a_1, a_2, \ldots, a_{m+1}\}, D)$ is required to be added. Thus, the expression of $csupp\langle Y, X, D\rangle$ at $p = m + 1$, is given as follows.

$csupp\,\langle Y, X, D\rangle = supp(Y, D) - \sum_{i=1}^{m+1} supp\,(Y \cap \{a_i\}, D) + \sum_{i<j; i,j=1}^{m+1} supp(Y \cap \{a_i, a_j\}, D) - \sum_{i<j<k; i,j,k=1}^{m+1} supp(Y \cap \{a_i, a_j, a_k\}, D) + \ldots + (-1)^{m+1} \times supp\,(Y \cap \{a_1, a_2, \ldots, a_{m+1}\}, D)$. •

Formulas (1.2.1) and (1.2.2) validate above theorem. We shall use this formula in the algorithm for computing conditional support of a conditional pattern.

Lemma 1.2.9. *The maximum number of non-trivial conditional patterns is equal to* $\sum_{X \in FIS(D);\, |X| \geq 2} 2^{|X|} - 2$, *where FIS(D) is the set of frequent itemsets in database D.*

Proof. The number of nonempty subsets of X excluding X is equal to $2^{|X|} - 2$. Each such subset of X corresponds to a non-trivial conditional pattern with reference to X. Thus, the lemma follows. •

The interestingness of a conditional pattern is judged by its conditional support and reference support. By combining both the measures one could define many interestingness measures of a conditional pattern. An appealing measure of interestingness of a conditional pattern $\langle Y, X\rangle$ in database D could be $csupp\langle Y, X, D\rangle$ + $rsupp\langle Y, X, D\rangle$.

1.2.4.1 Algorithm design

For mining conditional patterns in a database, we make use of an existing frequent itemset mining algorithm [13], [39], [66]. There are two approaches of mining conditional patterns in a database.

In the first approach, one could synthesize conditional patterns from current frequent itemset extracted during the mining process. As soon as a frequent itemset is found during the mining process, one could call an algorithm for finding conditional patterns using the current frequent itemset. When a frequent itemset is extracted, then all the non-null subsets of the frequent itemest have already been extracted. Thus, one could synthesize all the conditional patterns from the current frequent itemset extracted from the database. In the second approach, one could synthesize conditional patterns from the frequent itemsets in the given database after mining of all frequent itemsets. Thus, all the frequent itemsets are processed at the end of mining task. These two approaches seem to be the same so far as the computational complexity is concerned. In this chapter, we have followed the second approach of synthesizing conditional patterns. During the process of mining frequent itemsets, the frequent itemsets of smaller size get extracted before the frequent itemsets of larger size. The frequent itemsets are stored in array *FIS* and get sorted based on their size automatically. During the processing of current frequent itemset, all the non-null subsets are available before the current itemset in *FIS.*

Before presenting the proposed algorithm of synthesizing the conditional patterns, we first state how we have designed the synthesizing algorithm. The frequent itemsets of size 1 generate trivial conditional patterns. Thus, the algorithm skips processing frequent itemsets of size 1. There are $2^{|X|}-1$ non-null subsets of an itemset X. Each non-null subset of X may correspond to an interesting conditional pattern, for $|X| \geq 2$. The subset X of X corresponds to a trivial conditional pattern. Thus, we need to process $2^{|X|}-2$ subsets of X.

One could view a conditional pattern as an object having the following attributes: *pattern*, *reference*, *csupp*, and *rsupp*. We use an array *CP* to store conditional patterns in a database. The *y* attribute of *i*-th conditional pattern is accessed by notation *CP(i).y*. Also, a frequent itemset could be viewed as an object described by the following attributes: *itemset* and *supp*. Let *N* be the number of frequent itemsets in the given database *D*. The variables *i* and *j* are used to index the frequent itemset being processed and the conditional pattern being synthesized, respectively. An algorithm [9] for synthesizing interesting non-trivial conditional patterns is presented below.

Algorithm 1.2.1. Synthesize interesting non-trivial conditional patterns in a database.

procedure *ConditionalPatternSynthesis* (*N*, *FIS*)

Input:

N: number of frequent itemsets in the given database

FIS: array of frequent itemsets in the given database

Output:

Interesting non-trivial conditional patterns in the database

```
01: let $i = 1$;
02: let $j = 1$;
03: while ($|FIS(i)| = 1$) do
04:     increase $i$ by 1;
05: end while
06: while ($i \leq N$) do
07:     $CP(j).rsupp = FIS(i).supp$; $CP(j).reference = FIS(i).itemset$;
08:     let $sum = 0$;
09:     for $k = 1$ to $(2^{|FIS(i).itemset|} - 1)$ do
10:         let $tempItemset = k$-th subset of $FIS(i).itemset$;
11:         if ($FIS(i).itemset = tempItemset$) then go to line 24; end if
12:         let $kk = 1$;
13:         while ($kk \leq i$) do
14:             if ($FIS(kk).itemset = tempItemset$) then
15:                 $sum = sum + (-1)^{|FIS(kk).itemset| - |tempItemset|} \times FIS(kk).supp$;
16:                 go to line 21;
17:             end if
18:             increase $kk$ by 1;
19:         end while
20:     end for
21:     if ($sum \geq \delta$) then
22:         $CP(i).csupp = sum$; $CP(i).pattern = tempItemset$;
23:         increase $j$ by 1;
24:     end if
25:     increase $i$ by 1;
26: end while
```

```
27: sort conditional patterns on (csupp + rsupp) in non-increasing order;
28: for $k = 1$ to $j$ do
29:     display $k$-th conditional pattern;
30: end for
end procedure
```

In this paragraph, we explain and justify the statements of the above algorithm. The important parts of the algorithm are explained as follows. The frequent itemsets of size 1 generate trivial conditional patterns. Thus, we have skipped processing frequent itemsets of size one using lines 3-5. We synthesize conditional patterns using lines 6-26. There are $2^{|X|}-1$ non-null subsets for an itemset X. Each subset is considered using a for-loop in lines 9-20. The algorithm synthesizes conditional patterns with reference to a frequent itemset X, for $|X| \geq 2$. The algorithm bypasses processing itemset Y, if $Y = X$. When we synthesize conditional patterns with reference to a frequent itemset, we have already finished synthesizing its subsets. All the non-null subsets appear on or before the frequent itemset in *FIS*. Thus, if a frequent itemset X located at position i, then we search for a subset of X from index 1 to i in *FIS*, since *FIS* is sorted in non-decreasing order on length of an itemset. Thus, it justifies the condition of while loop at line 13. Formula (1.2.4) expresses $csupp\langle Y, X, D\rangle$ in terms of $\text{supp}(Y \cap Z, D)$, for all $Z \subseteq X\text{-}Y$. The coefficient of $supp(Y \cap Z, D)$ is $(-1)^{|Z|}$ in the expression of $csupp\langle Y, X, D\rangle$. Thus, $csupp\langle Y, X, D\rangle = \sum_{Z \subseteq X\text{-}Y} (-1)^{|Z|} \times csupp\langle Y, X, D\rangle$. This formula has been applied at line 15 to calculate $csupp\langle Y, X, D\rangle$. A conditional pattern is interesting if the conditional support is greater than or equal to δ, provided the reference support of the itemset is greater than or equal to α. We need not check the reference support, since we deal with the frequent itemsets. In line 21, we check whether the currently synthesized conditional pattern is interesting. The details of a synthesized conditional pattern are stored using lines 7 and 22. At line 27, we sort all interesting conditional patterns in the given database. Finally, we display interesting conditional patterns using lines 28-30. We calculate time complexity of algorithm *ConditionalPatternSynthesis* using Lemma 1.2.10.

Lemma 1.2.10. *Algorithm ConditionalPatternSynthesis executes in* $O(N^2 \times 2^p)$ *time, where N and p are the number of frequent itemsets and the average size of the frequent itemsets of size greater than 1 in the database, respectively.*

Proof. Lines 3-5 take $O(N)$ time. The while-loop at line 6 repeats maximum N times. Thus, the for-loop at line 9 repeats 2^p-1 times. The while-loop at line 13 repeats maximum N times. Thus, the time complexity of lines 6-26 is equal to $O(N^2 \times 2^p)$. The time complexity of line 27 is equal to $O(N \times 2^p \times \log(N \times 2^p))$, since the number of conditional patterns is equal to $O(N \times 2^p)$. The time complexity of lines 28-30 is equal to $O(N \times 2^p)$. Therefore, the time complexity of the algorithm is *maximum* $\{O(N^2 \times 2^p), O(N \times 2^p \times \log(N \times 2^p))\}$. •

1.2.5 Experiments

We have carried out several experiments to study the effectiveness of our approach. All the experiments have been implemented on a 1.6 GHz Pentium processor with 256 MB of memory using visual C++ (version 6.0) software. We present experimental results using three real databases. Database *retail* [34] is obtained from an anonymous Belgian retail supermarket store. Databases *BMS-Web-Wiew-1* and *BMS-Web-Wiew-2* can be found from KDD CUP 2000 [34]. They are processed for the purpose of conducting experiments. We present some characteristics of these databases in Table 1.2.7.

Table 1.2.7. Database characteristics

Database	# transaction	Avg length of a transaction	Avg frequency of an item	# items
retail	88,162	11.305755	99.673800	10000
BMS-Web-Wiew-1	1,49,639	2.000000	155.711759	1922
BMS-Web-Wiew-2	3,58,278	2.000000	7165.560000	100

Top five interesting conditional patterns of available categories are shown in Table 1.2.8. We have implemented apriori algorithm for the purpose of mining conditional patterns in the given databases. The conditional patterns in a database are ranked based on the sum of conditional support and reference support.

Table 1.2.8. Top 5 conditional patterns of each category available in *retail* at $\alpha = 0.05$ and $\delta = 0.03$

Conditional pattern	*csupp*	*rsupp*
$\langle\{39\}, \{1, 39\}\rangle$	0.520451	0.066332
$\langle\{39\}, \{8, 39\}\rangle$	0.524421	0.062362
$\langle\{39\}, \{0, 39\}\rangle$	0.526871	0.059912
$\langle\{39\}, \{2, 39\}\rangle$	0.525612	0.061171
$\langle\{39\}, \{3, 39\}\rangle$	0.525714	0.061069
$\langle\{39\}, \{39, 41, 48\}\rangle$	0.210317	0.083551
$\langle\{39\}, \{32, 39, 48\}\rangle$	0.221603	0.061274
$\langle\{39\}, \{38, 39, 48\}\rangle$	0.208106	0.069213
$\langle\{48\}, \{39, 41, 48\}\rangle$	0.139482	0.083551
$\langle\{32\}, \{32, 39, 48\}\rangle$	0.049432	0.061274
$\langle\{39,48\}, \{32, 39, 48\}\rangle$	0.269277	0.061274
$\langle\{39, 48\}, \{39, 41, 48\}\rangle$	0.247000	0.083551
$\langle\{39, 48\}, \{38, 39, 48\}\rangle$	0.261337	0.069213
$\langle\{38, 39\}, \{38, 39, 48\}\rangle$	0.048127	0.069213
$\langle\{32, 39\}, \{32, 39, 48\}\rangle$	0.034629	0.061274

Table 1.2.9. Top 5 conditional patterns of each category available in *BMS-Web-Wiew-1* at $\alpha = 0.01$ and $\delta = 0.009$

Conditional pattern	*csupp*	*rsupp*
$\langle\{5\}, \{1, 5\}\rangle$	0.235453	0.013740
$\langle\{5\}, \{3, 7\}\rangle$	0.236135	0.013058
$\langle\{5\}, \{5, 7\}\rangle$	0.235293	0.013900
$\langle\{5\}, \{5, 9\}\rangle$	0.236335	0.012858
$\langle\{7\}, \{7, 9\}\rangle$	0.203563	0.011568

Table 1.2.10. Top 5 conditional patterns of each category available in *BMS-Web-Wiew-2* at $\alpha = 0.009$ and $\delta = 0.007$

Conditional pattern	*csupp*	*rsupp*
$\langle\{7\}, \{1, 7\}\rangle$	0.174072	0.022943
$\langle\{7\}, \{6, 7\}\rangle$	0.185401	0.011614
$\langle\{7\}, \{7, 9\}\rangle$	0.175810	0.021204
$\langle\{7\}, \{0, 7\}\rangle$	0.185702	0.011312
$\langle\{7\}, \{2, 7\}\rangle$	0.185747	0.011268

In both *BMS-Web-Wiew-1* and *BMS-Web-Wiew-2*, only one category of conditional patterns is available, since the maximum length of a transaction in each of these two databases is 2.

We have also conducted experiments for execution time needed for finding conditional patterns in different databases. The execution time for finding conditional patterns in a database increases as the size, i.e., the number of transactions contained in a database increases. We observe this phenomenon in Figures 1.2.2 and 1.2.3. We have also conducted experiments for execution time needed for synthesizing conditional patterns in a database. The time required for synthesizing conditional patterns in each of the above databases is equal to zero millisecond at the respective values of α and δ shown in Tables 1.2.8, 1.2.9 and 1.2.10.

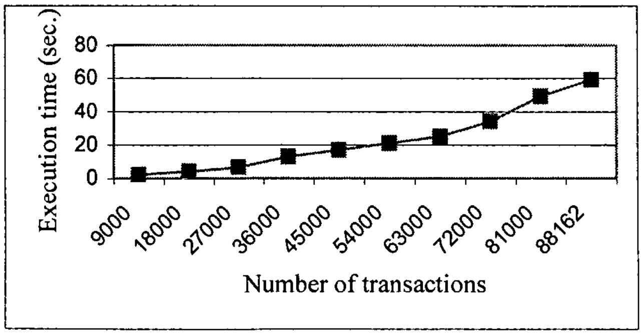


Figure 1.2.2. Execution time versus the number of transactions in *retail*

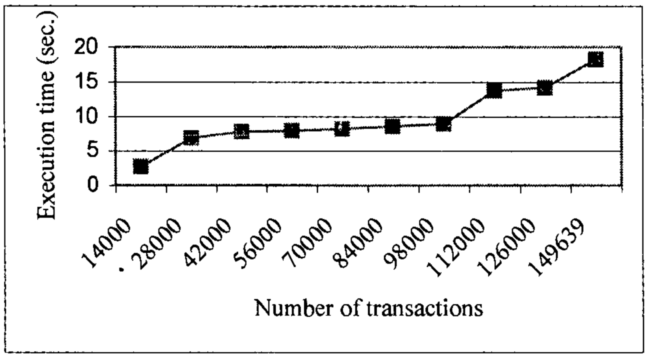


Figure 1.2.3. Execution time versus the number of transactions in *BMS-Web-Wiew-1*

We have also conducted experiments for finding the number of conditional patterns in a database at a given α. The number of conditional patterns in a database decreases as α increases. We observe this phenomenon in Figures 1.2.4 and 1.2.5.

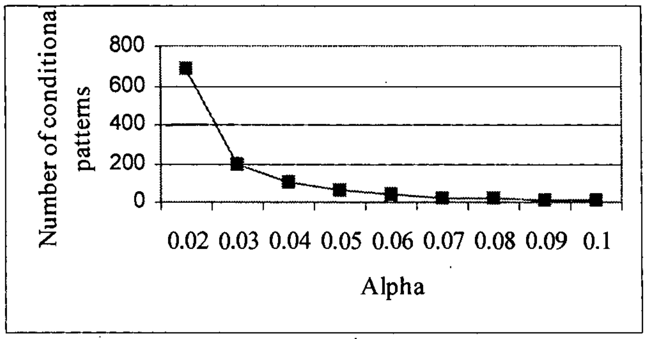


Figure 1.2.4. Number of conditional patterns versus α for *retail*

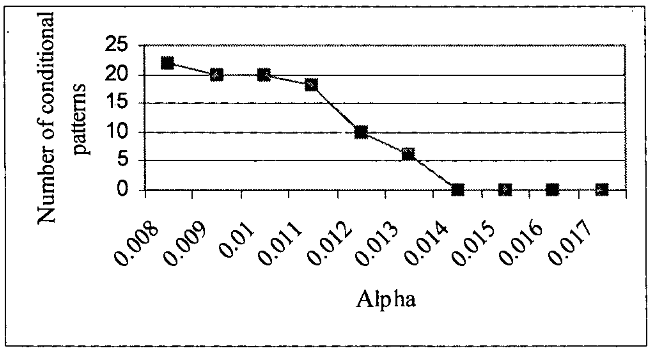


Figure 1.2.5. Number of conditional patterns versus α for *BMS-Web-Wiew-1*

We have also conducted experiments for finding execution time needed for mining conditional patterns in a database at a given α. The execution time needed for mining conditional patterns in a database decreases as α increases. We observe this phenomenon in Figures 1.2.6 and 1.2.7.

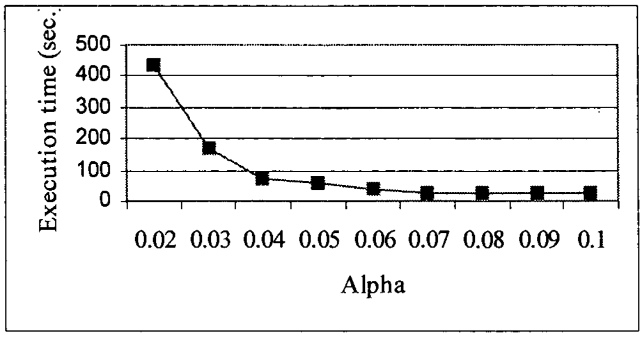


Figure 1.2.6. Execution time versus α for *retail*

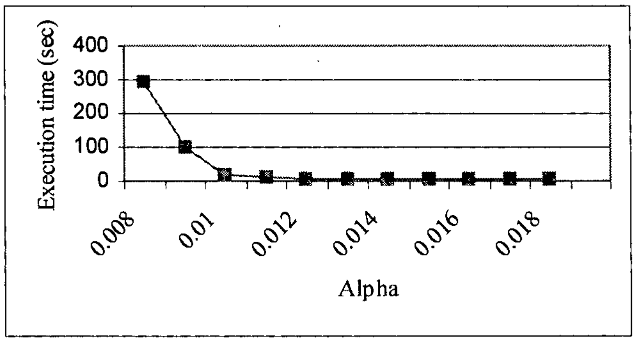


Figure 1.2.7. Execution time versus α for *BMS-Web-Wiew-1*

Also, we have conducted experiments to study the relationship between the size of a database and the number of conditional patterns in it. The experiments are conducted on databases *retail* and *BMS-Web-Wiew-1*. The results of the experiments are shown in Figures 1.2.8 and 1.2.9. From the graphs in Figures 1.2.8 and 1.2.9, we could conclude that there is no universal relationship between the size of a database and the number of conditional patterns in it.

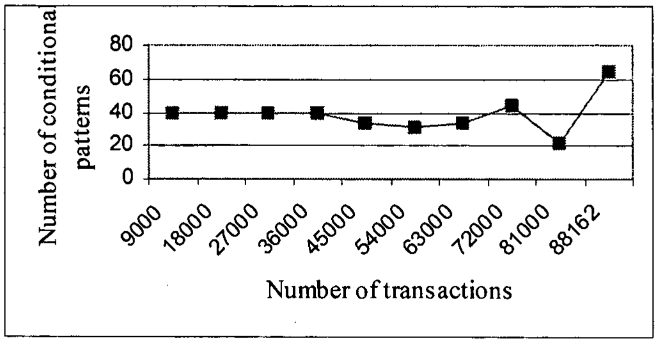


Figure 1.2.8. Number of conditional patterns versus the number of transactions in *retail*

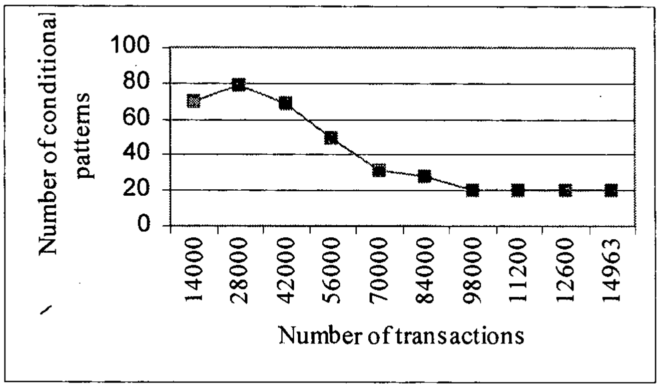


Figure 9. Number of conditional patterns versus the number of transactions in *BMS-Web-Wiew-1*

Also, we have conducted experiments to study the relationship between the number of conditional patterns and conditional support. The experiments have been conducted on databases *retail* and *BMS-Web-Wiew-1*. The number of conditional patterns in a database decreases as δ increases. We observe this phenomenon in Figures 1.2.10 and 1.2.11.

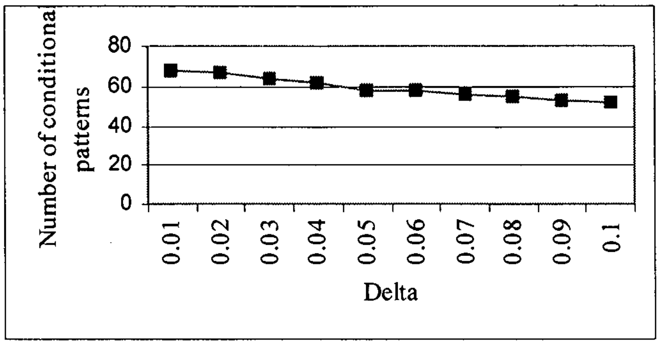


Figure 1.2.10. Number of conditional patterns versus δ for *retail*

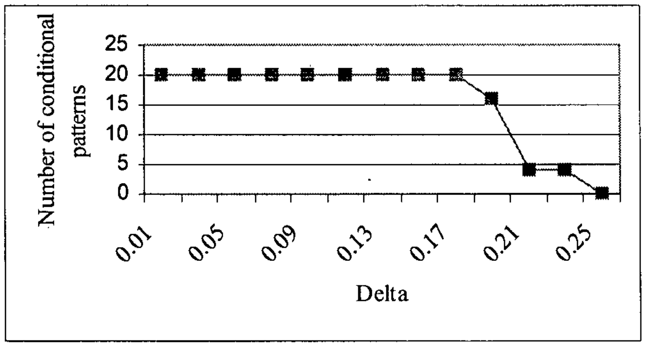


Figure 1.2.11. Number of conditional patterns versus δ for *BMS-Web-Wiew-1*

1.2.5.1 An application

Adhikari and Rao [3] have proposed a technique for mining arbitrary Boolean expressions induced by frequent itemsets using conditional patterns in a database. In Chapter 1.3, we have discussed how an arbitrary Boolean expression induced by frequent itemsets could be synthesized using conditional patterns in a database.

1.2.6 Related work

Agrawal et al. [11] introduced association rule and support-confidence framework and an algorithm to mine frequent itemsets. The algorithm is sometimes called AIS after the authors' initials. Since then, many algorithms have been reported to generate association database. Conditional patterns reveal more characteristics of a database. Also, we have observed that conditional patterns store significant nuggets of knowledge about a database that are not immediately available from frequent itemsets and association rules.

Chapter 1.3

A framework for synthesizing arbitrary Boolean expressions induced by frequent itemsets

An itemset could be thought as a basic type of pattern in a transactional database. Itemset patterns influence heavily KDD research in the following ways: Firstly, many interesting algorithms have been reported on mining itemset patterns in a database [11], [39], [66]. Secondly, many patterns are defined based on the itemset patterns in a database. They may be called as derived patterns. For example, positive association and negative association rules are examples of derived patterns. A good amount of work has been reported on mining / synthesizing such derived patterns in a database [13], [17], [81]. Thirdly, solutions of many problems are based on the analysis of patterns in a database. Such applications [79], [83] process patterns in a database for the purpose of making some decisions. Thus, mining and analysis of itemset patterns in a database is an interesting as well as important issue. Also, mining Boolean expressions induced by frequent itemsets could lead to significant nuggets of knowledge, with many potential applications in market basket data analysis, web usage mining, social network analysis and bioinformatics.

The *support* [11] of an itemset X in database D could be defined as the fraction of transactions in D containing all the items of X, denoted by $supp(X, D)$. The importance of an itemset could be judged by its support. Itemset X is *frequent* in D if $supp(X, D) \geq$ *minimum support* (α). Let $FIS(D)$ be the set of frequent itemsets in D. Frequent itemsets determine major characteristics of a database. Wu et al. [80] have proposed a solution of inverse frequent itemset mining. These authors argued that one could efficiently generate a synthetic market basket database from the frequent itemsets and their supports. Let X and Y be two itemsets in D. The characteristics of D are revealed more by the pair (X, *supp*(X, D)) than that of (Y, *supp*(Y, D)), if *supp*(X, D) > *supp*(Y, D). Thus, it is important to study frequent itemsets more than infrequent itemsets. In this chapter, we propose a framework for synthesizing arbitrary Boolean expressions induced by frequent itemsets in D. The proposed framework for synthesizing Boolean expressions is based on conditional patterns in D. In Chapter 1.2, we have presented the notion of conditional pattern in a database.

Let $X = \{a_1, a_2, \ldots, a_m\}$ be a set of m binary variables (items). Let $\vee$, $\wedge$ and $\neg$ denote the usual AND, OR and NOT operators in Boolean algebra, respectively. An arbitrary *Boolean expression induced by X* could be constructed using the following steps:

(i) a_i is a Boolean expression, for $i = 1, 2, \ldots, m$.

(ii) If a_i is a Boolean expression then $\neg a_i$ is a Boolean expression, for $i = 1, 2, \ldots, m$.

(iii) If a_i and a_j are Boolean expressions then $(a_i \vee a_j)$ and $(a_i \wedge a_j)$ are Boolean expressions, for $i, j = 1, 2, \ldots, m$.

(iv) Any expression obtained by applying steps (i), (ii) and (iii) finite number of times is a Boolean expression.

Our objective is not to give a formal definition of a well formed Boolean expression induced by X, but to understand how a Boolean expression induced by X could be constructed using a step-by-step approach. There are some other non-fundamental operators in Boolean algebra. Some examples of non-fundamental operators are NAND, NOR, and XOR. Any Boolean expression could be expressed by the set of operators $\{\neg, \wedge, \vee\}$. Thus, it is a functionally complete set of operators. Using De Morgan's laws, one could show that $\{\neg, \wedge\}$ and $\{\neg, \vee\}$ are the minimal sets of operators by which any Boolean function could be expressed. Thus, $\{\neg, \wedge\}$ and $\{\neg, \vee\}$ are also functionally complete sets of operators. An elaborate discussion on Boolean algebra could be found in Gregg [37].

The pattern itemset of a conditional pattern with reference to itemset $X = \{a_1, a_2, \ldots, a_m\}$ is of the form $b_1 \wedge b_2 \wedge \ldots \wedge b_m$, where $b_i = a_i$, or $\neg a_i$, for $i = 1, 2, \ldots, m$. Let $\psi(X)$ be the set of all such pattern itemsets with reference to X. $\psi(X)$ is called the *generator* of Boolean expressions induced by X. $\psi(X)$ contains 2^m-1 pattern itemsets. A pattern itemset of the corresponding conditional pattern is also called a *minterm*, or *standard product*. Every Boolean expression of items of X could be constructed using pattern itemsets in $\psi(X)$ (as mentioned in Lemma 1.3.3). In particular, let $X = \{a, b, c\}$. Then, $\psi(X) = \{a \wedge b \wedge c,\ a \wedge b \wedge \neg c,\ a \wedge \neg b \wedge c,\ a \wedge \neg b \wedge \neg c,\ \neg a \wedge b \wedge c,\ \neg a \wedge b \wedge \neg c,\ \neg a \wedge \neg b \wedge c\}$. The Boolean expression $\neg b \wedge c$ could be re-written as $(a \wedge \neg b \wedge c) \vee (\neg a \wedge \neg b \wedge c)$. Every Boolean expression can be expressed as a sum of some pattern itemsets of the corresponding generator. A Boolean expression expressed as a sum of pattern itemsets is said to be in *canonical form*. Each pattern itemset corresponds to a set of transactions in D. In the following, we show how each pattern itemset with reference to $\{a, b, c\}$ corresponds to a set of transactions in D.

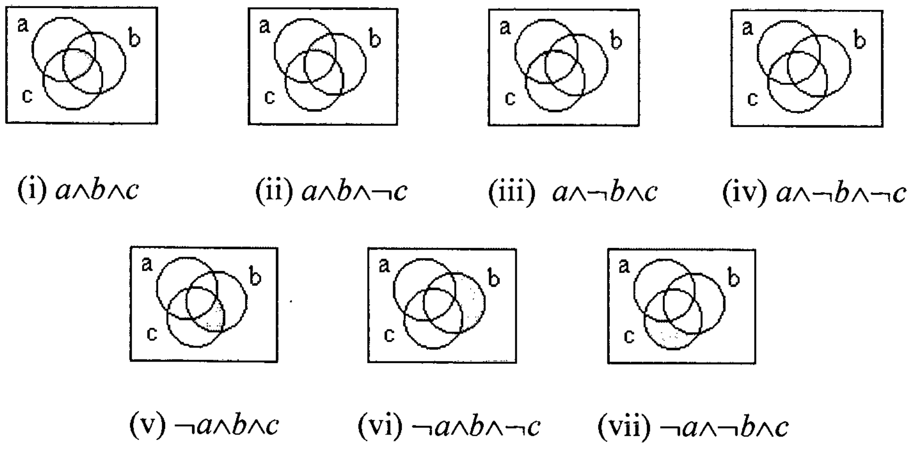

(i) $a \wedge b \wedge c$
(ii) $a \wedge b \wedge \neg c$
(iii) $a \wedge \neg b \wedge c$
(iv) $a \wedge \neg b \wedge \neg c$
(v) $\neg a \wedge b \wedge c$
(vi) $\neg a \wedge b \wedge \neg c$
(vii) $\neg a \wedge \neg b \wedge c$

Figure 1.3.1. Generator of $\{a, b, c\}$

is better to mine the generator of an itemset and synthesize the desired Boolean expressions afterwards. Zhao et al. [95] have proposed BLOSOM framework for mining arbitrary Boolean expressions. The framework suffers from the following limitations:

- It does not handle NOT operator.
- Let $\{a, b, c\}$ be a frequent itemset of our interest. We wish to mine some functions induced by $\{a, b, c\}$. It proposes a framework to mine minimal generators of (i) closed OR-clauses, (ii) closed AND-clauses, (iii) closed maximal min-DNF, and (iv) closed maximal min-CNF. It requires establishing a mapping from the space of minimal generators to the space of arbitrary Boolean expressions, so that we could study the desired Boolean expressions induced by $\{a, b, c\}$. Thus, BLOSOM might not provide the knowledge of Boolean expression that we wish to study.
- A specific framework for a specific type of Boolean expressions is introduced.

Therefore, we propose here a simple and elegant approach for synthesizing arbitrary Boolean expressions induced by frequent itemsets. We state our problem as follows.

We are given a database D of customer transactions. Mine all the members of $\psi(X)$, *for all* $X \in FIS(D)$ *such that* $|X| \geq 2$.

The rest of the chapter is organized as follows. We discuss related results in Section 1.3.2. In Section 1.3.3, we propose an algorithm for mining members of different generators. The results of the experiments are presented in Section 1.3.4. We discuss related work in Section 1.3.5.

1.3.2 Related results

In this section, we discuss a few results related to discussion held in previous section.

Lemma 1.3.1. *Let E be the event that a transaction contains at least one item of the itemset X in database D. Then, the support of event E in D,* $supp(E, D) = \sum_{Y \subseteq X, Y \neq \phi} csupp\langle Y, X, D\rangle$.

Proof. The number of non-null subsets of an itemset X is $2^{|X|} - 1$, for $X \in FIS(D)$. Each non-null subset of X corresponds to a conditional pattern. Thus, the result follows. •

1.3.3 Synthesizing generators

Conditional patterns are derived from the frequent itemsets in a database. Let X be a frequent itemset in D. We shall express $csupp\langle Y, X, D\rangle$ in terms of the supports of the frequent itemsets in D, for $Y \subseteq X$. Without loss of generality, let $X = Y \cup Z$, where $Z = \{a_1, a_2, \ldots, a_m\}$. The conditional support of i-th conditional pattern with reference to X is same as the support of i-th member of $\psi(X)$, for $i = 1, 2, \ldots, 2^{|X|} - 1$. Thus, the following theorem [3] enables us to compute the supports of members of $\psi(X)$ in D, for all $X \in FIS(D)$, and $|X| \geq 2$.

Lemma 1.3.5. *Let X, Y and Z are itemsets in database D such that $X = Y \cup Z$, where $Z = \{a_1, a_2, \ldots, a_m\}$. Then,* $csupp\langle Y, X, D\rangle = supp(Y, D) - \sum_{i=1}^{m} supp(Y \cap \{a_i\}, D) + \sum_{i<j;\, i,j=1}^{m} supp(Y \cap \{a_i, a_j\}, D) - \sum_{i<j<k;\, i,j,k=1}^{m} supp(Y \cap \{a_i, a_j, a_k\}, D) + \ldots + (-1)^m \times supp(Y \cap \{a_1, a_2, \ldots, a_m\}, D)$ (1.3.2)

Proof. Please refer Lemma 1.2.8. •

1.3.3.1 Algorithm design

For synthesizing arbitrary Boolean expressions induced by frequent itemsets in a database, we make use of an existing frequent itemset mining algorithm. We synthesize only the generator of Boolean expressions induced by a frequent itemset. The generator of Boolean expressions induced by the frequent itemset X contains $2^{|X|}-1$ pattern itemsets. The proposed algorithm synthesizes all the members of all the generators. There are two approaches for synthesizing generators of Boolean expressions induced by frequent itemsets in a database. In the first approach, one could synthesize the generator from the current frequent itemset. As soon as a frequent itemset is extracted, one could call an algorithm for synthesizing members of the corresponding generator. When a frequent itemset is found, then all the non-null subsets of this frequent itemest have already been extracted. Thus, one could synthesize all the members of the generator from the frequent itemsets extracted so far. In the second approach, one could synthesize members of the different generators after mining all the frequent itemsets. Thus, all the frequent itemsets are processed after the mining task. These two approaches seem to be the same so far as the computational complexity is concerned. In this chapter, we have followed the second approach of synthesizing members of different generators. During the process of mining frequent itemsets, the frequent itemsets of smaller size get extracted before the frequent itemsets of larger size. The frequent itemsets are kept in array *FIS*. During the processing of current frequent itemset, all the non-null subsets are available before the current itemset in *FIS*.

There are $2^{|X|} - 1$ non-null subsets of an itemset X. Each non-null subset of X corresponds to a conditional pattern, and hence, it corresponds to a member of the generator of Boolean expressions induced by X. The subset X of X corresponds to a trivial conditional pattern, and gets mined during the mining of frequent itemsets in D. Thus, we need to process $2^{|X|} - 2$ subsets of X.

One could view each conditional pattern as an object having the following attributes: *pattern*, *reference*, *csupp*, and *rsupp*. We use an array *CP* to store the conditional patterns in a database. The reference attribute of the i-th conditional pattern is accessed by the notation *CP*(i).*reference*. Similar notations are used to access other attributes of a conditional pattern. Also, each frequent itemset could be viewed as an object with the following attributes: *itemset* and *supp*. Let N be the number of frequent itemsets in the given database D. The following algorithm *SynthesizingGenerators* [3] synthesizes all the members of $\psi(X)$, for $X \in FIS(D)$.

Algorithm 1.3.1. Synthesize all the members of generator corresponding to each itemset in *FIS*(D).

procedure *SynthesizingGenerators* (*N*, *FIS*)

Input:

N: number of frequent itemsets in the given database

FIS: set of frequent itemsets in the given database

Output:

Generators corresponding to the frequent itemsets

```
1:  let $i = 1$;
2:  let $j = 0$;
3:  while ($i \le N$) do
4:    $CP(j).rsupp = FIS(i).supp$; $CP(j)$.reference = $FIS(i).itemset$;
5:    let $sum = 0$;
6:    for $k = 1$ to $(2^{|FIS(i).itemset|} - 1)$ do
7:      let $tempItemset$ = $k$-th subset of $FIS(i).itemset$;
8:      if ($FIS(i).itemset = tempItemset$) then
9:        $sum = FIS(i).supp$; go to line 19;
10:     end if
11:     let $kk = 1$;
12:     while ($kk \le i$) do
13:       if ($FIS(kk).itemset = tempItemset$) then
14:         $sum = sum + (-1)^{|FIS(kk).itemset| - |tempItemset|} \times FIS(kk).supp$;
15:         go to line 19;
16:       end if
17:       increase $kk$ by 1;
18:     end while
19:     $CP(j).csupp = sum$; $CP(j).pattern = tempItemset$;
20:     increase $j$ by 1; increase $i$ by 1;
21:   end for
22: end while
```

```
23: let $t = 0$;
24: for $i = 1$ to $N$ do
25:     for $k = 1$ to $(2^{|FIS(i).itemset|} - 1)$ do
26:         display $CP(t + k)$;
27:     end for
28:     $t = t + 2^{|FIS(i).itemset|} - 1$;
29: end for
end procedure
```

Variable *i* keeps track of the current frequent itemset being processed. Variable *j* keeps track of number of conditional patterns generated. Using lines 3-22, each frequent itemset is processed. There are $2^{|X|}$-1 non-null subsets of *X*. Each non-null subset corresponds to a conditional pattern. The generator of an itemset *X* is synthesized using lines 6-21. Let *Y* be a subset of *X*. If $Y = X$ then the algorithm bypasses from processing of *Y* (line 8). When the algorithm synthesizes generator corresponding to a frequent itemset *X*, then it has already finished the processing of its non-null subsets. All the non-null subsets appear on or before *X* in the array *FIS*. Thus, if a frequent itemset *X* located at position *i*, then we search for a subset of *X* from index 1 to *i* in array *FIS*, since the array is sorted non-decreasing order on length of an itemset. Thus, it justifies the condition of the while loop at line 12. Formula (1.3.2) expresses $csupp\langle Y, X, D\rangle$ in terms of $supp(Y \cap Z, D)$, for all $Z \subseteq X\text{-}Y$. The coefficient of $supp(Y \cap Z, D)$ is $(-1)^{|Z|}$ in the expression of $csupp\langle Y, X, D\rangle$. Thus, $csupp\langle Y, X, D\rangle = \sum_{Z \subseteq X\text{-}Y} (-1)^{|Z|} \times supp(Y \cap Z, D)$. This formula has been applied at line 14 to calculate $csupp\langle Y, X, D\rangle$. We need to synthesize both interesting and non-interesting conditional patterns with reference to frequent itemsets for the solution to the given problem. Lines 25-27 display the generator corresponding to *i*-th the frequent itemset, for $i = 1, 2, \ldots, N$. The generator corresponding to *i*-th frequent itemset contains $2^{|FIS(i).itemset|}$ - 1 members (i.e., pattern itemsets), for $i = 1, 2, \ldots, N$.

Lemma 1.3.6. *Algorithm SynthesizingGenerators executes in $O(N^2 \times 2^p)$ time, where N and p are the number of frequent itemsets and average size of frequent itemsets in the database, respectively.*

Proof. The while-loop at line 3 repeats N times. The for-loop at line 6 repeats 2^p-1 times. Also, the while-loop at line 12 repeats maximum of N times. Thus, the time complexity of lines 3-22 is $O(N^2 \times 2^p)$. The time complexity of lines 24-29 is $O(N \times 2^p)$. Therefore, the time complexity of algorithm *SynthesizingGenerators* is $O(N^2 \times 2^p)$. •

1.3.3.2 Synthesizing first *k* Boolean expressions induced by top *p* frequent itemsets

Using the truth table, one could determine the algebraic forms of Boolean expressions induced by a frequent itemset. A Boolean expression could be synthesized by the members of the corresponding generator. We classify the frequent itemsets in a database into different categories. The frequent itemsets of the same size are put in the same category. We sort the frequent itemsets of each category in non-increasing order by support and top frequent itemsets in each category are considered for synthesis. We perform experiments for synthesizing first k Boolean expressions induced by top p frequent itemsets of each category.

Example 1.3.1. Let $\{a, b\}$ and $\{a, b, c\}$ be two frequent itemsets in D of size 2 and 3, respectively. We would like to determine first k Boolean expressions induced by $\{a, b\}$ and $\{a, b, c\}$. Let E_{ij} be the j-th Boolean expression induced by the frequent itemset of size i, for j = 1, 2, ..., 2^i -1, and i = 2, 3. Then, the truth tables for the first six Boolean expressions are given in Table 1.3.1.

Table 1.3.1 Truth tables for the first six Boolean expressions induced by $\{a, b\}$ and $\{a, b, c\}$

a	b	c	E_{21}	E_{22}	E_{23}	E_{24}	E_{25}	E_{26}	E_{31}	E_{32}	E_{33}	E_{34}	E_{35}	E_{36}
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	1	1	0	0	0	0	0	0
0	1	1	0	0	0	0	1	1	0	0	0	0	0	0
1	0	0	0	0	1	1	0	0	0	0	0	0	0	0
1	0	1	0	0	1	1	0	0	0	0	0	0	1	1
1	1	0	0	1	0	1	0	1	0	0	1	1	0	0
1	1	1	0	1	0	1	0	1	0	1	0	1	0	1

First six Boolean expressions are based on items of $\{a, b\}$, and the rest of the Boolean expressions are based on items of $\{a, b, c\}$. The algebraic expressions of first six Boolean expressions are given as follows: $E_{21}(a, b) = 0$, $E_{22}(a, b) = a\wedge b$, $E_{23}(a, b) = a\wedge\neg b$, $E_{24}(a, b) = a$, $E_{25}(a, b) = \neg a\wedge b$, $E_{26}(a, b) = b$; $E_{31}(a, b, c) = 0$, $E_{32}(a, b, c) = a\wedge b\wedge c$, $E_{33}(a, b, c) = a\wedge b\wedge\neg c$, $E_{34}(a, b, c) = a\wedge b$, $E_{35}(a, b, c) = a\wedge\neg b\wedge c$, $E_{36}(a, b, c) = a\wedge c$. We express E_{ij}s in terms of members of the corresponding generator. Boolean expressions E_{22}, E_{23}, and E_{25} have already been expressed in terms of the members of the concerned generator. We need not compute E_{21} and E_{31}, since $E_{21}(a, b) = E_{31}(a, b, c) = 0$. $E_{24}(a, b) = (a\wedge b)\vee(a\wedge\neg b)$, and $E_{26}(a, b) = (a\wedge b)\vee(\neg a\wedge b)$. Also, $E_{34}(a, b, c) = (a\wedge b\wedge c)\vee(a\wedge b\wedge\neg c)$, and $E_{36}(a, b, c) = (a\wedge b\wedge c)\vee(a\wedge\neg b\wedge c)$. Expressions E_{32}, E_{33} and E_{35} have already been expressed in terms of the members of the concerned generator. •

1.3.4 Experiments

We have carried out several experiments to study the effectiveness of our approach. All the experiments have been implemented on a 1.6 GHz Pentium IV with 256 MB of memory using visual C++ (version 6.0) software. We present the experimental results using three real and one synthetic databases. The database *retail* [34] is obtained from an anonymous Belgian retail supermarket store. The databases *BMS-Web-Wiew-1* and *BMS-Web-Wiew-2* can be found from KDD CUP 2000 [34]. They are processed for the purpose of conducting experiments. The database *T10I4D100K* [34] was generated using the generator from IBM Almaden Quest research group. We present some characteristics of these databases in Table 1.3.2. Among four databases of Table 1.3.2, the first three databases are real and the forth one is synthetic.

Table 1.3.2. Database characteristics

Database	# transactions	Avg length of a transaction	Avg frequency of an item	# items
retail	88,162	11.305755	99.673800	10000
BMS-Web-Wiew-1	1,49,639	2.000000	155.711759	1922
BMS-Web-Wiew-2	3,58,278	2.000000	7165.560000	100
T10I4D100K	1,00,000	11.10228	1276.12413	870

For the purpose of synthesizing Boolean expressions, we have implemented apriori algorithm [13], since it is simple and easy to implement. In Tables 1.3.3, 1.3.4, 1.3.5 and 1.3.6, we present first six Boolean expressions induced top five frequent itemsets from *retail*, *BMS-Web-Wiew-1*, *BMS-Web-Wiew-2* and *T10I4D100K*, respectively.

Table 1.3.3. First six Boolean expressions induced by top five frequent itemsets of size 2 in *retail* at $\alpha = 0.05$

Frequent itemset	*supp* (E_{22}, D)	*supp* (E_{23}, D)	*supp* (E_{24}, D)	*supp* (E_{25}, D)	*supp* (E_{26}, D)
{39, 48}	0.3306	0.2562	0.5868	0.1582	0.4888
{39, 41}	0.1295	0.4573	0.5868	0.0422	0.1717
{38, 39}	0.1173	0.0603	0.1776	0.4694	0.5868
{41, 48}	0.1023	0.0694	0.1717	0.3865	0.4888
{32, 39}	0.0959	0.0793	0.1752	0.4909	0.5868

Table 1.3.3(continued). First six Boolean expressions induced by top five frequent itemsets of size 3 in *retail* at $\alpha = 0.05$

Frequent itemset	*supp* (E_{32}, D)	*supp* (E_{33}, D)	*supp* (E_{34}, D)	*supp* (E_{35}, D)	*supp* (E_{36}, D)
{39,41,48}	0.0836	0.0459	0.1295	0.2470	0.3306
{38,39,48}	0.0692	0.0481	0.1173	0.0209	0.0901
{32,39,48}	0.0613	0.0346	0.0959	0.0299	0.0911
{1,39,48}	0.0449	0.0215	0.0663	0.0170	0.0618
{5,39,48}	0.0432	0.0197	0.0629	0.0162	0.0594

The Boolean functions E_{22} and E_{32} are not shown, since they are all equal to 0. The Boolean expressions induced by frequent itemsets of size one are not studied here.

Table 1.3. 4. First six Boolean expressions induced by top five frequent itemsets of size 2 in *BMS-Web-Wiew-1* at $\alpha = 0.01$

Frequent itemset	*supp* (E_{22}, D)	*supp* (E_{23}, D)	*supp* (E_{24}, D)	*supp* (E_{25}, D)	*supp* (E_{26}, D)
{5, 7}	0.0139	0.2353	0.2491	0.2012	0.2151
{1, 5}	0.0137	0.1761	0.1899	0.2355	0.2491
{3, 5}	0.0131	0.1953	0.2083	0.2361	0.2491
{5, 9}	0.0129	0.2363	0.2491	0.2014	0.2142
{1, 7}	0.0124	0.1774	0.1899	0.2027	0.2151

BMS-Web-Wiew-1 and *BMS-Web-Wiew-2* do not report any frequent itemsets of size greater than two.

Table 1.3.5. First six Boolean expressions induced by top five frequent itemsets of size 2 in *BMS-Web-Wiew-2* at $\alpha = 0.009$

Frequent itemset	*supp* (E_{22}, D)	*supp* (E_{23}, D)	*supp* (E_{24}, D)	*supp* (E_{25}, D)	*supp* (E_{26}, D)
{1, 3}	0.0236	0.1695	0.1932	0.1710	0.1946
{1, 7}	0.0229	0.1702	0.1932	0.1741	0.1970
{3, 7}	0.0228	0.1719	0.1946	0.1743	0.1970
{3, 5}	0.0220	0.1726	0.1946	0.1572	0.1793
{1, 9}	0.0220	0.1712	0.1932	0.1512	0.1732

Table 1.3.6. First six Boolean expressions induced by top five frequent itemsets of size 2 in *T10I4D100K* at $\alpha = 0.01$

Frequent itemset	*supp* (E_{22}, D)	*supp* (E_{23}, D)	*supp* (E_{24}, D)	*supp* (E_{25}, D)	*supp* (E_{26}, D)
{217, 346}	0.0134	0.0405	0.0539	0.0214	0.0347
{789, 829}	0.0119	0.0335	0.0454	0.0690	0.0809
{368, 829}	0.0119	0.0665	0.0785	0.0690	0.0809
{368, 682}	0.0119	0.0665	0.0785	0.0319	0.0438
{39, 825}	0.0119	0.0307	0.0426	0.0237	0.0356

Table 1.3.6(continued). First six Boolean expressions induced by frequent itemsets of size 3 in *T10I4D100K* at $\alpha = 0.01$

Frequent itemset	*supp* (E_{32}, D)	*supp* (E_{33}, D)	*supp* (E_{34}, D)	*supp* (E_{35}, D)	*supp* (E_{36}, D)
{39, 704, 825}	0.0104	0.0004	0.0111	0.0015	0.0119

T10I4D100K reports only one frequent itemset of size 3. We observe that the proposed framework is simple and elegant. It enables us to synthesize arbitrary Boolean expressions induced by frequent itemsets in a database.

Also, we have conducted experiments to study the relationship between the size of a database and the execution time required for mining generators. The execution time required for mining generators in a database increases as the number of transactions contained in a database increases. We observe this phenomenon in Figures 1.3.2 and 1.3.3.

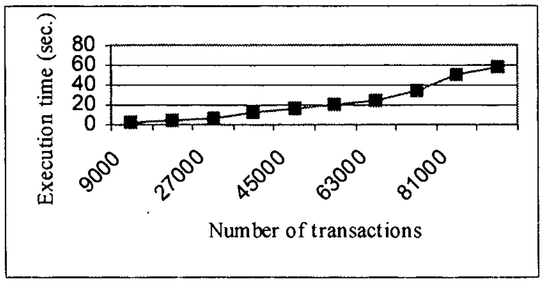


Figure 1.3.2. Execution time versus the number of transactions from *retail* at $\alpha = 0.05$

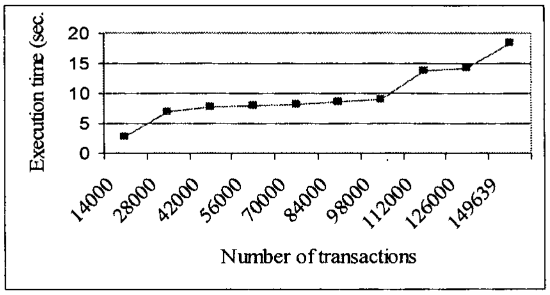


Figure 1.3.3. Execution time versus the number of transactions from *BMS-Web-Wiew-1* at $\alpha = 0.01$

We have also conducted experiments to find the execution time for synthesizing generators in a database. The time required (only) for synthesizing generators for each of the above databases is 0 millisecond at the respective value of α shown in each of the Tables 1.3.3, 1.3.4, 1.3.5, and 1.3.6.

Also, we have conducted experiments to study the relationship between the size of a database and the number of generators of Boolean expressions induced by frequent itemsets of size greater than or equal to 2. The experiments are conducted on databases *retail* and *BMS-Web-Wiew-1*. The results of the experiments are shown in Figures 1.3.3 and 1.3.4. From these figures, we could conclude that there is no universal relationship between the size of the database and the number of generators in it.

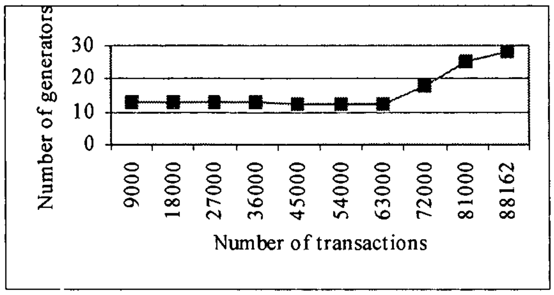


Figure 1.3.3. Number of generators versus the number of transactions from *retail* at $\alpha = 0.05$

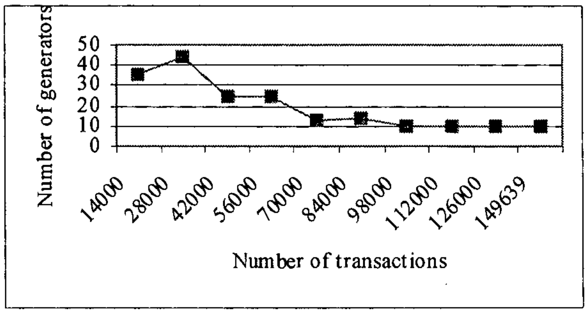


Figure 1.3.4. Number of generators versus the number of transactions from *BMS-Web-Wiew-1* at $\alpha = 0.01$

We have also conducted experiments for finding the number of generators corresponding to frequent itemsets of size greater than or equal to 2 in a database at a given α. The number of generators in a database decreases as α increases. We observe this phenomenon in Figures 1.3.5 and 1.3.6.

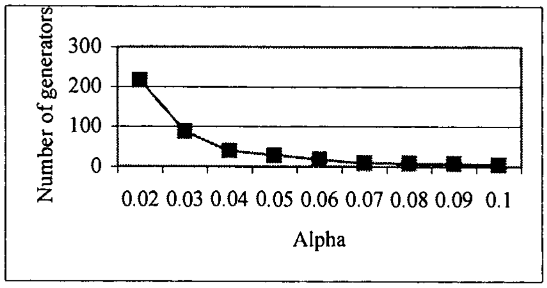


Figure 1.3.5. Number of generators versus α for *retail*

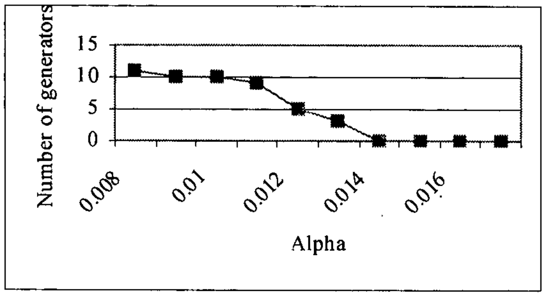


Figure 1.3.6. Number of generators versus α for *BMS-Web-Wiew-1*

1.3.4.1 Application: Effect of a specific item on other items over time

The proposed framework could be applied to study the effect of a specific item on other items over time in a market basket data. Let D_t be the database at time t, for $t = 1, 2, \ldots, n$. We wish to study how a specific item, say A, helps promoting other items over time. The proposed study is based on the frequent itemsets in these databases. Let the set of frequent itemsets related with A be $FIS_A \subseteq FIS(D_i)$, for $i = 1, 2, \ldots, n$. In particular, let $FIS_A = \{A, AB, AC, ABC, AD, AE\}$. We wish to study how the item A helps promoting items B, C, D, E, and BC over time. To study the effect of item A on item B, the Boolean expressions $A \wedge B$ and $\neg A \wedge B$ would provide some useful information. In other words, the expression $effect\ (A, B, D_i) = supp(A \wedge B, D_i)\ /\ supp(\neg A \wedge B, D_i)$ would provide some useful information to study the effect of item A on item B in D_i, for $i = 1, 2, \ldots, n$. Thus, to study induced by frequent itemsets. Thus, the generators enable us in synthesizing arbitrary Boolean expressions induced by the frequent itemsets. It is a simple and elegant technique. There is no need to introduce a specific framework for a specific type of Boolean expressions. The proposed framework is effective and promising.

Chapter 1.4

Capturing association among items in a database

The analysis of relationships among variables is a fundamental task being at the heart of many data mining problems. For instance, association rules [11] find relationships between sets of items in a database of transactions. Such rules express buying patterns of customers, e.g., finding how the presence of one item affects the presence of another and so forth.

Many measures of association have been reported in the literature of data mining, machine learning, and statistics. They could be categorized into two groups. Some measures deal with a set of objects, or could be generalized to deal with a set of objects. On the other hand, the remaining measures could not be generalized. Confidence [11], conviction [22] are examples of the second category of measures. On the other hand, measures such as Jaccard [75] could be generalized to find association among a set of items in a database. We shall see later why measures such as support [11], generalized Jaccard, and all-confidence [58] have not been effective in measuring association among a set of items in a database.

Various problems could be addressed using association among a set of items in market basket data. For example, a company might be interested in analyzing items that are purchased frequently. Let the items P, Q, and R be purchased frequently. A few specific problems are stated below involving these items.

(i) Some items (products) could be high profit making. Naturally, the company would like to promote them. There are various ways one could promote an item. An indirect way of promoting an item P is to promote items that are highly associated with it. The implication of high association between P and another item Q is that if Q is purchased by a customer then P is likely to be purchased by the same customer at the same time. Thus, P gets indirectly promoted.

(ii) Again, some items could be low-profit making. Thus, it is important to know how they promote sales of other items. Otherwise, the company could stop dealing with such items.

To solve the above problems, one could cluster the frequent items in a database. In the context of (i), one could promote item P indirectly, by promoting other items in the class containing P. In the context of (ii), the company could keep on dealing with R if the class size containing R is reasonably large. Thus, a suitable metric for capturing association among a set of items could enable us to cluster frequent items in a database. In general, many corporate decisions could be taken effectively by incorporating knowledge inherent in data. Later, we shall show that a measure of association based on a 2×2 contingency table might not be effective in clustering a set of items in a database. Thus, we propose measures of association for capturing association among a set of items in a database.

In this chapter, we present two measures of association among a set of items in a database. The second measure of association is based on a weighting model. We provide theoretical foundation of the work. For the purpose of measuring association among a set of items, we express second measure in terms of supports of itemsets. The main contributions of this chapter are given as follows: (1) We propose two measures of association among a set of items in a database, (2) We introduce the notion of associative itemset in a database, (3) We provide theoretical foundation of the work, and (4) We express second measure in terms of supports of itemsets.

In the following section, we study some existing measures and explain why these measures are not suitable for capturing association among a set of items in a database.

The rest of the chapter is organized as follows. We discuss related work in Section 1.4.2. In Section 1.4.3, we propose two new measures of association among a set of items in a database. We discuss various properties of proposed measures in Section 1.4.4. Also, we express second measure in terms of supports of itemsets. In Section 1.4.5, we mention an application of proposed measure of association. Experimental results are provided in Section 1.4.6 to show the effectiveness of the second measure of association.

1.4.2 Related work

Tan et al. [75] have described several key properties of twenty one interestingness measures proposed in statistics, machine learning and data mining literature. One needs to examine these properties in order to select right interestingness measure for a given application domain. Hershberger and Fisher [40] discuss some measures of association proposed in statistics. Most of the existing measures are based on a 2×2 contingency table. Thus, they might not be suitable for measuring association among a set of items.

Agrawal et al. [11] have proposed support measure in the context of finding association rules in a database. To find support of an itemset, it requires counting frequency of the itemset in the given database. An itemset in a transaction could be a source of association among items in the itemset. But, support of an itemset does not consider frequencies of it subsets. As a result, support of an itemset might not be a good measure of association among items in an itemset.

Piatetsky-Shapiro [63] has proposed leverage measure in the context of mining strong rules in a database. Adhikari and Rao [1] have proposed a measure called *OA*, to measure overall association between two items in a database. They might not be suitable for measuring of association among a set of items in a database.

Aggarwal and Yu [10] have proposed collective strength of an itemset. Collective strength is based of the concept of violation of an itemset. An itemset X is said to be in violation of a transaction, if some items of X are present in the transaction and others are not. Collective strength of an itemset X has been defined as follows.

$C(X) = \frac{1 - v(X)}{1 - E(v(X))} \times \frac{E(v(X))}{v(X)}$, where

v(*X*) is the violation rate of itemset *X*. It is the fraction of transactions in violation of itemset *X*. *E*(*v*(*X*)) is the expected violation rate of itemset *X*. The major concern regarding computation of *C*(*X*) is that the computation of *E*(*v*(*X*)) is based on statistical independence of items of *X*.

Cosine [38] and correlation [38] are used to measure association between two objects. They might not be suitable as a measure of association among items of an itemset.

Confidence and conviction are used to measure strength of association between itemsets in some sense. They might not be useful in the current context, since we are interested in capturing association among items of an itemset. In the following section, we introduce two measures to capture association among a set of items in a database.

1.4.3 New measures of association

Before we present our measures of association, we mention a few definitions and notations used frequently in this chapter.

A set of items in a database is called an *itemset*. Every itemset *X* in a database is associated with a statistical measure, called *support*. Support of an itemset *X* in database *D* is the fraction of transactions in *D* containing *X*, denoted by *S*(*X*, *D*). In general, let *S*(*E*, *D*) be the support of Boolean expression *E* defined on the transactions in database *D*. An itemset *X* is called *frequent* in *D* if $S(X, D) \geq \alpha$, where α is user-defined level of *minimum support*. If *X* is frequent then *Y* is also frequent, since $S(Y, D) \geq S(X, D)$, for $\phi \neq Y \subseteq X$. Each item in a frequent itemset is called a *frequent item*. Let $|X|$ denote the number of items of itemset *X*. Let *X* be $\{x_1, x_2, \ldots, x_m\}$. The following notations are used frequently in this chapter:

- $S_X(Y, D)$: support of Boolean expression defined on the transactions in *D* such that it contains all the items of *Y*, but not items of *X*-*Y*, for $\phi \neq Y \subseteq X$

- $S\left(\bigcup_{i=1}^{m}\{x_i\}, D\right)$: support of Boolean expression defined on the transactions in D such that it contains at least one item of X
- $S\left(\bigcap_{i=1}^{m}\{x_i\}, D\right)$: support of Boolean expression defined on the transactions in D such that it contains all the items of X

A measure of association gives numerical estimate of magnitude of statistical dependence among items in an itemset. Highly associated items are likely to be purchased together. In other words, items in X are highly associated, if one of the items of X is purchased then the remaining items of X are also likely to be purchased in the same transaction. One could define association among a set of items in many ways. Our first measure of association A_1 is defined as follows.

Definition 1.4.1. Let $X = \{x_1, x_2, \ldots, x_m\}$ be an itemset in database D. Let δ be the *minimum level of association.* The measure of association A_1 is defined as follows.

$$A_1(X, D) = \begin{cases} S(X, D)/S\left(\bigcup_{i=1}^{m}\{x_i\}, D\right), \text{for } |X| \geq 2 \\ \delta, \text{for } |X| = 1 \end{cases} \cdot \bullet \tag{1.4.1}$$

Measure A_1 is the proportion of the number of transactions containing all the items of X and the number of transactions containing at least one of the items of X. The association among items of an itemset and the number of association rules generated from the itemset are positively correlated, provided the support of the itemset is high. If the association among items of an itemset is more then it is expected to generate more association rules and vice versa. Palshikar et al. [59] have proposed heavy itemsets for mining association rules. An itemset X is *heavy* for given support and confidence values, if all possible association rules made up of items of X are present. Thus, items of heavy itemsets are expected to have high association among themselves. Measure A_1 could be considered as a generalized Jaccard measure for capturing association among items of an itemset.

A transaction in a database D provides the following information regarding association among items of X: (i) A transaction that contains all the items of X contributes maximum value towards overall association among items of X. We attach weight 1.0 to each such transaction. (ii) A transaction that contains k items of X contributes some value towards overall association among the items of X, for $2 \leq k \leq |X|$. We attach weight $k / |X|$ to each such transaction. (iii) A transaction that does not contain any item of X contributes no information regarding association among the items of X. (iv) A transaction that contains only one item of X contributes maximum value towards overall dispersion among the items of X. At a given X, we attach a weight to each transaction that contributes some value towards overall association among the items of X. Our second measure of association A_2 is defined as follows.

Definition 1.4.2. Let $X = \{x_1, x_2, \ldots, x_m\}$ be an itemset in database D. Let δ be the *minimum level of association.* Our second measure of association A_2 is defined as follows. $A_2(X, D) = \begin{cases} \sum_{Y \subseteq X, |Y| \geq 2} \left\{ \frac{S_X\langle Y, D\rangle}{S\left(\bigcup_{i=1}^{m} \{x_i\}, D\right)} \times \frac{|Y|}{|X|} \right\}, \text{for } |X| \geq 2 \\ \delta, \text{for } |X| = 1 \end{cases}$ • (1.4.2)

A_2 could be expressed as follows.

$$A_2(X, D) = \sum_{Y \subseteq X, |Y| \geq 2} \left\{ CS_X(Y, D) \times \frac{|Y|}{|X|} \right\}, \text{where } CS_X(Y, D) = \frac{S_X\langle Y, D\rangle}{S\left(\bigcup_{i=1}^{m} \{x_i\}, D\right)}. \tag{1.4.3}$$

1.4.4 Properties of A_1 and A_2

A_1 and A_2 could be used to measure association among a set of items in a database. Thus, each of these measures could be considered as a *generalized measure of association.* The following corollary is obtained from Definitions 1.4.1 and 1.4.2.

Corollary 1.4.1. *Let $X = \{x_1, x_2\}$ be an itemset in database D. Then, $A_1(X, D) = A_2(X, D)$ $= S(\{x_1\} \cap \{x_2\}, D) / S(\{x_1\} \cup \{x_2\}, D)$, for $|X| = 2$.* • (1.4.4)

The measure in (1.4.4) has been reported as a measure of similarity between two objects [83], [84]. To judge goodness of proposed measures, we state below *monotone* property of a measure of association.

Property 1.4.1. *Given an itemset X, if a subset Y occurs more frequently in the transactions containing at least two items of X then the items of X have stronger association, for $Y \subseteq X$, and $|Y| \geq 2$.* •

A_2 satisfies monotone property, for every $Y \subseteq X$, and $|Y| \geq 2$. But, A_1 satisfies monotone property, for $Y = X$, and $|Y| \geq 2$. Therefore, A_2 is more appealing measure of association than A_1. The following example verifies that A_2 measures association among items of an itemset more accurately than A_1.

Example 1.4.1: Consider the database $D = \{\{a, b, c, d\}, \{a, c\}, \{a, h\}, \{b, c\}, \{b, c, d\}, \{b, d, e\}, \{b, e\}, \{c, d, e\}\}$. Here, $A_1(\{b, c, d\}, D) = 0.2857$, and $A_2(\{b, c, d\}, D) = 0.5714$. In D, we observe that association among items of $\{b, c, d\}$ is closer to 0.5714 than 0.2857. •

In the context of monotone property of a measure of association, we discuss the effectiveness of all-confidence measure. For an itemset X, all-confidence measure has been defined as follows: all-confidence $(X) = S(X, D)$ / *maximum*$(S(\{x\}, D))$, for $x \in X$. Measure all-confidence satisfies monotone property, only for $Y = X$, where $Y \subseteq X$, and $|Y| \geq 2$. Thus, measure all-confidence might not be effective in capturing association among items in an itemset.

In Section 1.4.6, experimental results are provided using measure A_2. The following two lemmas are useful to show some interesting properties of A_2.

Lemma 1.4.1. *Let $X = \{x_1, x_2, \ldots, x_m\}$ be an itemset in database D. Also, let $T(i) = \sum_{Y \subseteq X, |Y| = m-i} S_X\langle Y, D\rangle$, for $i = 0, 1, \ldots, m-1$. Then $T(i)$ can be expressed as follows.*

$$\Big\{ \sum_{Y \subseteq X, |Y|=m-i} S(Y, D) - {}^{m-i+1}C_{m-i} \times \sum_{Y \subseteq X, |Y|=m-i+1} S(Y, D) + {}^{m-i+2}C_{m-i} \times \sum_{Y \subseteq X, |Y|=m-i+2} S(Y, D)$$
$$+ \ldots + (-1)^{i-1} \times {}^{m-1}C_{m-i} \times \sum_{Y \subseteq X, |Y|=m-1} S(Y, D) + (-1)^{i} \times {}^{m}C_{m-i} \times S(X, D) \Big\} \quad (1.4.5)$$

Proof. Let us consider the definition of $T(i)$, for $i = 0, 1, \ldots, m\text{-}1$. In particular, $T(k)$ is defined in terms of $S_X\langle Y, D\rangle$, for $|Y| = m\text{-}k$ such that $Y \subseteq X$. $S_X\langle Y, D\rangle$ could be expressed by itemsets of size greater than or equal to $m\text{-}k$. There are ${}^{m}C_{m-k}$ distinct Ys, and thus, we have ${}^{m}C_{m-k}$ distinct expressions for $S_X\langle Y, D\rangle$, one for each Y. Each expression of $S_X\langle Y, D\rangle$ contains a $S(X, D)$. Thus, the last term of $T(k)$ is ${}^{m}C_{m-k} \times S(X, D)$. The second last term contains itemsets of size $m\text{-}1$. Not all expressions of $S_X\langle Y, D\rangle$ contain a particular itemset Y_1 of size $m\text{-}1$. Y_1 is present in the expression of $S_X\langle Y, D\rangle$, if $Y_1 \subseteq Y$. The number of expressions of $S_X\langle Y, D\rangle$ that contain Y_1 is ${}^{m-1}C_{m-k}$. Other terms could be obtained in a similar way. •

We verify Lemma 1.4.1 with the help of following example. Let $X = \{x_1, x_2, x_3\}$. Then, $T(1) = S_X\langle\{x_1, x_2\}, D\rangle + S_X\langle\{x_1, x_3\}, D\rangle + S_X\langle\{x_2, x_3\}, D\rangle = S(\{x_1, x_2\}, D) - S(\{x_1, x_2, x_3\}, D) + S(\{x_1, x_3\}, D) - S(\{x_1, x_2, x_3\}, D) + S(\{x_2, x_3\}, D) - S(\{x_1, x_2, x_3\}, D) = \sum_{|Y|=3\text{-}1, Y\subseteq X} S(Y, D) + (\text{-}1)^1 \times {}^{3}\mathrm{C}_{3\text{-}1} \times S(X, D)$. Again, $T(2) = S_X\langle\{x_1\}, D\rangle + S_X\langle\{x_2\}, D\rangle + S_X\langle\{x_3\}, D\rangle = S(\{x_1\}, D) - S(\{x_1, x_2\}, D) - S(\{x_1, x_3\}, D) + S(\{x_1, x_2, x_3\}, D) + S(\{x_2\}, D) - S(\{x_1, x_2\}, D) - S(\{x_2, x_3\}, D) + S(\{x_1, x_2, x_3\}, D) + S(\{x_3\}, D) - S(\{x_1, x_3\}, D) - S(\{x_2, x_3\}, D) + S(\{x_1, x_2, x_3\}, D) = \sum_{|Y|=3\text{-}2, Y\subseteq X} S(Y, D) + (\text{-}1)^1 \times {}^{3\text{-}1}\mathrm{C}_{3\text{-}2} \times \sum_{|Y|=3\text{-}1, Y\subseteq X} S(Y, D) + (\text{-}1)^2 \times {}^{3}\mathrm{C}_{3\text{-}2} \times S(X, D)$. Thus, the expressions of $T(1)$ and $T(2)$ verify Lemma 1.4.1. We prove Lemma 1.4.2 based on Lemma 1.4.1.

Lemma 1.4.2. *Let* $X = \{x_1, x_2, \ldots, x_m\}$ *be an itemset in database* D. *Then*

$$\sum_{Y \subseteq X,\ 1 \le |Y| \le |X|} \left\{ S_X\langle Y, D\rangle \times \frac{|Y|}{|X|} \right\} = \frac{1}{m} \times \sum_{i=1}^{m} S(\{x_i\}, D) \tag{1.4.6}$$

Proof. $$\sum_{Y \subseteq X, |Y| = |X| \text{ down to } 1} \left\{ S_X\langle Y, D\rangle \times \frac{|Y|}{|X|} \right\} = \sum_{i=0}^{m-1} \left(\frac{m-i}{m} \right) \times T(i), \text{ where } i = m\text{-}|Y| \tag{1.4.7}$$

$$= S(X, D) + \left(\frac{m\text{-}1}{m}\right) \times \left(\sum_{Y \subseteq X, |Y| = m-1} S(Y, D) - {}^{m}C_{m-1} \times S(X, D) \right) + \ldots +$$

$$\left(\frac{m\text{-}2}{m}\right) \times \left(\sum_{Y \subseteq X, |Y| = m-2} S(Y, D) - {}^{m-1}C_{m-2} \times \left(\sum_{Y \subseteq X, |Y| = m-1} S(Y, D) \right) + {}^{m}C_{m-2} \times S(X, D) \right) -$$

$$\left(\frac{m-3}{m}\right)\times\left(\sum_{Y\subseteq X,\,|Y|=m-3} S(Y,D) - {}^{m-2}C_{m-3}\times\left(\sum_{Y\subseteq X,\,|Y|=m-2} S(Y,D)\right) + {}^{m-1}C_{m-3}\times\left(\sum_{Y\subseteq X,\,|Y|=m-1} S(Y,D)\right) - {}^{m}C_{m-3}\times S(X,D)\right)+$$

$$\left(\frac{m-(m-1)}{m}\right)\times\left(\sum_{Y\subseteq X,|Y|=1} S(Y,D) - {}^{2}C_{1}\times\left(\sum_{Y\subseteq X,|Y|=2} S(Y,D)\right) + \ldots \pm {}^{m}C_{1}\times S(X,D)\right), \text{[Lemma 1.4.1]} \tag{1.4.8}$$

$$= S(X,D)\times\{1-{}^{m-1}C_1+{}^{m-1}C_2-\ldots\pm{}^{m-1}C_{m-1}\}+\left(\frac{m-1}{m}\right)\times\left(\sum_{Y\subseteq X,|Y|=m-1} S(Y,D)\right)\times\{1-{}^{m-2}C_1+{}^{m-2}C_2+\ldots\pm{}^{m-2}C_{m-2}\}+$$

$$\left(\frac{m-2}{m}\right)\times\left(\sum_{Y\subseteq X,|Y|=m-2} S(Y,D)\right)\times\{1-{}^{m-3}C_1+{}^{m-3}C_2+\ldots\pm{}^{m-3}C_{m-3}\}+\ \ldots\ +$$

$$\left(\frac{2}{m}\right)\times\left(\sum_{Y\subseteq X,|Y|=2} S(Y,D)\right)\times\{1-{}^{1}C_1\}+\left(\frac{1}{m}\right)\times\left(\sum_{Y\subseteq X,|Y|=1} S(Y,D)\right) \tag{1.4.9}$$

$= \left(\frac{1}{m}\right)\times\left(\sum_{Y\subseteq X,|Y|=1} S(Y,D)\right)$, since the coefficient of $\left(\frac{p}{m}\right)\times\left(\sum_{Y\subseteq X,|Y|=p} S(Y,D)\right)$ is zero, for $2\le p\le m$. •

We verify Lemma 1.4.2 with the help of following example. Let $X = \{x_1, x_2, x_3\}$.

Then, $\sum_{Y\subseteq X,1\le|Y|\le|X|}\left\{S_X\langle Y,D\rangle\times\frac{|Y|}{|X|}\right\} = \frac{1}{3}\{S_X\langle\{x_1\},D\rangle+S_X\langle\{x_2\},D\rangle+S_X\langle\{x_3\},D\rangle\}$

$+\frac{2}{3}\{S_X\langle\{x_1,x_2\},D\rangle+S_X\langle\{x_1,x_3\},D\rangle+S_X\langle\{x_2,x_3\},D\rangle\}+S_X\langle\{x_1,x_2,x_3\},D\rangle$

$=\frac{1}{3}\{S_X(\{x_1\},D)-S_X(\{x_1,x_2\},D)-S_X(\{x_1,x_3\},D)+S_X(\{x_1,x_2,x_3\},D)+\ldots\}+$

$\frac{2}{3}\{S_X(\{x_1,x_2\},D)-S_X(\{x_1,x_2,x_3\},D)+\ldots\}+S_X(\{x_1,x_2,x_3\},D)$

$=\frac{1}{3}\{S_X(\{x_1\},D)+S_X(\{x_2\},D)+S_X(\{x_3\},D)\}$. We prove Lemma 1.4.3 based on Lemma 1.4.2.

Lemma 1.4.3. *Let $X = \{x_1, x_2, \ldots, x_m\}$ be an itemset in database D, for an integer $m\ge 2$.*

Then, $$A_2(X,D)=\frac{1}{m\times S(\bigcup_{i=1}^{m}\{x_i\},D)}\times\left[\sum_{i=1}^{m}\{S(\{x_i\},D)-S_X\langle\{x_i\}\rangle,D\}\right], x_i\in X \tag{1.4.10}$$

Proof. $$A_2(X,D)=\frac{1}{S(\bigcup_{i=1}^{m}\{x_i\},D)}\times\left[\sum_{Y\subseteq X,|Y|\ge1}^{m} S_X\langle Y,D\rangle\times\frac{|Y|}{|X|}-\frac{\sum_{i=1}^{m}S_X\langle\{x_i\},D\rangle}{m}\right] \tag{1.4.11}$$

$$= \frac{1}{S\left(\bigcup_{i=1}^{m}\{x_i\}, D\right)} \times \left[\frac{\sum_{i=1}^{m} S(\{x_i\}, D)}{m} - \frac{\sum_{i=1}^{m} S_X\langle\{x_i\}, D\rangle}{m} \right], \text{ [Lemma 1.4.2]. } \bullet \qquad (1.4.12)$$

Lemma 1.4.3 gives a simple expression for A_2. A few corollaries of Lemma 1.4.3 are given below.

Corollary 1.4.2. *Let* $X = \{x_1, x_2, ..., x_m\}$ *be an itemset in database D. If all the items in X have equal support then* $A_2(X, D) = q / S\left(\bigcup_{i=1}^{m}\{x_i\}, D\right)$,

where $q = S(\{x_i\}, D) - S_X\langle\{x_i\}, D\rangle$, for $i = 1, 2, ..., n$.• (1.4.13)

Corollary 1.4.3. *Let* $X = \{x_1, x_2, ..., x_m\}$ *be an itemset in database D. Then,* $A_2(X, D) = \sum_{i=1}^{m}[q_i/m]$, where $q_i = \left[S(\{x_i\}, D) - S_X\langle\{x_i\}, D\rangle\right] / S\left(\bigcup_{i=1}^{m}\{x_i\}, D\right)$. q_i / m *is the contribution of item* x_i *towards overall association among items in X, for i = 1, 2, ..., m.* •

(1.4.14)

Based on measure A_2, we define an *associative itemset* as follows.

Definition 1.4.3. Let $X = \{x_1, x_2, ..., x_m\}$ be an itemset in database D. Also, let δ be the *minimum level of association.* X is associative at level δ if $A_2(X, D) \geq \delta$. •

From definition of A_2, an itemset of size 1 is associative at level δ. If $\delta > \alpha$ then a frequent itemset X might not be associative at level δ. This is because of the fact that the association among items of X might lie in $[\alpha, \delta)$. In many applications, we are interested in the itemsets that are frequent as well as associative. In the next section, we mention one such application. In Examples 1.4.2 and 1.4.3, we illustrate the difference between associative itemsets and frequent itemsets.

Example 1.4.2. We consider the following three transactional databases. Let $D_1 = \{\{a, b, c, d\}, \{a, c\}, \{a, h\}, \{b, c\}, \{b, c, d\}, \{b, d, e\}, \{b, e\}, \{c, d, e\}\}$, $D_2 = \{\{a, b, c, e\}, \{a, b, f\}, \{a, d\}, \{a, g, h\}, \{b, d, g\}, \{b, f, g\}, \{b, g, i\}, \{b, j\}, \{c, d\}\}$, and $D_3 = \{\{a, b, c, e\}, \{a, b, g, i\}, \{a, b, j\}, \{a, c, d, e\}, \{b, d, g\}, \{c, d\}, \{f, g\}, \{g, h\}\}$. Let $\alpha = 0.2$, and $\delta = 0.4$. In database D_1, $S(\{a, b\}, D_1) = 1/8$, and $A_1(\{a, b\}, D_1) = A_2(\{a, b\}, D_1) = 1/7$. Thus, the itemset $\{a, b\}$ is not frequent and also not associative. In database D_2, $S(\{a, b\}, D_2) = 2/9$, and $A_1(\{a, b\}, D_2) = A_2(\{a, b\}, D_2) = 1/4$. Thus, the itemset $\{a, b\}$ is frequent but not associative. In database D_3, $S(\{a, b\}, D_3) = 3/8$, and $A_1(\{a, b\}, D_3) = A_2(\{a, b\}, D_3) = 3/5$. Thus, the itemset $\{a, b\}$ is frequent as well as associative. •

An associative itemset not necessarily be frequent at level α. This is because of the fact that the subsets of the itemset might be available frequently in different transactions.

Example 1.4.3. Consider the database D_1 of Example 1.4.1. Let $\alpha = 0.2$, and $\delta = 0.4$. $S(\{c, d, e\}, D_1) = 0.125$, and $A_2(\{c, d, e\}, D_1) = 0.429$. Thus, the itemset $\{c, d, e\}$ is associative, but not frequent. •

Example 1.4.3 is another instance that shows that the support of an itemset could not effectively measure association among items of an itemset. In the following lemma, we shall prove that association among items of a frequent itemset is always greater than or equal to α under A_2. Thus, a frequent itemset is associative at level α.

Lemma 1.4.4. *Let $X = \{x_1, x_2, ..., x_m\}$ be a frequent itemset in database D, for an integer $m \geq 2$. Then, $A_2(X, D) \geq \alpha$.*

Proof. $A_2(X, D) = \sum_{Y \subseteq X, |Y| \geq 2} \left\{ \frac{S_X\langle Y, D\rangle}{S\left(\bigcup_{i=1}^{m} \{x_i\}, D\right)} \times \frac{|Y|}{|X|} \right\}$

$$= \frac{S(X, D)}{S\left(\bigcup_{i=1}^{m} \{x_i\}, D\right)} + \sum_{Y \subset X, |Y| \geq 2} \frac{S_X\langle Y, D\rangle}{S\left(\bigcup_{i=1}^{m} \{x_i\}, D\right)} \times \frac{|Y|}{|X|} \quad (1.4.15)$$

$S(X, D) \geq \alpha$ implies $S(X, D)/S\left(\bigcup_{i=1}^{m} \{x_i\}, D\right) \geq \alpha$, since $0 < S\left(\bigcup_{i=1}^{m} \{x_i\}, D\right) \leq 1$.

$$\text{Thus, } A_2(X, D) \geq \alpha + \sum_{Y \subset X, |Y| \geq 2} \left\{ \frac{S_X\langle Y, D\rangle}{S\left(\bigcup_{i=1}^{m} \{x_i\}, D\right)} \times \frac{|Y|}{|X|} \right\}. \bullet \quad (1.4.16)$$

In the following lemma, we discuss an important property of A_2. Association among items of an itemset under A_2 lies in (0, 1].

Lemma 1.4.5. *Let $X = \{x_1, x_2, ..., x_m\}$ be a frequent itemset in database D, for an integer $m \geq 1$. Then, $0 < A_2(X, D) \leq 1$.*

Proof. From Definition 1.4.2, we get $0 < A_2(X, D) \leq 1$, for $m = 1$. Thus, we need to prove the result for $m \geq 2$. Also from Definition 1.4.2, we get $A_2(X, D) > 0$, for $m \geq 2$. We shall use the method of induction on $|X|$ to show that $A_2(X, D) \leq 1$, for $m \geq 2$. For $m = 2$, $X = \{x_1, x_2\}$. Then,

$$A_2(X, D) = \frac{S(\{x_1\}, D) - S_X\langle\{x_1\}, D\rangle + S(\{x_2\}, D) - S_X\langle\{x_2\}, D\rangle}{2\times\{S(\{x_1\}, D) + S(\{x_2\}, D) - S(\{x_1\}\cap\{x_2\}, D)\}}, \text{ [Lemma1.4.3]} \quad (1.4.17)$$

Also, $S_X\langle\{x_i\}, D\rangle = S(\{x_i\}, D) - S(\{x_1\}\cap\{x_2\}, D)$, for $i = 1, 2$.

$$\text{Thus, } A_2(X, D) = \frac{S(\{x_1, x_2\}, D)}{S(\{x_1\}, D) + S(\{x_2\}, D) - S(\{x_1\}\cap\{x_2\}, D)} \quad (1.4.18)$$

We have, $S(\{x_1\}\cap\{x_2\}, D) \leq S(\{x_1\}, D)$ and $S(\{x_2\}, D) - S(\{x_1\}\cap\{x_2\}, D) \geq 0$. So, $A_2(X, D) \leq 1$. Thus, the result is true for $m = 2$. We assume that the result is true for $m \leq k$-1. Now, we shall prove that it is true for $m = k$.

$$A_2(X, D) = \frac{1}{k\times S(\cup_{i=1}^{k}\{x_i\}, D)}\times\left[\sum_{i=1}^{k}\{S(\{x_i\}, D) - S_X\langle\{x_i\}, D\rangle\}\right] \quad (1.4.19)$$

$$= \frac{1}{k\times S(\cup_{i=1}^{k}\{x_i\}, D)}\times\left[\sum_{i=1}^{k-1}\{S(\{x_i\}, D) - S_X\langle\{x_i\}, D\rangle\}\right] + \frac{1}{k\times S(\cup_{i=1}^{k}\{x_i\}, D)}\times\{S(\{x_k\}, D) - S_X\langle\{x_k\}, D\rangle\}$$

(1.4.20)

$$= \frac{1}{k\times S(\cup_{i=1}^{k}\{x_i\}, D)}\times\left[c_1\times(k-1)\times S(\cup_{i=1}^{k-1}\{x_i\}, D) + c_2\times S(\{x_k\}, D)\right],$$ $0 \leq c_1, c_2 \leq 1$, [induction hypothesis] (1.4.21)

$$\leq \frac{c_1\times(k-1)}{k} + \frac{c_2}{k} \leq \frac{(k-1)}{k} + \frac{1}{k} = 1. \bullet \quad (1.4.22)$$

Let $X = \{x_1, x_2, \ldots, x_m\}$ be a frequent itemset in D. Let $Y \subset X$ such that $|Y| = |X|$-1, and $|X| \geq 3$. Let $Y = \{x_1, x_2, \ldots, x_{m-1}\}$. We try to establish the relationship between $A_2(X, D)$ and $A_2(Y, D)$. Using formula (1.4.20) we get,

$$A_2(X, D)\times m\times S(\cup_{i=1}^{m}\{x_i\}, D) = A_2(Y, D)\times(m-1)\times S(\cup_{i=1}^{m-1}\{x_i\}, D) + S(\{x_m\}, D) - S_X\langle\{x_m\}, D\rangle \quad (1.4.23)$$

or, $A_2(X, D) = A_2(Y, D) \times K_1 + K_2$, where

$$K_1 = \frac{(m-1)\times S\left(\bigcup_{i=1}^{m-1}\{x_i\}, D\right)}{m\times S\left(\bigcup_{i=1}^{m}\{x_i\}, D\right)}, \text{ and } K_2 = \frac{S(\{x_m\}, D) - S_X\langle\{x_m\}, D\rangle}{m\times S\left(\bigcup_{i=1}^{m}\{x_i\}, D\right)}. \quad (1.4.24)$$

We note that, $0 < K_1, K_2 \leq 1$. There does not exist any fixed relationship between $A_2(X, D)$ and $A_2(Y, D)$, for all $Y \subset X$ such that $|Y| = |X|-1$, and $|X| \geq 3$. We consider the following example.

Example 1.4.4. Consider the database $D_4 = \{\{a, b, c, d\}, \{a, b, d\}, \{a, b, e\}, \{a, b, f\}, \{a, f\}, \{a, g\}, \{d, i\}, \{i, j\}\}$. $A_2(\{a, b\}, D_4) = 0.66667$, $A_2(\{a, c\}, D_4) = 0.16667$, $A_2(\{a, b, c\}, D_4) = 0.5$. We observe that, $A_2(\{a, b, c\}, D_4) < A_2(\{a, b\}, D_4)$, and $A_2(\{a, b, c\}, D_4) > A_2(\{a, c\}, D_4)$. •

We wish to express A_2 in terms of supports of itemsets. Given an itemset X, the following lemma expresses $S_X\langle\{x_i\}, D\rangle$ in terms of the supports of itemsets $Y \subseteq X$, for $x_i \in X$.

Lemma 1.4.6. *Let $X = \{x_1, x_2, ..., x_m\}$ be an itemset in database D, for an integer $m \geq 1$.* Then, $S_X\langle\{x_i\}, D\rangle = S(\{x_i\}, D) - \sum_{j=1;\, j\neq i}^{m} S(\{x_i\}\cap\{x_j\}, D) + \sum_{j,k=1;\, j<k;\, j,k\neq i}^{m} S(\{x_i\}\cap\{x_j\}\cap\{x_k\}, D) - \ldots +$ $(-1)^{m-1}\times S\left(\bigcap_{i=1}^{m}\{x_i\}, D\right)$, *for i = 1, 2, ..., m.* (1.4.25)

Proof. We shall prove the result using the method of induction on m. The result trivially follows for $m = 1$. For $m = 2$, $X = \{x_1, x_2\}$. Then, $S_X\langle\{x_i\}, D\rangle = S(\{x_i\}, D) - S(\{x_1\}\cap\{x_2\}, D)$, for $i = 1, 2$. Hence, the result is true for $m = 2$. Assume that the result is true for $m = p$. We shall prove that the result is true for $m = p + 1$. Let $X = \{x_1, x_2, ..., x_{p+1}\}$. Due to the addition of x_{p+1}, the following observations are made. $S(\{x_i\}\cap\{x_{p+1}\}, D)$ is required to be subtracted, for $1 \leq i \leq p$. $S(\{x_i\}\cap\{x_j\}\cap\{x_{p+1}\}), D)$ is required to be added, for $1 \leq i < j \leq p$. Finally, the term $(-1)^p \times S(\{x_1\}\cap\{x_2\}\cap\ldots\cap\{x_{p+1}\}, D)$ is required to be added. Thus, $S_X\langle\{x_i\}, D\rangle = S(\{x_i\}, D) - \sum_{j=1,\, j\neq i}^{p+1} S(\{x_i\}\cap\{x_j\}, D) + \sum_{j,k=1;\, j<k;\, j,k\neq i}^{p+1} S(\{x_i\}\cap\{x_j\}\cap\{x_k\}, D) + \ldots + (-1)^p$ $\times S(\{x_1\}\cap\{x_2\}\cap\ldots\cap\{x_{p+1}\}, D)$. (1.4.26)

Thus, the induction step follows. Hence, the result follows. •

The following lemma expresses association among items of X in terms of supports of subsets of X.

Lemma 1.4.7. *Let $X = \{x_1, x_2, \ldots, x_m\}$ be an itemset in database D, for $m \geq 2$. Then*

$$A_2(X,D) = \frac{\sum_{i=1}^{m}\left[\sum_{j=1;\, j\neq i}^{m} S(\{x_i\}\cap\{x_j\}, D) - \sum_{j,k=1;\, j,k\neq i}^{m} S(\{x_i\}\cap\{x_j\}\cap\{x_k\}, D) + \ldots \pm S(\{x_1\}\cap\ldots\cap\{x_m\}, D)\right]}{m\times\left[\sum_{i=1}^{m} S(\{x_i\}, D) - \sum_{i,j=1;\, i<j}^{m} S(\{x_i\}\cap\{x_j\}, D) + \ldots \pm S(\{x_1\}\cap\{x_2\}\cap\ldots\cap\{x_m\}, D)\right]} \quad (1.4.27)$$

Proof. We state the theorem of total probability [60] as follows. For any m events x_1, x_2, …, x_m, we have $S\left(\bigcup_{i=1}^{m}\{x_i\}, D\right) = \sum_{i=1}^{m} S(\{x_i\}, D) - \sum_{1\leq i<j\leq m} S(\{x_i\}\cap\{x_j\}, D) + \ldots \pm S(\{x_1\}\cap\{x_2\}\cap\ldots\cap\{x_m\}, D)$

(1.4.28)

Result follows using Lemmas 1.4.3 and 1.4.6. •

A few corollaries of Lemma 1.4.7 are presented below.

Corollary 1.4.4. $A_2(X, D) = \dfrac{S(\{x_1\}\cap\{x_2\}, D)}{S(\{x_1\}, D) + S(\{x_2\}, D) - S(\{x_1\}\cap\{x_2\}, D)}$, *for $m = 2$.* • (1.4.29)

Corollary 1.4.5. *For $m = 3$, $A_2(X, D) = E_1 / E_2$, where*

$E_1 = 2\times\{S(\{x_1\}\cap\{x_2\}, D) + S(\{x_1\}\cap\{x_3\}, D) + S(\{x_2\}\cap\{x_3\}, D)\} - 3\times S(\{x_1\}\cap\{x_2\}\cap\{x_3\}, D)$, *and*

$E_2 = 3\times\{S(\{x_1\}, D) + S(\{x_2\}, D) + S(\{x_3\}, D) - S(\{x_1\}\cap\{x_2\}, D) - S(\{x_1\}\cap\{x_3\}, D)\}$
$3\times\{- S(\{x_2\}\cap\{x_3\}, D) + S(\{x_1\}\cap\{x_2\}\cap\{x_3\}, D)\}$. • (1.4.30)

Thus, $A_2(X, D)$ could be computed when supports of subsets of X are available. In Lemma 1.4.8, we study some properties of measure A_1.

Lemma 1.4.8. *Let X be an itemset in database D. Then the measure of association A_1 satisfies the following properties.* (i) *$0 < A_1(X, D) \leq 1$,* (ii) *$A_1(X, D) \leq A_1(Y, D)$, for $Y \subseteq X$, and $|Y| \geq 2$, and* (iii) *$A_1(X, D) \geq \alpha$, if X is a frequent itemset.*

Proof. (i) $A_1(X, D) > 0$, since $S(X, D) > 0$. Also, $A_1(X, D) \leq 1$, since $S\left(\bigcup_{y\in X}\{y\}, D\right) \geq S(X, D)$.

(ii) Let $X = \{x_1, x_2, \ldots, x_m\}$. We consider an itemset $Y \subseteq X$, such that $|Y| \geq 2$. Then, $A_1(X, D) \leq A_1(Y, D)$, since $S(X, D) \leq S(Y, D)$ and $S\left(\bigcup_{z\in X}\{z\}, D\right) \geq S\left(\bigcup_{z\in Y}\{z\}, D\right)$.

(iii) $A_1(X, D) = S(X, D)/S(\cup_{x_i \in X}\{x_i\}, D) \geq S(X, D) \geq \alpha$, since $0 < S(\cup_{y \in X}\{y\}, D) \leq 1$. •

The proposed measures of association could be used in many applications. In the following section, we mention an application of proposed measures of association.

1.4.4.1 Capturing association

Let $X = \{x_1, x_2, ..., x_k\}$ be an itemset in the given database D. In finding association among items in X, the following procedure could be followed [6], [83]. The algorithm finds association between every pair of items. The items in X form a class, if ${}^kC_2$ association values corresponding to ${}^kC_2$ pairs of items are close. The level of association among the items in this class is assumed as the minimum of ${}^kC_2$ association values. If the number of items in a class is more than two, then we observe that this technique fails to estimate correctly the association among the items in a group. Then the accuracy of association among items in X becomes low. Instead of that, one could estimate the association among items in X using measure A_2. The difference in similarity using measure A_2 is given by $DS(X, D) = A_2(\{x_1, x_2, ..., x_k\}, D)$ - *minimum*$\{EMS(x_i, x_j, D) : 1 \leq i < j \leq k\}$, where *EMS* is an existing measure of similarity that measures similarity between two items in D. Due to the *monotone* property of A_2, the amount of difference could be significant (as reported in Tables 1.4.4 and 1.4.5). In particular, $DS(X, D)$ becomes 0, if $k = 2$ [Corollary 1.4.1].

1.4.5 An application: Clustering frequent items in a database

We have observed that the measure A_2 is effective in finding association among items of an itemset in a database. For the purpose of clustering frequent items in a database, we could find associations among items of each frequent itemset of size greater than 1. Items of a highly associative itemset could be put in the same class. Thus, one could cluster the frequent items in a given database using A_2.

Adhikari and Rao [6] have proposed a technique for clustering multiple databases. If a cluster contains a class of size greater than 2, we have observed that the proposed clustering technique might cluster frequent items with higher degree of accuracy, since A_2 possesses monotone property. In the context of clustering data, an overview of different clustering techniques is given by Jain et al. [44].

There are two approaches of measuring association among items of each itemset in a database. In the first approach, one could synthesize association among items of the current frequent itemset. As soon as a frequent itemset is found during the mining process, one could call an algorithm of finding association among items of the current frequent itemset. When a frequent itemset is extracted, then all the non-null subsets of the frequent itemest might have been available [13]. Thus, one could synthesize association among items of the current frequent itemset. Also, one could synthesize association among items of each frequent itemsets after the mining task. In the second approach, all the frequent itemsets could be processed at the end of mining task.

1.4.6 Experimental results

We have carried out several experiments to find association among items of each frequent itemset in a database. All the experiments have been implemented on a 2.8 GHz Pentium D dual processor with 512 MB of memory using visual C++ (version 6.0) software. We present the experimental results using four databases. The databases *retail* [34], *mushroom* [34] are real. The database *ecoli* is a subset of *ecoli database* [77] and has been processed for the purpose of conducting experiment. Also, we have omitted non-numeric fields of *ecoli database* for the purpose of conducting experiments. The fourth database *check* is artificial. The database *check* contains the following transactions: {34, 47, 62}, {34, 55, 62, 102}, {47, 62}, {47, 62, 90}, {55, 102}. We have introduced database *check* for the purpose of verifying results. We present some characteristics of these databases in Table 1.4.1.

Table 1.4.1. Database characteristics

Database	# transactions	Avg length of a transaction	Avg frequency of an item	# items
retail	88,162	11.30576	80.45880	10,000
mushroom	8,124	23.00000	1570.18487	119
ecoli	336	7.000000	25.565217	92
check	5	2.80000	2.33333	6

There are many interesting algorithms [13], [39], [66] reported to mine frequent itemsets in a database. We have implemented apriori algorithm [13] for mining frequent itemsets in a database, since it is simple and easy to implement.

1.4.6.1 Top associative itemsets among frequent itemsets

We have conducted experiments on different databases to mine itemsets that are frequent as well as associative. Databases *retail*, *mushroom*, *ecoli* and *check* are mined at α 0.03, 0.1, 0.5, and 0.1, respectively. Top 10 associative itemsets among frequent itemsets in these databases are given in Tables 1.4.2, and 1.4.3. Some itemsets in mushroom database are highly frequent. Thus, associations among items in these itemsets are significantly high.

Table 1.4.2. Top 10 associative itemsets in databases *retail* and *mushroom*

retail		*mushroom*	
Itemset	A_2(Itemset)	Itemset	A_2(Itemset)
{39,41,48}	0.39183	{85,86}	0.97538
{38,39,48}	0.38044	{34,85,86}	0.97530
{32,39,48}	0.36929	{34,85}	0.97415
{38,39,41}	0.23977	{85,86,90}	0.96570
{39,41}	0.21057	{34,85,90}	0.96455
{38,170}	0.19350	{34,86,90}	0.94847
{41,48}	0.18763	{36,85,86}	0.93763
{38,39}	0.18498	{34,36,85}	0.93755
{36,38}	0.17723	{85,90}	0.92171
{38,110}	0.17396	{34,36,86}	0.92114

Table 1.4.3. Top 10 associative itemsets in databases *ecoli* and *check*

ecoli		*check*	
Itemset	A_2(Itemset)	Itemset	A_2(Itemset)
{48,50}	0.94152	{47,62}	0.75000
{44,48,50}	0.68469	{47,62,90}	0.58333
{37,48,50}	0.68264	{34,55,102}	0.55556
{40,48,50}	0.68051	{34,47,62}	0.50000
{48,50,54}	0.67758	{34,62}	0.50000
{42,48,50}	0.67650	{55,62,102}	0.33333
{48,50,51}	0.66764	{34,62,102}	0.33333
{48,49,50}	0.65511	{34,55,62}	0.33333
{47,48,50}	0.65024	{47,90}	0.33333
{44,48}	0.14873	{34,102}	0.33333

1.4.6.2 Finding the difference in similarity

We have conducted experiments on different databases to mine itemsets that are frequent as well as associative. For the purpose of comparison, we take similarity measure $sim_I(x_i, x_j, D) = S(\{x_1\} \cap \{x_2\}, D)/S(\{x_1\} \cup \{x_2\}, D)$ [83]. We present top 10 associative itemsets along with their difference in similarities in Tables 1.4.4, and 1.4.5.

Table 1.4.4. Top 10 associative itemsets along with their difference in similarities for *retail*, and *mushroom*

retail		*mushroom*	
Itemset	*DS*(Itemset, *retail*)	Itemset	*DS*(Itemset, *mushroom*)
{39,41,48}	0.20419	{85,86}	0
{38,39,48}	0.22089	{34,85,86}	0.00115
{32,39,48}	0.22196	{34,85}	0
{38,39,41}	0.09351	{85,86,90}	0.08861
{39,41}	0	{34,85,90}	0.06448
{38,170}	0	{34,86,90}	0.07138
{41,48}	0	{36,85,86}	0.12196
{38,39}	0	{34,36,85}	0.12490
{36,38}	0	{85,90}	0
{38,110}	0	{34,36,86}	0.10849

Some itemsets in the mushroom database are highly frequent. Thus, associations among items in these itemsets are significantly high.

Table 1.4.5. Top 10 associative itemsets along with their difference in similarities for *ecoli*, and *check*

ecoli		*check*	
Itemset	*DS*(Itemset, *ecoli*)	Itemset	*DS*(Itemset, *check*)
{48,50}	0	{34,47,62}	0.50000
{44,48,50}	0.79279	{47,62}	0
{37,48,50}	0.54808	{47,62,90}	0.25000
{40,48,50}	0.54119	{34,55,102}	0.22222
{48,50,54}	0.55296	{34,62}	0
{42,48,50}	0.53718	{55,62,102}	0.13333
{48,50,51}	0.52654	{34,62,102}	0.13333
{48,49,50}	0.54366	{34,55,62}	0.13333
{47,48,50}	0.54048	{47,90}	0
{44,48}	0	{34,102}	0

1.4.6.3 Execution time for measuring association

We have studied execution time for measuring association among items in each frequent itemset of size greater than one. In the first case, the execution time is studied with respect to the number of transactions in a database. As the number of transactions in a database increases, the number of frequent itemsets is likely to increase. Therefore, execution time increases as the size of a database increases. We observe the phenomenon in Figures 1.4.1, 1.4.2 and 1.4.3.

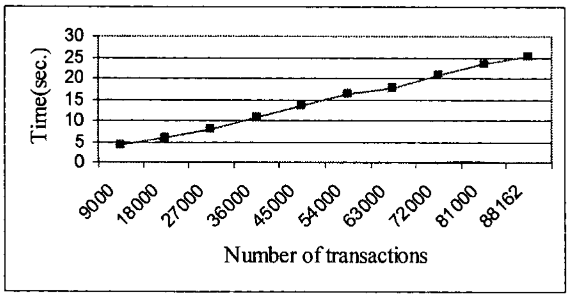


Figure 1.4.1. Execution time versus database size at $\alpha = 0.03$ (*retail*)

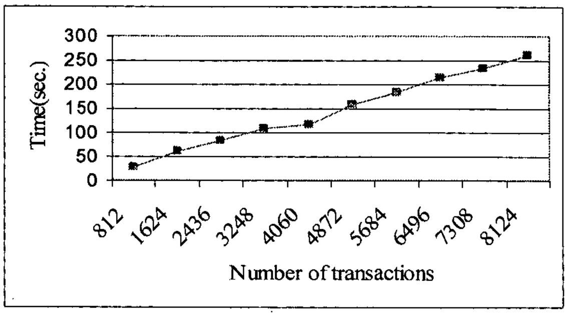


Figure 1.4.2. Execution time versus database size at $\alpha = 0.1$ (*mushroom*)

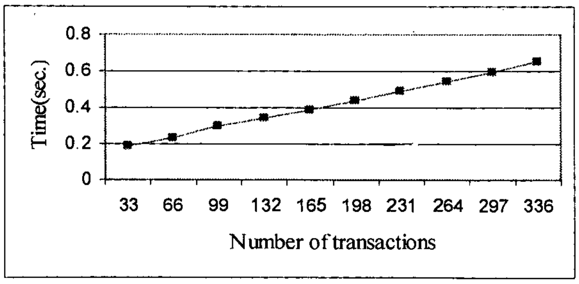


Figure 1.4.3. Execution time versus database size at $\alpha = 0.1$ (*ecoli*)

In the second case, the execution time is studied with respect to α. As we increase α, the number of frequent itemsets is likely to decrease. Therefore, the execution time decreases as α increases. We observe the phenomenon in Figures 1.4.4, 1.4.5, and 1.4.6.

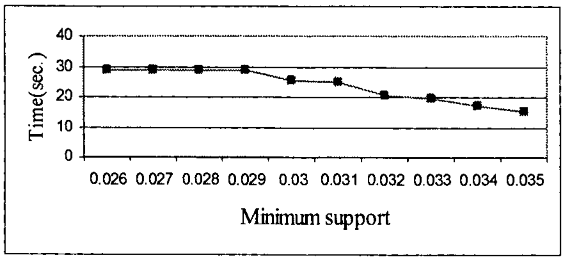


Figure 1.4.4. Execution time versus α (*retail*)

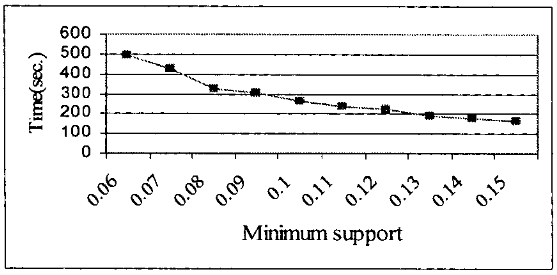


Figure 1.4.5. Execution time versus α (*mushroom*)

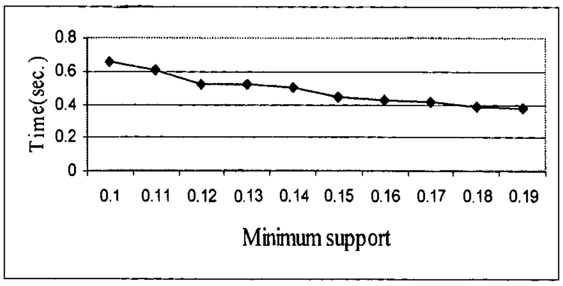


Figure 1.4.6. Execution time versus α (*ecoli*)

1.4.7 Conclusion

In this chapter, we present two measures of association among items in an itemset in a database. An existing measure might not be effective in capturing association among items in an itemset of size greater than 2. Many research problems could boil down to capturing association among items in an itemset. We have given an example of one such application in Section 1.4.5.

We have introduced the notion of associative itemset in a database. We have provided many useful lemmas and examples to make foundation of proposed measures strong and clear. Using monotone property of a measure of association, we have shown that A_2 measures association among items in an itemset more accurately than A_1.

For the purpose of computing A_2, we express it in terms of supports of itemsets. The measure of association A_2 is effective in capturing statistical association among items in a database.

Chapter 1.5

Association rules induced by item and quantity purchased

Pattern recognition [11], [81] and interestingness measures [41], [75] are two important as well as interesting topics being at the heart of many data mining problems. Association analysis using association rules [11], [17] has been studied well on binary data. A pattern is normally associated with some interestingness measures. Thus, a pattern would become interesting if the values of associated interestingness measures satisfy some conditions. Positive association rules in a database are expressed in the form of a forward implication, $X \rightarrow Y$, between two itemsets X and Y in the database such that $X \cap Y = \phi$. The meaning attached to association rule $X \rightarrow Y$ is that if all the items in X are purchased by a customer then it is likely that all the items in Y are purchased by the same customer at the same time. On the other hand, negative association rules are expressed by one of the following three forward implications: $X \rightarrow \neg Y$, $\neg X \rightarrow Y$, and $\neg X \rightarrow \neg Y$, for itemsets X and Y in the database such that $X \cap Y = \phi$. Let us consider the negative association rules of the form $X \rightarrow \neg Y$. The meaning attached with this implication is that if all the items in X are purchased by a customer then it is unlikely that all the items in Y are purchased by the same customer at the same time. Most of the real life transactional data are non-binary, in sense that an item could be purchased multiple times in a transaction. Thus, it is necessary to study the applicability of traditional support-confidence framework for mining association rules in these databases.

Association rule mining in a binary database is based on support (supp)-confidence (conf) framework established by Agrawal et al. [11]. Let $I(BD)$ be the set of items in binary database BD. A positive association rule in BD expresses positive association between itemsets X and Y, called the antecedent and consequent of the association rule, respectively. Each itemset in BD is associated with a statistical measure, called the support of the itemset. *Support* of itemset X in BD is the fraction of transactions in BD containing X, denoted by $supp(X, BD)$. The interestingness of an association rule is expressed by its support and confidence measures. The support and confidence of an association rule r: $X \rightarrow Y$ in a binary database BD are defined as follows: $supp(r, BD) = supp(X \cap Y, BD)$, and $conf(r, BD) = supp(X \cap Y, BD) / supp(X, BD)$. In other words, the support of association rule r in BD is the fraction of transactions in BD containing both X and Y, and the confidence of association rule r in BD is the fraction of transactions in BD containing Y among the transactions containing X. An association rule r in BD is *interesting* if $supp(r, BD) \geq$ *minimum support*, and $conf(r, BD) \geq$ *minimum confidence*. The parameters, minimum support and minimum confidence, are user inputs given to an association rule mining algorithm.

Though association rule mining in a binary database has been studied well, it has got limited usage, since in a real life transaction items are often purchased multiple times in the same transaction. Let TIMT be the type of a database such that a transaction in the database might contain an item multiple times. In this chapter, a database refers to a TIMT type database, if the type of the database is unspecified. Then, the question comes to our mind whether the traditional support-confidence framework still works for mining association rules in a TIMT type database. Before answering to this question, first we take an example of a TIMT type database DB as follows.

Example 1.5.1. Let $DB = \{\{A(300), B(500), C(1)\}, \{A(2), B(3), E(2)\}, \{A(3), B(2), E(1)\}, \{A(2), E(1)\}, \{B(3), C(2)\}\}$, where $x(\eta)$ denotes item x purchased η numbers at a time in the corresponding transaction. The number of occurrences of itemset $\{A, B\}$ in the first transaction is equal to *minimum* $\{300, 500\}$, i.e., 300. Thus, the total number of occurrences of $\{A, B\}$ in DB is 304. Also, $\{A, B\}$ has occurred in 3 out of 5 transactions in DB. Thus, the following attributes of itemset X are important consideration for making association analysis of items in X: number of occurrences of X in DB, and number of transactions in DB containing X. •

In Section 1.5.2, we have explained why the traditional support-confidence is not adequate for mining association rules in a TIMT type database.

Rest of the chapter is organized as follows. In Section 1.5.2, we study association rules in a TIMT type database and introduce three categories of association rules. In Section 1.5.3, we introduce a framework based on traditional support-confidence framework for mining each category of association rules. We study the properties of proposed interestingness measures in Section 1.5.4. In Section 1.5.5, we discuss a method for mining association rules in a TIMT type database. Experimental results are provided in Section 1.5.6. We discuss related work in Section 1.5.7.

1.5.2 Association rules in a TIMT type database

We are given a TIMT type database DB. A transaction in DB containing p items could be stored as follows: $\{i_1(n_1), i_2(n_2), \ldots, i_p(n_p)\}$, where item i_k is purchased n_k (≥ 1) numbers at a time in the transaction, for $i = 1, 2, \ldots, p$. Also, a transaction is stored along with other attributes: transaction id, date of purchase, and so forth. These attributes do not have impact on association rules in a database and thus, we shall not deal with them.

Each itemset X in a transaction is associated with the following two attributes: transaction-itemset frequency (TIF), and transaction-itemset status (TIS). These two attributes are defined as follows:

$TIF(X, \tau, DB) = m$, if X occurs m times in transaction τ in DB

$$TIS(X,\tau,DB)=\begin{cases}1, \text{for } X\in\tau, \text{and } \tau\in DB\\ 0, \text{for } X\notin\tau, \text{and } \tau\in DB\end{cases}$$

Also, each itemset X in DB is associated with the following two attributes: transaction frequency (TF), and database frequency (DF). These two attributes are defined as follows:

$TF(X, DB) = \sum_{\tau\in DB} TIS(X, \tau, DB)$

$DF(X, DB) = \sum_{\tau \in DB} TIF(X, \tau, DB)$

Steinbach et. al [73] have proposed a generalized support measure using *eval* and *norm* functions. Steinbach and Kumar [72] have also proposed a framework that encompasses the traditional concept of confidence as a special case and can be used as the basis for designing a variety of new confidence measures. When an item is purchased multiple times in a transaction then the above generalized frameworks might not be adequate for mining association rules, since they are based on a binary database. The following example shows why the traditional support-confidence framework is not adequate for mining association rules in a TIMT type database.

Example 1.5.2. There are three TIMT type databases DB_1, DB_2, and DB_3 containing five transactions each. $DB_1 = \{\{A(1000), B(2000), C(1)\}, \{A(5), C(2)\}, \{B(4), E(2)\}, \{E(2), F(1)\}, \{F(2)\}\}$; $DB_2 = \{\{A(1), B(1), C(2)\}, \{A(1), B(1), E(2)\}, \{A(1), B(1), F(1)\}, \{A(1), B(1), G(2)\}, \{H(3)\}\}$; $DB_3 = \{\{A(500), B(600)\}, \{A(700), B(400), E(1)\}, \{A(400), B(600), E(3)\}, \{G(3)\}, \{A(200), B(500), H(1)\}\}$. The numbers of occurrences of itemset {A, B} in transactions of different databases are given below.

Table 1.5.1. Distributions of itemset $\{A, B\}$ in transactions of different databases

Database	Trans #1	Trans #2	Trans #3	Trans #4	Trans #5
DB_1	1000	0	0	0	0
DB_2	1	1	1	1	0
DB_3	500	400	400	0	200

From Table 1.5.1, we observe the following points regarding itemset $\{A, B\}$:

- It has high database frequency, but low transaction frequency in DB_1.
- In DB_2, it has high transaction frequency, but relatively low database frequency.
- It has high transaction frequency and high database frequency in DB_3. •

Based on the above observations, it might be required to consider database frequencies and transaction frequencies of $\{A\}$, $\{B\}$ and $\{A, B\}$ to study association between the items A and B. Thus, one could have the following categories of association rules in a

TIMT type database [7].

I. Association rules induced by transaction frequency of an itemset

II. Association rules induced by database frequency of an itemset

III. Association rules induced by both transaction frequency and database frequency of an itemset

1.5.3 Frameworks for mining association rules under different categories

In this section, we introduce a framework for each category of association rules discussed in Section 1.5.2. Each framework is based on traditional support-confidence framework for mining association rules in a binary database.

1.5.3.1 Framework for mining association rules under category I

Based on the number of transactions containing an itemset, we define *transaction-support* (*tsupp*) [7] of the itemset as follows.

Definition 1.5.1. Let X be an itemset in TIMT type database DB. Transaction-support of X in DB is given as follows: $tsupp\ (X, DB) = TF(X, DB) / |DB|$. •

Let X and Y be two itemsets in DB. An itemset X is *transaction-frequent* in DB if $tsupp\ (X, DB) \geq \alpha$, where α is the *minimum transaction-support* level. We define transaction-support of an association rule r: $X \rightarrow Y$ in DB as follows: $tsupp(r, DB) = tsupp(X \cap Y, DB)$. In other words, the transaction-support of association rule r in DB is the fraction of transactions containing both X and Y. We define *transaction-confidence* (*tconf*) of association rule r in DB as follows: $tconf(r, DB) = tsupp(X \cap Y, DB) / tsupp(X, DB)$. In other words, the transaction-confidence of association rule r in DB is the fraction of transactions containing Y among the transactions containing X. An association rule r in DB is *interesting* with respect to transaction frequency of an itemset if $tsupp(r, DB) \geq$ *minimum transaction-support*, and $tconf(r, DB) \geq$ *minimum transaction-confidence* (β).

The parameters, minimum transaction-support and minimum transaction-confidence are user-defined inputs, given to a category I association rule mining algorithm.

1.5.3.2 Framework for mining association rules under category II

The number of occurrences of an itemset in a database is also an important issue. Let $X = \{x_1, x_2, ..., x_k\}$ be an itemset in database *DB*. Also, let τ be a transaction in *DB*. Let item x_i be purchased η_i numbers at a time in τ, for $i = 1, 2, ..., k$. Then, $TIF(X, \tau, DB) = minimum\{\eta_1, \eta_2, ..., \eta_k\}$. Based on the frequency of an itemset in a database, we define *database-support* (*dsupp*) [7] of the itemset as follows.

Definition 1.5.2. Let *X* be an itemset in TIMT type database *DB*. Database-support of *X* in *DB* is given as follows: $dsupp(X, DB) = DF(X, DB) / |DB|$. •

An item in a transaction could occur more than once. Thus, $dsupp(X, DB)$ could be termed as the *multiplicity* of itemset *X* in *DB*. An important characteristic of a database is the average multiplicity of an item (*AMI*) in the database. Let *m* be the number of distinct items in *DB*. We define *AMI* in a TIMT type database *DB* as follows: $AMI(DB) = \sum_{i=1}^{m} dsupp(x_i, DB)/m$, where x_i is the *i*-th item in *DB*, for $i = 1, 2 \ldots$. An itemset *X* is *database-frequent* in *DB* if $dsupp(X, DB) \geq \gamma$, where γ is the *minimum database-support* level. Let *Y* be another itemset in *DB*. We define *database-support* of association rule *r*: $X \rightarrow Y$ in *DB* as follows: $dsupp(r, DB) = dsupp(X \cap Y, DB)$. In other words, database-support of association rule *r* in *DB* is the multiplicity of $X \cap Y$ in *DB*. Also, we define *database-confidence* (*dconf*) of association rule *r*: $X \rightarrow Y$ in *DB* as follows: $dconf(r, DB) = dsupp(X \cap Y, DB) / dsupp(X, DB)$. In other words, database confidence of association rule *r* in *DB* is the multiplicity of *Y* in the transactions containing *X*. An association rule *r*: $X \rightarrow Y$ is *interesting* with respect to database frequency of an itemset if $dsupp(r, DB) \geq$ *minimum database-support* and $dconf(r, DB) \geq$ *minimum database-confidence* (δ). The parameters, minimum database-support and minimum database-confidence, are user defined inputs given to a category II association rule mining algorithm.

1.5.3.3 Framework for mining association rules under category III

Interesting association rules under category III are based on interestingness measures defined in Sections 1.5.3.1 and 1.5.3.2. An association rule r: $X \rightarrow Y$ in TIMT type database *DB* is *interesting* with respect to both transaction frequency and database frequency of an itemset if $tsupp(r, DB) \geq \alpha$, $tconf(r, DB) \geq \beta$, $dsupp(r, DB) \geq \gamma$, and $dconf(r, DB) \geq \delta$. The parameters α, β, γ, and δ are defined in Sections 1.5.3.1 and 1.5.3.2. They are user-defined inputs given to a category III association rule mining algorithm. Based on the framework, we extract association rules in a database as follows.

Example 1.5.3. Let $DB = \{\{A(1), B(1)\}, \{A(2), B(3), C(2)\}, \{A(1), B(4), E(1)\}, \{A(3), E(1)\}, \{C(2), F(2)\}\}$. Let $\alpha = 0.4$, $\beta = 0.6$, $\gamma = 0.4$, and $\delta = 0.5$. Transaction-frequent itemsets and database-frequent itemsets are given in Tables 1.5.2 and 1.5.3, respectively.

Table 1.5.2. Transaction-frequent itemsets in *DB*

Itemset	*A*	*B*	*C*	*E*	*AB*	AE
tsupp	0.8	0.6	0.4	0.4	0.6	0.4

Table 1.5.3. Database-frequent itemsets in *DB*

Itemset	*A*	*B*	*C*	*E*	*F*	*AB*	*AC*	*AE*	*BC*	*CF*	*ABC*
dsupp	1.4	1.6	0.8	0.4	0.4	0.8	0.4	0.4	0.4	0.4	0.4

Interesting association rules under category III are given in Table 1.5.4.

Table 1.5.4. Interesting category III association rules in *DB*

r: $X \rightarrow Y$	*tsupp* (r, *DB*)	*tconf* (r, *DB*)	*dsupp* (r, *DB*)	*dconf* (r, *DB*)
$A \rightarrow B$	0.6	0.75	0.8	0.57143
$B \rightarrow A$	0.6	1.0	0.8	0.5

•

The goal of this chapter is to provide frameworks for mining association rules under different categories in a TIMT type database.

1.5.3.4 Dealing with items measured in continuous scale

The above framework works well for items that are measured in discrete scale. Now, we discuss the issue of handling items that are measured in continuous scale using an example. Consider the item milk in a departmental store. Let there four types of milk packets: 0.5 kilolitre, 1 kilolitre, 1.5 kilolitres, and 2 kilolitres. The minimum packing unit could be considered as 1 unit. Thus, 3.5 kilolitres of milk could be considered as 7 units of milk.

1.5.4 Properties of different interestingness measures

The following properties are based on a TIMT type database *DB*. Transactional support measure is the same as the traditional support measure of an itemset in a database. Thus, it satisfies all the properties that are satisfied by traditional support measure.

Property 1.5.1. *$0 \leq$ tsupp (Y, DB) $\leq$ tsupp (X, DB) ≤ 1, for itemsets X, Y in DB such that $X \subseteq Y$.* •

Transaction-support measure satisfies anti-monotone property [87] of traditional support measure. Transaction-confidence measure is the same as the traditional confidence measure of an association rule. Thus, it satisfies all the properties that are satisfied by traditional confidence measure of an association rule.

Property 1.5.2. *tconf (r, DB) lies in [tsupp(r, DB), 1], for an association rule r in DB.* •

If an itemset *X* is present in a transaction τ in *DB* then $TIF(X, \tau, DB) \geq 1$. Thus, we have the following property.

Property 1.5.3. *tsupp* $(X, DB) \leq$ *dsupp* $(X, DB) < \infty$, *for an itemset X in DB.* •

Let *BD* be a binary transactional database. Then the maximum database frequency of an itemset in *BD* is equal to $|BD|$. Thus, database-support of an itemset in *BD* lies in [0, 1]. Hence, database-confidence of an association rule *r* in *BD* lies in [*dsupp*(*r*, *BD*), 1]. In the context of a TIMT type database *DB*, database-confidence of an association rule *r* might not lie in [*dsupp*(*r*, *DB*), 1.0], since database-support of an itemset in *DB* might be greater than 1 (as reported in Table 1.5.3). But, the database confidence of an association rule *r* in *DB* satisfies the following property.

Property 1.5.4. *dconf* (*r*, *DB*) *lies in* [*0, 1*], *for an association rule r in DB.* •

1.5.5 Mining association rules

Association rule mining has received a lot of attention in KDD community. Many interesting algorithms have been proposed for mining positive association rules in a binary database [13], [39], [66]. Thus, there are several implementations [32] of mining positive association rules. In the context of mining association rules in a TIMT type database, we shall implement apriori algorithm [13], since it is simple and easy to implement. For mining association rules in a TIMT type database, one could apply apriori algorithm directly. For mining association rules under category III, the pruning step of interesting itemset generation requires testing on two conditions: minimum transaction-support and minimum database-support. The interesting association rules under category III satisfy the following two additional conditions: minimum transaction-confidence and minimum database-confidence.

1.5.6 Experiments

We have carried out several experiments to extract association rules under proposed frameworks. All the experiments have been implemented on a 2.8 GHz Pentium D dual core processor with 512 MB of memory using visual C++ (version 6.0) software. We present the experimental results using real databases *retail* (*R*) and *ecoli* (*E*). Also, we have used an artificial database *check* (*C*) to verify the result of the experiment. The database *retail* [34] is obtained from an anonymous Belgian retail supermarket store. The database *ecoli* is a subset of *ecoli database* [77]. The database *ecoli* has been processed for the purpose of conducting experiments. The database *check* contains the following transactions: {34, 47, 62}, {34, 55, 62, 102}, {47, 62}, {47, 55, 75}, {55, 62, 120}. We present some characteristics of these databases in Table 1.5.5.

Table 1.5.5. Database characteristics

Database	# transactions	Avg length of a transaction	Avg frequency of an item	# items
R	88,162	11.30576	99.67380	10000
E	336	7.00000	22.40000	90
C	5	3.00000	2.14286	7

Due to unavailability of TIMT type database, we have used above data by applying a preprocessing technique. If an item is present in a transaction then the number of occurrences of the item is generated randomly between 1 and 5. Thus, a binary transactional database gets converted into a TIMT type database. Now, database *check* contains the following transactions: {34, 2, 47, 1, 62, 5}, {34, 3, 55, 4, 62, 1, 102, 1}, {47, 2, 62, 3}, {47, 2, 55, 4, 75, 4}, {55, 1, 62, 5, 120, 3}. Each item in a transaction follows its transaction frequency. Initially, we perform experiments to extract association rules under category I. A few transaction-frequent itemsets in different databases are given in Table 1.5.6.

Table 1.5.6. Top 5 transaction-frequent itemsets in different databases (sorted on transaction-support)

Database	α	Itemset (transaction-support)				
R	0.05	{39} (0.57479)	{48} (0.47793)	{39, 48} (0.33055)	{38} (0.17690)	{32} (0.17204)
E	0.3	{50} (1.00000)	{48} (0.97619)	{48,50} (0.95833)	{44} (0.15095)	{40} (0.15002)
C	0.1	{62} (0.80000)	{47} (0.60000)	{55} (0.60000)	{34} (0.40000)	{34, 62} (0.40000)

A few association rules under category I in different databases are given in Table 1.5.7.

Table 1.5.7. Top 5 association rules in different databases under category I (sorted on transaction-support)

Database	(α, β)	(antecedent, consequent, transaction-support, transaction-confidence)				
R	(0.05, 0.2)	(48, 39, 0.33055, 0.69163)	(39, 48, 0.33055, 0.57508)	(41, 39, 0.12947, 0.76373)	(39, 41, 0.12947, 0.22524)	(38, 39, 0.11734, 0.66331)
E	(0.1, 0.3)	(48, 50, 0.95833, 0.98171)	(50, 48, 0.95833, 0.95833)	(44,48, 0.13988, 0.40286)	(40, 50, 0.13691 0.41039)	(37, 48, 0.13393 0.39571)
C	(0.1, 0.3)	(34, 62, 0.40000, 1.00000)	(47, 62, 0.40000, 0.66667)	(55, 62, 0.40000, 0.66667)	(62, 34, 0.40000, 0.50000)	(62, 47, 0.40000, 0.50000)

Also, we have computed average multiplicity of an item in the given databases. In Table 1.5.8, we present AMI for each of the given databases.

Table 1.5.8. AMI for the given databases

Database	*R*	*E*	*C*
AMI	0.00309	0.23062	1.17143

A few database-frequent itemsets in different databases are given in Table 1.5.9.

Table 1.5.9. Top 5 database-frequent itemsets in different databases (sorted on database-support)

Database	γ	Itemset (database-support)				
R	0.07	{39} (1.75906)	{0} (0.66511)	{1} (0.57741)	{8} (0.55327)	{38} (0.53030)
E	0.3	{50} (3.00595)	{48} (2.98800)	{48,50} (2.75833)	{44} (0.30045)	{40} (0.28276)
C	0.1	{62} (2.80000)	{55} (1.80000)	{34} (1.00000)	{47} (1.00000)	{75} (0.80000)

A few association rules under category II in different databases are given in Table 1.5.10.

Table 1.5.10. Top 5 association rules in different databases under category II (sorted on database-support)

Database	(γ, δ)	(antecedent, consequent, database-support, database-confidence)				
R	(0.07, 0.4)	(48, 39, 0.72554, 0.50731)	(39, 48, 0.72554, 0.42078)	(41, 39, 0.28436, 0.55841)	(38, 39, 0.25744, 0.48475)	(41, 48, 0.22580, 0.44341)
E	(0.1, 0.2)	(48, 50, 2.75833, 0.92314)	(50, 48, 2.75833, 0.91762)	(44, 48, 0.30045, 1.00000)	(40, 50, 0.28276 1.00000)	(37, 48, 0.28393 1.00000)
C	(0.1, 0.3)	(75, 55, 0.80000, 1.00000)	(55, 75, 0.80000, 0.44444)	(120, 62, 0.60000, 1.00000)	(34, 55, 0.60000, 0.60000)	(34, 62, 0.60000, 0.60000)

We observe the following interesting points in database *check*.

(i) Though the itemset {55, 75} has high database-support, but it has low transaction-support. It generates an interesting association rule under category II. But, it fails to generate an interesting association rule under category I.

(ii) Both the association rules {34} → {62} and {62} → {34} are interesting when transaction frequency of an itemset is considered. But, the association rule {62} → {34} is not interesting when database frequency of an itemset is considered.

The above observations show why we need to study association rules with respect to transaction frequency of an itemset and database frequency of an itemset.

Also, we have obtained execution times for extracting association rules at different database sizes. As the size of a database increases, the execution time also increases. We have observed such phenomenon in Figures 1.5.1 and 1.5.2.

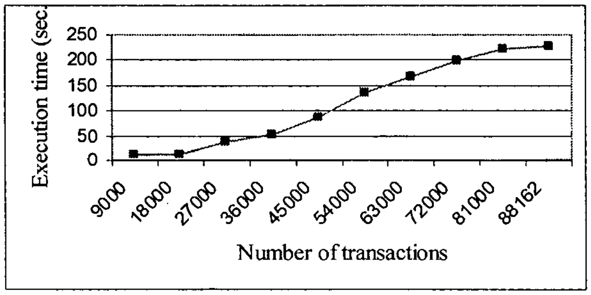


Figure 1.5.1. Execution time versus size of database *retail* at $\alpha = 0.05$, $\gamma = 0.07$, $\beta = 0.2$, and $\delta = 0.4$

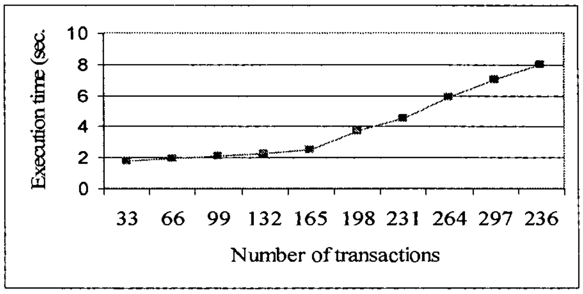


Figure 1.5.2. Execution time versus size of database *ecoli* at $\alpha = 0.1$, $\gamma = 0.1$, $\beta = 0.3$, and $\delta = 0.3$

Also, we have obtained execution times for extracting association rules at different minimum database-supports. As the value of minimum database-support increases, the number of interesting itemsets decreases. Thus, the number of interesting category II association rules decreases. So, the time required to extract category II association rules also decreases. We have observed such phenomenon in Figures 1.5.3 and 1.5.4.

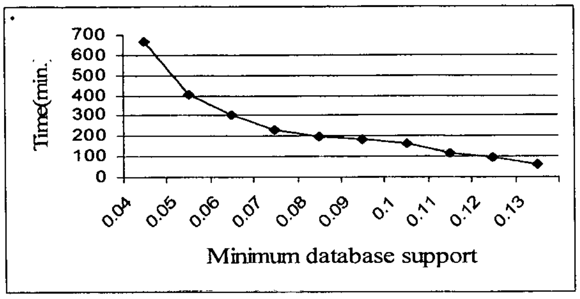


Figure 1.5.3. Execution time versus γ at $\alpha = 0.04$, $\beta = 0.2$, and $\delta = 0.4$ (*retail*)

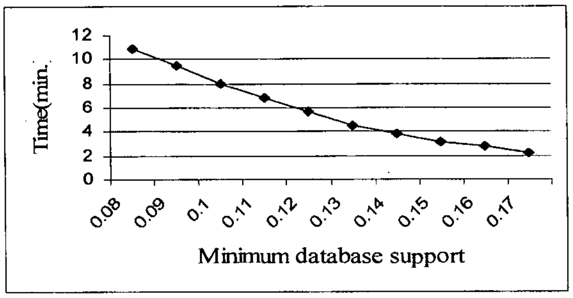


Figure 1.5.4. Execution time versus γ at $\alpha = 0.08$, $\beta = 0.3$, and $\delta = 0.2$ (*ecoli*)

The graph of execution time versus α at a given tuple (γ, β, δ) is similar to the graph of execution time versus γ at a given tuple (α, β, δ). As the value of minimum transaction-support increases, the number of interesting itemsets decreases. Thus, the number of interesting category I association rules decreases. So, the time required to extract category I association rules also decreases.

1.5.7 Related work

Association rule mining finds interesting association between two itemsets in a database. The notion of association rule is introduced by Agrawal et al. [11]. The authors have proposed an algorithm to mine frequent itemsets in a database. Many algorithms have been reported to extract association rules in a database. In the following we mention a few interesting algorithms for extracting association rules in a database. Agrawal and

Srikant [13] have proposed apriori algorithm that uses breadth-first search strategy to count the supports of itemsets. The algorithm uses an improved candidate generation function, which exploits the downward closure property of support and makes it more efficient than earlier algorithm. Han et al. [39] have proposed data mining method FP-growth (frequent pattern growth) which uses an extended prefix-tree (FP-tree) structure to store the database in a compressed form. FP-growth adopts a divide-and-conquer approach to decompose both the mining tasks and databases. It uses a pattern fragment growth method to avoid the costly process of candidate generation and testing. Savasere et al. [66] have introduced partition algorithm. The database is scanned only twice. In the first scan the database is partitioned and in each partition support is counted. Then the counts are merged to generate potential frequent itemsets. In the second scan, the potential frequent itemsets are counted to find the actual frequent itemsets. Wu and Zhang [81] have proposed a weighting model for synthesizing high-frequent association rules from different databases.

In the context of interestingness measures, Tan et al. [75] present an overview of twenty one interestingness measures proposed in the statistics, machine learning and data mining literature. The measures in general lack to agree with each other. However, the authors show that if support-based pruning or table standardization (of the contingency tables) is used, the measures become highly correlated. Brin et al. [22] introduce measures conviction and lift as the improvements to confidence based on implication rules. Aggarwal and Yu [10] point out weaknesses of the large frequent itemset method using support and that lift gives only values close to one for items which are very frequent, even if they are positively correlated. The authors have introduced collective strength. Collective strength uses the violation rate for an itemset which is the fraction of transactions which contains some, but not all items of the itemset.

1.5.8 Conclusion

The traditional support-confidence framework for mining association rules is based on a binary database. It has limited usage in association analysis of items, since a real life transaction might contain an item multiple times.

The traditional support-confidence framework is based on the frequency of an itemset in a binary database. In a TIMT type database, there are two types of frequency of an itemset viz., transaction frequency, and database frequency. Due to these reasons, we get the following categories of association rules in a TIMT type database: (i) Association rules induced by transaction frequency of an itemset, (ii) Association rules induced by database frequency of an itemset, and (iii) Association rules induced by both transaction frequency and database frequency of an itemset. We have introduced a framework for mining each category of association rules. The proposed frameworks are effective for studying association among items in real life market basket data.

Chapter 1.6

Conclusion

It seems many interesting and useful patterns remain undiscovered. In Chapter 1.2, we have proposed conditional patterns in a database. Conditional patterns provide interesting knowledge about items in frequent itemsets in a database. They could be useful for solving many problems. In Chapter 1.3, we have dealt with mining arbitrary Boolean expressions induced by frequent itemsets using conditional patterns in a database. We have provided a simple framework for synthesizing an arbitrary Boolean expression. In future, we shall search for more applications of conditional patterns in a database.

Many data mining problems could be solved by capturing association among items in a database. In Chapter 1.4, we have proposed measure A_2 for this purpose. We have observed that measure A_2 is effective in capturing statistical association among items in a database. Also, we have introduced the notion of associative itemset in a database. In Chapter 2.4, we have proposed a technique for clustering frequent items in multiple databases using the measure of association A_2. In future, we shall search for more applications of the measure of association A_2.

Traditional support-confidence framework has not been effective in finding association rules in real market basket data. An item in a database could be purchased multiple times in a transaction. In this case, there are two types of frequency of an itemset in a database: the number of transactions in the database containing the itemset, and the number of occurrences of the itemset in the database. Thus, one could study association rules with respect to these types of frequency of an itemset. We have proposed frameworks for three different categories of association rules in a database. We believe that such framework would help studying association between a pair of itemsets in real market basket data.

Part 2

Pattern recognition in multiple databases

Chapter 2.1

Introduction

There are various reasons why mining on multiple databases becomes an important issue in the recent time. In the following paragraph, we mention a few reasons that motivate us to work on mining multiple databases.

- Due to a liberal economic policy adopted by many countries across the globe, the number of branches of a multi-national company as well as the number of multi-national companies is increasing over time. Moreover, the economies of many countries are growing at a faster rate. As a result the number of multi-branch companies within a country is also increasing. Many multi-branch companies deal with multiple databases, since local transactions are stored locally. Thus, it is necessary to study data mining on multiple databases. Many decision-making problems are based on knowledge distributed across the branch databases.
- Most of the previous pieces of data mining work are based on a single database. Thus, it is necessary to study data mining on multiple databases.
- The number of data mining solution providers is increasing over time. Please visit http://www.kdnuggets.com/ for reference.
- In addition, the number of data mining products is also increasing over time. Please see the above site for reference.
- In the year 2002, Dr Shichao Zhang has submitted his thesis on multi-database mining. His research has made significant impact on data mining community. So, it is a recent topic in data mining, and thus it deserves much attention.

In discovering knowledge in a database, we have come across various types of patterns. Some examples of patterns in a database are frequent itemset, positive association rule, negative of association rule, and conditional pattern. In Part 2, we have studied various patterns originated due to a study of multiple databases. These patterns are not only interesting, but also help solving different problems. We have made the following contributions in Part 2.

- We have proposed a definition of global exceptional frequent itemset in multiple databases.
- The notion of exceptional sources for a global exceptional frequent itemset is introduced.
- We have designed an algorithm for synthesizing global exceptional frequent itemsets.
- An extended model is proposed for synthesizing global patterns from local patterns in different databases.
- The notion of heavy association rule in multiple databases is introduced, and an algorithm for synthesizing such association rules in multiple databases is thus proposed.
- The notion of exceptional association rule in multiple databases is introduced, and an extension is made to the proposed algorithm to notify whether a heavy association rule is high-frequent or exceptional.
- We have designed an algorithm for clustering frequent items in multiple databases.

Chapter 2.2

Synthesizing global exceptional patterns in multiple databases

Many multi-branch companies transact from different locations. Many of them collect a huge amount of transactional data continuously through their different branches. Due to a growth-oriented and liberal economic policy adopted by many countries across the globe, the number of such companies as well as the number of branches of such a company is increasing over time. Moreover, most of the pieces of data mining work are based on a single database. Thus, it is important to study data mining on multiple databases. Analysis and synthesis of patterns in multiple databases is an important as well as interesting issue.

Based on the number of data sources, patterns in multiple databases could be classified into three categories. They are local patterns, global patterns, and patterns that are neither local nor global. A pattern based on a branch database is called a *local pattern.* On the other hand, a *global pattern* is based on all the databases under consideration. Global patterns are useful for global data analyses and global decision making problems [6], [79], [83]. There exist other types of patterns in multiple databases. For example, frequent itemset, positive associative rule and clustering of relevant objects. There is no fixed set of attributes to describe these patterns, since there are different types of pattern in a database. Each type of pattern could be described by a specific set of attributes.

Itemset patterns influence KDD research heavily in following ways: Firstly, many interesting algorithms have been reported on mining itemset patterns in a database [13], [39], [66]. Secondly, an itemset could be considered as a basic type of pattern in a transactional database, since many patterns are derived from the itemset patterns in a database. Some examples of derived patterns are positive association rule [11], negative association rule [82], conditional pattern [9] in a database and high-frequent association rule [81], heavy association rule [5], exceptional association rule [5] in multiple databases. Considerable amount of work have been reported on mining / synthesizing such derived patterns in databases. Thirdly, solutions of many problems are based on the analysis of patterns in a database. Such applications [83], [79] process patterns in a database for the purpose of making some decisions. Thus, mining and analysis of itemset patterns in a database is an interesting as well as important issue. Each itemset in a database is associated with a statistical measure, called *support* [11]. The support (supp) of an itemset X in database D could be defined as the fraction of transactions in D containing all the items of X, denoted by $supp(X, D)$. In most of the cases, the importance of an itemset is judged by its support. The itemset X is *frequent* in D if $supp(X, D) \geq \alpha$, where α is user defined *minimum support.* Let $FIS(D)$ be the set of frequent itemsets in D. Frequent itemsets determine major characteristics of a database. Wu et al. [80] have proposed a solution of inverse frequent itemset mining. They argued that one could efficiently generate a synthetic market basket database from the frequent itemsets and their supports. Let X and Y be two itemsets in D. The characteristics of D are revealed more by the pair $(X, supp(X, D))$ than that of $(Y, supp(Y, D))$, if $supp(X, D) > supp(Y, D)$. Thus, it is important to study frequent itemsets more than infrequent itemsets. Here, we study frequent itemsets in multi-databases.

In the next section, we have defined global exceptional frequent itemset in multi-databases. Also, we have designed an algorithm for synthesizing global exceptional frequent itemsets in multiple databases. There are useful applications of global exceptional frequent itemsets. For example, the company might plan to collect the feedback of customers for the global exceptional products and implement similar strategies to increase the sales of other products. Also, the company could identify the branches having high sales of the global exceptional items. It might plan to manufacture and / or procure such items locally to reduce the transportation cost. The global exceptional frequent itemsets would affect many such decisions of a multi-branch company.

The first question comes to our mind whether a traditional data mining technique could deal with the multiple large databases. To apply a traditional data mining technique we need to amass all the databases together. A single computer might take unreasonable amount of time to process the entire database. Sometimes, it might not be feasible to carry out the mining task. Another solution would be to employ parallel machines and the associated software. But, it requires high investment on hardware and software. Moreover, it is difficult to find local patterns when mining techniques are applied on the entire database. Thus, a traditional data mining techniques is not suitable in this situation. So, it is a different problem. Hence, it is required to be dealt with in a different way. We would employ the model of local pattern analysis [91] for mining multiple databases. Under this model of mining multiple databases, each branch requires to mine local database using a traditional data mining technique. Afterwards, each branch is required to forward the pattern base to the central office. Then, the central office would process the pattern bases collected from different branches for synthesizing the global patterns, or making decisions related to some problems.

The rest of the chapter is organized as follows. We discuss related work and state the problem in Section 2.2.2. In Section 2.2.3, we discuss a simple method for synthesizing support of an itemset in the union of all databases. In Section 2.2.4, we design an algorithm to synthesize global exceptional frequent itemsets in multi-databases. In Section 2.2.5, we define two types of errors for synthesizing global exceptional frequent itemsets. We present experimental results in Section 2.2.6.

2.2.2 Problem statement

Consider a multi-branch company that has n branches. Let D_i be the database corresponding to the i-th branch, for $i = 1, 2, \ldots, n$. Also, let D be the union of these databases. In the context of multiple databases, one could conceive the idea of global exceptional frequent itemset in two ways: Firstly, a frequent itemset extracted from most of the databases might have moderate support in D. Secondly, a frequent itemset extracted from a few databases might have high support in D. We are interested in the second category of frequent itemsets and call them as global exceptional frequent itemsets henceforth. Before we define a global exceptional frequent itemset formally, we first study work related to this issue.

2.2.2.1 Related work

Multi-database mining has been recently recognized as an important research topic in KDD community. Zhang et al. [93] studied knowledge discovery in multiple databases using local pattern analysis.

Zhang et al. [94] have proposed a strategy for mining local exceptions in multiple databases. The authors have defined an exceptional pattern as follows:

A pattern p in local instances is an exceptional pattern if EPI(p) $\geq$ *minEP*, where *EPI(p)* is an interestingness measure of *p* and has been defined as follows:

- $$EPI(p) = \frac{nExtrn(p) - avgNoExtrn}{-avgNoExtrn} \qquad (2.2.1)$$

  where, *nExtrn(p)* and *avgNoExtrn* are the number of times *p* gets extracted and the average number times a pattern gets extracted from different data sources, respectively.
- *minEP* is the user-defined threshold for minimum interest degree.

Also, the authors have defined interestingness of a pattern in a branch as follows:

A pattern p in i-th branch is of interest if $RI_i(p) \geq$ *minEPsup*, where $RI_i(p)$ is interestingness degree of *p* in *i*-th branch and has been defined as follows:

- $RI_i(p) = (supp(p,\ D_i) - \alpha_i)/\alpha_i$ (2.2.2)

  α_i is the minimum support given for mining D_i, for $i = 1, 2, \ldots, n$.
- *minEPsup* is the user-defined threshold for minimum interest degree.

From the above two definitions, we observe the following points:

- The definition of exceptional pattern is considered with respect to all the databases. The definition of interestingness of a pattern is considered with respect to a local database. Thus, an exceptional pattern in multiple databases and interestingness of the pattern in a local branch are of two different issues.
- For a pattern p in local instances, the authors have shown that $0 < EPI(p) \leq 1$. We take the following example to show that the above property does not hold always. Let there are only 4 patterns in 15 databases. The number of extractions of 4 patterns are given as follows: $nExtrn(p_1) = 2$, $nExtrn(p_2) = 15$, $nExtrn(p_3) = 4$, $nExtrn(p_4) = 5$. Thus, $avgNoExtrn = 26/4 = 6.5$. $EPI(p_1) = (2\text{-}6.5)/(\text{-}6.5) = 0.69$, $EPI(p_2) = (15\text{-}6.5)/(\text{-}6.5) = \text{-}1.31$, $EPI(p_3) = (4\text{-}6.5)/(\text{-}6.5) = 0.38$, $EPI(p_4) = (5\text{-}6.5)/(\text{-}6.5) = 0.23$. Thus, $EPI(p_2) \notin (0, 1]$.
- An interesting exceptional pattern might not emerge as a global exceptional pattern, since the support of the pattern is not considered in the union of all databases.

We feel that an exceptional global frequent itemset should be constrained on the number of times it gets extracted and its support in the union of all databases. Thus, none of the above two definitions, nor the both the definitions together does serve as a definition of exceptional global frequent itemset in multiple databases.

Zhang et al. [89] have proposed a technique for identifying global exceptional patterns in multiple databases. The authors have described global exceptional pattern as follows:

Global exceptional patterns are highly supported by only a few branches, that is to say, these patterns have very high support in these branches and zero support in other branches.

From the above descriptions, we observe the following points:

- Let there are ten branches of a multi-branch company. A pattern p has very high support in first two databases that have small sizes. Also, p does not get extracted from the remaining databases. According to the above description, p is a global exceptional pattern. We observe that pattern p might not have high support in the union of all databases. Thus, such description does not serve the purpose.
- Also, it is not necessarily true that an exceptional pattern will have zero supports in the remaining databases.

Thus, the above description does not describe a global exceptional pattern in true sense. Also, we observe the following points regarding algorithm *IdentifyExPattern* [89] for identifying exceptional patterns in multiple databases.

- We believe that the size (i.e., the number of transactions) of a database and support of an itemset in the database are two important parameters for determining the presence of an itemset in a database, since the number of transactions containing the itemset X in a database D_1 is equal to $supp(X, D_1) \times size(D_1)$. The algorithm does not consider size of a database to synthesize the global support of a pattern. Global support of a pattern has been synthesized using only supports of the pattern in concerned databases. We take following example to illustrate this issue. Let there are two databases D_1 and D_2, where $size(D_1)$ is significantly larger than $size(D_2)$. At a given α, we assume that pattern p does not get extracted from D_2, and pattern q does not get extracted from D_1. Thus, $supp(p, D_2)$ and $supp(q, D_1)$ both are assumed as 0s. Then, $supp(p, D_1 \bigcup D_2)$ could be synthesized by $[supp(p, D_1) \times size(D_1) + 0 \times size(D_2)] / size(D_1 \bigcup D_2)$. If $supp(p, D_1) < supp(q, D_2)$ then it might so happen that $supp(p, D_1 \bigcup D_2) > supp(q, D_1 \bigcup D_2)$. In particular, let $size(D_1) = 10000$, $size(D_2) = 100$. At $\alpha = 0.05$, let $supp(p, D_1) = 0.1$, $supp(q, D_1) = 0$, $supp(p, D_2) = 0$, and $supp(q, D_2) = 0.2$. We note that $supp(p, D_1) < supp(q, D_2)$. But, $supp(p, D_1 \bigcup D_2) = 0.99$, and $supp(q, D_1 \bigcup D_2) = 0.002$. So, $supp(p, D_1 \bigcup D_2) > supp(q, D_1 \bigcup D_2)$. Thus, the size of a database is an important parameter for synthesizing the support of a pattern in the union of all databases.

- The algorithm does not identify global exceptional patterns correctly in all the situations. For example, let there are 10 similar databases. Assume that the number of times each pattern gets extracted is either 8, or 9, or 10. Thus, these patterns are supported by most of the databases. According to the nature of global exceptional patterns, a high voted pattern is not a global exceptional pattern. But, the algorithm would report some of them as global exceptional patterns.
- The algorithm returns patterns that have high supports among the patterns that are extracted less than average number of times. We feel that a global exceptional pattern should have the following properties: (i) the support of a global exceptional pattern in the union of all databases is greater than or equal to a threshold value, and (ii) the number of extractions of a global exceptional pattern is less than another threshold value, where these threshold values are user-defined.

In the context of association rule in multiple databases, Wu and Zhang [81] have proposed a technique for synthesizing high-frequent association rules in different data sources.

In the context of support estimation of frequent itemsets, Jaroszewicz and Simovici [45] have proposed a method for estimating supports of frequent itemsets using Bonferroni-type inequalities [35]. Also, the maximum-entropy approach to support estimation of a general Boolean expression is proposed by Pavlov et al. [62]. But, these support estimation techniques are suitable for problems that deal with single database.

Existing parallel mining techniques [12], [26], [28] could also be used to deal with multiple databases.

2.2.2.2 Our approach

The difficulty of synthesizing global exceptional frequent itemsets is that a frequent itemset in a database may not get extracted from all the databases. Apart from synthesized support of an itemset in *D*, the number of extractions of the itemset is an important issue. An itemset may be high-frequent or, low-frequent, or neither high-frequent nor low-frequent. In this context, we need to consider only low-frequent itemsets. We could arrive in such a conclusion only if we have predefined threshold of minimum number of extractions. The problem of synthesizing global exceptional frequent itemsets is similar to awarding distinction grade to students in our examination system. In particular, a student could be awarded distinction grade if he/she gets average marks greater than or equal to 70%, and attends more than 75% of classes, provided the student passes all the papers. In a similar way, an exceptional frequent itemset in multiple databases could be judged against two thresholds, viz., high support and low extraction. We use the symbol γ to denote the threshold of low extraction of an itemset, where $0 < \gamma \leq 1$. Thus, we define a low-voted frequent itemset as follows.

Definition 2.2.1. An itemset X has been extracted from k out of n databases. Then X is low-voted, if $k < n \times \gamma$, where γ is the user defined threshold of low extraction. •

Among low-voted itemsets, we shall search for global exceptional frequent itemsets. An itemset may not get extracted from all the databases. Sometimes we need to estimate the support of an itemset in a database to synthesize the support of the itemset in D. Let $supp_a(X, D_i)$ and $supp_e(X, D_i)$ be the actual and estimated support of an itemset X in D_i, respectively, for $i = 1, 2, \ldots, n$. We use the symbol μ to denote the threshold of high support for an itemset in a database, where $\alpha < \mu \leq 1$. For a single database, we define an itemset with high support as follows:

Definition 2.2.2. Let X be an itemset in database D_i, for some $i = 1, 2, \ldots, n$. X possesses high support in D_i if $supp_a(X, D_i) \geq \mu$, where μ ($> \alpha$) is the user defined threshold of high support, for some $i = 1, 2, \ldots, n$. •

The method of synthesizing support of an itemset is discussed in Section 2.2.3. Let $supp_s(X, D)$ denote the synthesized support of the itemset X in D. For multiple databases, we define an itemset with high support as follows:

Definition 2.2.3. Let D be the union of all branch databases. An itemset X in D possesses high support if $supp_s(X, D) \geq \mu$, where μ is the user-defined threshold of high support. •

Based on the concepts stated above, we define a global exceptional frequent itemset [2] in D as follows:

Definition 2.2.4. Let D be the union of all branch databases. Let X be a frequent itemset in some branch databases. Then X is global exceptional in D if it is low-voted and possesses high support in D. •

Based on the above discussion, it might be worth noting the following points: (i) A global exceptional frequent itemset in D is low-voted. (ii) A low-voted frequent itemset in D is not necessarily be global exceptional. (iii) A global exceptional frequent itemset in D has high support. (iv) An itemset with high support in D is not necessarily be global exceptional.

Let X be a global exceptional frequent itemset in D. Without loss of generality, let X be extracted from $D_1, D_2, \ldots, D_k$, for $0 < k < n$. Support of X in D_i is $supp_a(X, D_i)$, for $i = 1, 2, \ldots$, k. Then the average of these supports is obtained by the following formula:

$$avg(supp(X), D_1, D_2, \ldots, D_k) = \left(\sum_{i=1}^{k} supp_a(X, D_i)\right)/k \qquad (2.2.3)$$

Database D_i is called an *exceptional source* [2] with respect to the global exceptional frequent itemset X, if $supp_a(X, D_i) \geq avg(supp(X), D_1, D_2, \ldots, D_k)$, for some $i = 1, 2, \ldots, k$. We take an example to explain this issue. Let X be a global exceptional frequent itemset in D, and it has been extracted from D_1, D_2, and D_3. $supp_a(X, D_1) = 0.09$, $supp_a(X, D_2) = 0.17$, and $supp_a(X, D_3) = 0.21$. Then, $avg(supp(X), D_1, D_2, D_3) = (0.09 + 0.17 + 0.21) / 3 = 0.15667$. The databases D_2 and D_3 are exceptional sources with respect to the global exceptional frequent itemset X, since $0.17 > 0.15667$, and $0.21 > 0.15667$. We state the problem as follows:

Let there are n databases $D_1, D_2, \ldots, D_n$, and D be the union of these databases. Let $FIS(D_i)$ be the set of frequent itemsets in D_i, for $i = 1, 2, \ldots, n$. Find the global exceptional frequent itemsets in D using $FIS(D_i)$, for $i = 1, 2, \ldots, n$. Also, report the exceptional sources for the global exceptional frequent itemsets in D.

2.2.3 Synthesizing support of an itemset

Synthesizing support of an itemset in multiple databases is an important issue. A method of synthesizing frequent itemsets in D has been proposed in [5]. It deals with multiple real databases. The method is explained as follows.

The trend of the customers' behaviour exhibited in one database is usually present in other databases, since databases are real. In particular, a frequent itemset in a database is usually present in some transactions of other databases even if it does not get extracted there. The estimation procedure captures such trend and estimates the support of an itmset that fails to get extracted in a database. The estimated support of a missing itemset usually reduces the error of synthesizing a frequent itemset in multiple databases. If an itemset X fails to get extracted from database D_1, then we assume that D_1 contributes some amount of support for X. The support of X in D_1 satisfies the following inequality: $0 \leq \text{supp}_a(X, D_1) < \alpha$. The estimated support of such an itemset is called *average low-support* (*als*). The procedure of estimating *als* is discussed below.

Let the itemset X be extracted from m databases, for $1 \leq m < n$. Without loss of generality, we assume that X has been extracted from the first m databases. We shall use the average behaviour of the customers of the first m branches to estimate the average behaviour of the customers in remaining branches. Let $D_{1,m}$ be the union of D_1, D_2, ..., and D_m. $supp_a(X, D_{1,m})$ could be viewed as the average behaviour of customers of the first m branches with respect to X. $supp_a(X, D_{1,m})$ could be obtained by the following formula.

$$supp_a(X, D_{1,m}) = \left(\sum_{i=1}^{m} supp_a(X, D_i) \times size(D_i)\right) / \sum_{i=1}^{m} \text{size}(D_i) \quad (2.2.4)$$

One could estimate the support of X for each of the remaining $(n\text{-}m)$ databases as follows.

$$als(X, D_i) = \alpha \times supp_a(X, D_{1,m}), \text{ for } i = m + 1, m + 2, \ldots, n \quad (2.2.5)$$

The technique discussed above might not be suitable for synthesizing global exceptional frequent itemsets. The reason is given as follows. A global exceptional frequent itemset X gets extracted from a few databases. During the process of synthesis, we need to estimate the supports of X for the remaining databases. So, the number of actual supports of X is much less than the number of estimated supports of X. Thus, the error of synthesizing the support of X in D would be high. Therefore, we shall follow a different strategy for synthesizing support of an itemset in D. The strategy is explained as follows. We shall mine each database at a reasonably low value of α. If the itemset X fails to get extracted from D_i then we assume that $supp_a(X, D_i) = 0$, for some $i = m + 1, m + 2, \ldots, n$. The itemset X is present in D_i, for $i = 1, 2, \ldots, m$. Then, the number of the transactions containing X in D_i is $supp_a(X, D_i) \times size(D_j)$, for $i = 1, 2, \ldots, m$. Also, the itemset X is not present in D_i, for $i = m + 1, m + 2, \ldots, n$. We assume that the estimated number of the transactions containing X in D_i is 0, for $i = m + 1, m + 2, \ldots, n$. Thus, the estimated support of X in a database is given as follows:

$$supp_e(X, D_i) = \begin{cases} supp_a(X, D_i), \text{for } i = 1, 2, \ldots, m \\ 0, \text{for } i = m+1, m+2, \ldots, n \end{cases} \tag{2.2.6}$$

The synthesized support of X in D could be obtained by the following formula.

$$supp_s(X, D) = \left(\sum_{i=1}^{n} supp_e(X, D_i) \times size(D_i)\right) / \sum_{i=1}^{n} size(D_i) \tag{2.2.7}$$

2.2.4 Synthesizing global exceptional itemsets

In this section, we present an algorithm for synthesizing global exceptional frequent itemsets in D. We discuss here various data structures required to implement the algorithm. Let N be the number of frequent itemsets in $D_1, D_2, \ldots$, and D_n. The frequent itemsets are kept in a 2-dimensional array *FIS*. The (i, j)-th element of *FIS* stores the j-th frequent itemset extracted from D_i, for $j = 1, 2, \ldots, |FIS(i)|$, and $i = 1, 2, \ldots, n$. An itemset could be described by the following attributes: *itemset*, *supp* and *did*. The attributes *itemset*, *supp* and *did* represent itemset, support and database identification of the corresponding frequent itemset, respectively. Synthesized global exceptional frequent itemsets are kept in array *synFIS*. Each global exceptional itemset has been described by the following attributes: *itemset*, *ssupp*, *nSources*, *databases*, *nExSources*, and *exDbases*. The attributes *itemset* and *ssupp* represent the itemset and synthesized support of a global exceptional frequent itemset in *D*, respectively. The attributes *nSources* and *databases* store the number of sources of exceptional frequent itemsets and the list of identifications of source databases of a global exceptional frequent itemset, respectively. The attributes *nExSources* and *exDbases* store the number of exceptional sources and the list of identifications of exceptional sources for a global exceptional frequent itemset, respectively. The algorithm [2] is given below:

Algorithm 2.2.1. Synthesize global exceptional frequent itemsets in the union of all branch databases.

```
procedure ExceptionalFrequentItemsetSynthesis ($n$, $FIS$, $\mu$, $size$, $\gamma$)
Input:
$n$: number of databases
$FIS$: array of frequent itemsets
$\mu$: threshold of high support
$size$: array of total number of transactions in different databases
$\gamma$: threshold of low extraction
Output:
Global exceptional frequent itemsets in $D$
01: collect all the frequent itemsets into array FIS;
02: sort frequent itemsets in FIS in non-decreasing order on itemset attribute;
03: calculate total number of transactions into totTrans;
04: let $nSynFIS = 0$; let $curPos = 1$;
05: while ($curPos \leq N$) do
06:     let $i = curPos$; let $count = 0$;
07:     let $nTransCurFIS = 0$; $totSupp = 0$;
08:     while ($i \leq curPos + n$) do
09:       if ($FIS(i).itemset = FIS(curPos).itemset$) then
10:         update count, sources(count), totalSupp, nTransCurFIS, supports(count);
11:       else break;
```

```
12:      increase $i$ by 1;
13:    end if
14:  end while
15:  if (($count$ / $n$) < $\gamma$) and ($nTransCurFIS$ / $totTrans \geq \mu$)) then
16:    increase $nSynFIS$ 1;
17:    update attributes $supp$, $itemset$ and $nSources$ of $synFIS(nSynFIS)$;
18:    $avgSupp = totalSupp / count$; $exCount = 0$;
19:    for $j = 1$ to $count$ do
20:      $synFIS(nSynFIS).databases(j) = source(j)$;
21:      if ($supports$(exCount) $\geq avgSupp$) then
22         increase $exCount$ by 1;
23:        $synFIS(nSynFIS).exDbases(exCount) = sources(j)$ ;
24:      end if
25:    end for
26:    $synFIS(nSynFIS).nExSources = exCount$;
27:  end if
28:  update $curPos$ by $i$;
29:  end while
30:  sort itemsets of $synFIS$;
31:  for $i = 1$ to $nSynFIS$ do
32:    display details of $synFIS(i)$;
33:  end for
end procedure
```

We explain here the above algorithm. The frequent itemsets having the same *itemset* attribute are kept consecutive in *FIS*. It helps processing one itemset at a time. An exceptional frequent itemset is synthesized using the lines 8-27. The array *sources* is used to store the database identifications of all the databases that report the current frequent itemset. Also, the array *supports* is used to store the supports of the current frequent itemset in different databases. The information about the current itemset is obtained by the while-loop in lines 8-14. The information includes the number of extractions, database identifications of the source databases, supports in different databases, total supports, and the number of transactions containing current frequent itemset in different databases. We explain the update-statement at line 10 as follows. The number of extraction of current frequent itemset i.e., *count* is increased by 1. The *did* of the corresponding database is copied into cell *sources(count)*. Variable *nCurFIS* is added by expression *FIS*(*i*).*supp* × *size*(*FIS*(*i*).*did*). Variable *totalSupp* is also added by expression *FIS*(*i*).*supp*. The support of frequent itemset in the current database is copied into *supports*(*count*). We also explain the update-statement at line 17 as follows. The synthesized support of current global exceptional frequent itemset is obtained by expression *nCurFIS* / *totTrans*. The *itemset* attribute of current global exceptional frequent itemset is the same as the *itemset* attribute of current frequent itemset. The variable *count* is copied into *synFIS*(*nSynFIS*).*nSources*. The if-statement in lines 15-27 checks whether the current itemsets is a global exceptional frequent itemset in multiple databases and it synthesizes each global exceptional frequent itemset. All the frequent itemsets are processed using the lines 4-29. We sort global exceptional itensets at line 30 for better presentation. All the global exceptional itemsets are kept in non-decreasing order on the length of the itemset. Again, the itemsets of the same length are sorted on non-increasing order on support of the itemset. Finally, global exceptional itemsets are displayed using lines 31-33. We display all the global exceptional itemsets and their synthesized supports. For every global exceptional frequent itemset, we display the source databases from which it has been extracted. Also, for every global exceptional frequent itemset, we also display the exceptional source databases from which it has been highly supported.

Theorem 2.2.1. *The time complexity of the procedure ExceptionalFrequentItemsetSynthesis is maximum*{*O*(*N* × *log*(*N*)), *O*(*n* × *N*)}, *where N is the number of frequent itemsets extracted from n databases.*

Proof. The time complexity of line 1 is O(N), since there are N frequent itemsets in all the databases. Line 2 sorts N frequent itemsets in O($N \times \log(N)$) time. The time complexity of line 3 is O(n), since there are n databases. The program segment lines 5-29 repeats maximum N times. Within this program segment, there is a while-loop and a for-loop. The while-loop in lines 8-14 takes O(n) time. Also, the for-loop in lines 19-25 takes O(n) time. Thus, the time complexity of the program segment lines 5-29 is O($n \times N$). Line 30 takes O($N \times \log(N)$) time for sorting maximum N synthesized global exceptional itemsets. To display the details of a global exceptional itemset it takes O(n) time, since there are maximum n sources of the itemset. Thus, the program segment in lines 31-33 take O($n \times N$) time. Therefore, the time complexity of the procedure *ExceptionalFrequentItemsetSynthesis* is *maximum*{O($N \times \log(N)$), O($n \times N$)}. •

In the following example, we manually execute the above algorithm and verify that it works correctly.

Example 2.2.1. Let D_1, D_2 and D_3 be three databases of sizes 4000 transactions, 3290 transactions, and 10200 transactions, respectively. Let D be the union of D_1, D_2, and D_3. Assume that $\alpha = 0.1$, $\gamma = 0.4$, and $\mu = 0.25$. Let $X(\eta)$ denote the frequent itemset X with support η. The sets of frequent itemsets extracted from these databases are given as follows. $FIS(D_1)$ = {A(0.12), B(0.14), AB(0.11), C(0.20)}, $FIS(D_2)$ = {A(0.10), B(0.20), C(0.25), D(0.16), CD(0.12), E(0.16)}, $FIS(D_3)$ = {A(0.11), C(0.60), F(0.77)}. We keep frequent itemsets in array *FIS* and sort them on *itemset* attribute. The sorted frequent itemsets are given in Table 2.2.1.

Table 2.2.1. Sorted frequent itemsets in different databases

itemset	*A*	*A*	*A*	*B*	*B*	*C*	*C*	*C*	*D*	*E*	*F*	*AB*	*CD*
supp	.12	.10	.11	.14	.20	.20	.25	.60	.16	.16	.77	.11	.12
did	1	2	3	1	2	1	2	3	2	2	3	1	2

Here, *totTrans* is 17490. We synthesize the frequent itemsets in *FIS*. Synthesized frequent itemsets are given in Table 2.2.2.

Table 2.2.2. Synthesized frequent itemsets in multi-databases

itemset	A	B	C	D	E	F	AB	CD
synthesized *supp*	0.11	0.07	0.44	0.03	0.03	0.45	0.03	0.02

In Algorithm 2.2.1, we maintain synthesized global exceptional frequent itemset in array *synFIS*. For the purpose of explanation, Table 2.2.2 has been introduced here. From Table 2.2.2, we find that the synthesized supports of C and F are high, since $supp_s(C, D) \geq \mu$ and $supp_s(F, D) \geq \mu$. Itemset F has been extracted from one out of three databases. Thus, F is low-voted, since $1/3 < \gamma$. Therefore, the itemset F is a global exceptional frequent itemset in D. •

In the following theorem, we determine time complexity of algorithm *IdentifyExPattern* [89] for the purpose of comparing algorithms *IdentifyExPattern* and *ExceptionalFrequentItemsetSynthesis* theoretically.

Theorem 2.2.2. *The algorithm IdentifyExPattern takes $O(n^2 \times N \times log(N))$ time, where N is the number of frequent itemsets extracted from n databases.*

Proof. Please refer to algorithm *IdentifyExPattern.* Step 5 of the algorithm ranks candidate exceptional patterns based on their global supports. Step 4 calculates global support of a candidate exceptional pattern based on the number of databases that support the pattern. Step 1 counts the number databases that support a specific pattern. Thus, step 5 is based on step 4, and step 4 is based on step 1. Step 1 takes $O(n)$ time for a specific pattern. This implies that step 4 takes $O(n \times n)$ time for each candidate exceptional pattern. Thus, step 5 takes $O(n^2 \times N \times \log(N))$ time, and hence the theorem follows. •

From Theorems 2.2.1 and 2.2.2, one could conclude that the algorithm *ExceptionalFrequentItemsetSynthesis* executes faster than algorithm *IdentifyExPatter.* We also compare these two algorithms experimentally in Section 2.2.6.1.

2.2.5 Error calculation

To evaluate the proposed technique for synthesizing global exceptional frequent itemsets, we have measured the amount of error occurred in the experiments. Error of the experiment is relative to the number of transactions (i.e., the size of the database), number of items, and length of a transaction in a database. Let *ANT*, *ALT*, and *ANI* denote the average number of transactions, average length of a transaction and average number of items in the database, respectively. Then, the error of the experiment needs to be expressed along with *ANT*, *ALT*, and *ANI*. The error of the experiment is based on the global exceptional frequent itemsets in *D*. Let $\{X_1, X_2, \ldots, X_m\}$ be set of global exceptional frequent itemsets in *D*. There are several ways one could define the error of an experiment. We have defined following two types of error of an experiment.

1. Average Error (AE)

$$\text{AE}(D, \alpha, \mu, \gamma) = \frac{1}{m} \Sigma_{i=1}^{m} \left| supp_a(X_i, D) - supp_s(X_i, D) \right| \qquad (2.2.8)$$

2. Maximum Error (ME)

$$\text{ME}(D, \alpha, \mu, \gamma) = maximum \left\{ \left| supp_a(X_i, D) - supp_s(X_i, D) \right|, i = 1, 2, \ldots, m \right\} \qquad (2.2.9)$$

Actual support of X_i in *D*, $supp_a(X_i, D)$, is obtained by mining *D* using a traditional data mining technique, for $i = 1, 2, \ldots, m$. Synthesized support of X_i in *D*, $supp_s(X_i, D)$, is obtained by the technique discussed in Section 2.2.3, for $i = 1, 2, \ldots, m$. Then we compute error of synthesizing support of X_i in *D* as $| supp_a(X_i, D) - supp_s(X_i, D) |$, for $i = 1, 2, \ldots, m$.

Example 2.2.2. With reference to Example 2.2.1, the itemset *F* is the only global exceptional frequent itemset present in *D*. Thus, AE(*D*, 0.1, 0.25, 0.4) = ME(*D*, 0.1, 0.25, 0.4) = $|supp_a(F, D) - supp_s(F, D)|$. •

2.2.6 Experiments

We have carried out several experiments to study the effectiveness of our approach. All the experiments have been implemented on a 2.8 GHz Pentium D dual processor with

512 MB of memory using visual C++ (version 6.0) software. The experimental results are presented on both artificial and real databases. We have constructed artificial database *check* to verify that the proposed algorithm works correctly. Each item is represented by an integer number to perform experiments more conveniently. Thus, a transaction in *check* is a collection of integer numbers separated by commas. The database *retail* [34] is obtained from an anonymous Belgian retail supermarket store. We present some characteristics of these databases in Table 2.2.3.

Table 2.2.3. Characteristics of the databases

Database	# transactions (*NT*)	Avg length of a transaction (*ALT*)	Avg frequency of an item (*AFI*)	# items (*NI*)
check(C)	40	3.025000	3.102564	39
retail(R)	88,162	11.305755	99.673800	10000

Each of the above databases has been divided into 10 databases for the purpose of carrying out experiments. These databases are called input databases. The algorithm is based on the frequent itemsets in the input databases. There are many algorithms [13], [39], [66] for mining frequent itemsets in a database. Thus, there exist many implementations [32] of mining frequent itemsets in a database.

Database *check* consists of 40 transactions. The input databases obtained from *check* are given as follows: C_0 = {{1, 4, 9, 31}, {1, 4, 7, 10, 50}, {1, 4, 10, 20, 24}, {1, 4, 10, 23}; C_1 = {{1, 4, 10, 34}, {1, 3, 44}, {1, 2, 3, 10, 20, 44}, {2, 3, 20, 39}}; C_2 = {{2, 3, 20, 44}, {2, 3, 45}, {2, 3, 44, 50}, {2, 3, 20, 44, 50}}; C_3 = {{ 3, 44},{3, 19, 50}, {5, 7, 21}, {5, 8}}; C_4 = {{ 5, 41, 45}, {5, 49}, {5, 7, 21}, {5, 11, 21}}; C_5 = {{6, 41}, {6, 15, 19}, {11, 12, 13}, {11, 21, 49}}; C_6 = {{11, 19}, {21}, {21, 24, 39}, {22, 26, 38}}; C_7 = {{22, 30,31}, {24, 35}, {25, 39, 49}, {26, 41, 46}}; C_8 = { {30, 32, 42}, {32, 49}, {41, 45, 59}, {42, 45}}; C_9 = { {42, 47},{45, 49}, {47, 48, 49}, {49}}. The input databases obtained from *retail* are named as R_i, for i = 0, 1, ..., 9. For the purpose of mining input databases, we have implemented apriori algorithm [13], since it is simple and popular. Some characteristics of these input databases are presented in the Table 2.2.4.

Table 2.2.4. The characteristics of databases obtained from *retail*

Input database	# transactions	Avg length of a transaction	Avg frequency of an item	# items
R_0	9000	11.24389	12.07001	8384
R_1	9000	11.20922	12.26541	8225
R_2	9000	11.33667	14.59657	6990
R_3	9000	11.48978	16.66259	6206
R_4	9000	10.95678	16.03953	6148
R_5	9000	10.85578	16.70977	5847
R_6	9000	11.20011	17.41552	5788
R_7	9000	11.15511	17.34554	5788
R_8	9000	11.99711	18.69032	5777
R_9	7162	11.69199	15.34787	5456

The global exceptional frequent itemsets corresponding to *check* and *retail* are presented in Tables 2.2.5 and 2.2.6, respectively.

Table 2.2.5. Global exceptional frequent itemsets in $\{C_0, C_1, \ldots, C_9\}$ at $\alpha = 0.05$, $\gamma = 0.4$ and $\mu = 0.1$

itemset	ssupp	sources	exceptional sources	itemset	ssupp	sources	exceptional sources
{1}	.17500	C_0, C_1	C_0	{1,10}	.12500	C_0, C_1	C_1
{2}	.17500	C_1, C_2	C_2	{2,3}	.15000	C_1, C_2	C_2
{3}	.20000	C_1, C_2, C_3	C_1, C_2	{2,20}	.12500	C_1, C_2	C_2
{4}	.12500	C_0, C_1	C_0	{2,44}	.12500	C_1, C_2	C_2
{5}	.15000	C_3, C_4	C_4	{3,20}	.10000	C_1, C_2	C_2
{10}	.12500	C_0, C_1	C_0	{3,44}	.15000	C_1, C_2, C_3	C_2, C_3
{11}	.10000	C_4, C_5, C_6	C_5	{4,10}	.10000	C_0, C_1	C_1
{20}	.12500	C_0, C_1, C_2	C_1, C_2	{1,4,10}	.10000	C_0, C_1	C_0, C_1
{44}	.15000	C_1, C_2, C_3	C_1, C_2	{2,3,20}	.10000	C_1, C_2	C_1, C_2
{50}	.10000	C_0, C_2, C_3	C_2	{2,3,44}	.10000	C_1, C_2	C_1, C_2
{1,4}	.12500	C_0, C_1	C_1				

A global exceptional frequent itemset might not be supported with equal degree from each of the source databases. For example, the global exceptional frequent itemset {50} has been extracted from databases C_0, C_2, and C_3. But, the database C_2 reports itemset {50} exceptionally more.

Table 2.2.6. Global exceptional frequent itemsets in $\{R_0, R_1, \ldots, R_9\}$ at $\alpha = 0.02$, $\gamma = 0.4$ and $\mu = 0.1$

itemset	ssupp	sources	exceptional sources
{2,6,9}	0.10228	R_0, R_1	R_0, R_1
{2,9,41}	0.10272	R_0, R_1, R_7	R_0, R_1, R_7
{6,9,41}	0.10662	R_0, R_1, R_3	R_0, R_1, R_3
{8,9,271}	0.10184	R_0	R_0
{9,41,48}	0.10184	R_1	R_1

We observe that some databases do no report any global exceptional frequent itemsets. On the contrary, some other databases are the source of many global exceptional frequent itemsets. In Tables 2.2.7 and 2.2.8, we present the distributions of global exceptional frequent itemsets in $\{C_0, C_1, \ldots, C_9\}$ and $\{R_0, R_1, \ldots, R_9\}$, respectively.

Table 2.2.7. Distribution of global exceptional frequent itemsets in $\{C_0, C_1, \ldots, C_9\}$ at $\alpha = 0.05$, $\gamma = 0.4$ and $\mu = 0.1$

Database	C_0	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	C_9
# global exceptional frequent itemsets	9	18	12	5	2	1	1	0	0	0

In Table 2.2.7, we observe that three out of ten databases do not report any global exceptional frequent itemsets. We have also conducted experiments on synthetic databases. The items in most of the synthetic databases are more or less uniformly distributed. Thus, a set of synthetic databases rarely reports global exceptional frequent itemsets.

Table 2.2.8. Distribution of global exceptional frequent itemsets in $\{R_0, R_1, \ldots, R_9\}$ at $\alpha = 0.02$, $\gamma = 0.4$ and $\mu = 0.1$

Database	R_0	R_1	R_2	R_3	R_4	R_5	R_6	R_7	R_8	R_9
# global exceptional frequent itemsets	4	4	0	0	0	0	0	1	0	0

The distribution of global exceptional frequent itemsets in $\{R_0, R_1, \ldots, R_9\}$ is different from that in $\{C_0, C_1, \ldots, C_9\}$. In Table 2.2.8, we notice that three out of ten databases report global exceptional frequent itemsets. The items in *retail* are more uniformly distributed than that in *check*, since the number of global exceptional frequent items in *retail* is much less than that in *check*.

Also, we have studied the number of global exceptional frequent itemsets in multi-databases at different γs. As we increase γ, we allow more frequent itemsets to be global exceptional. In Figures 2.2.1 and 2.2.2, we study the relationship between γ and the number of global exceptional frequent itemsets in multiple databases.

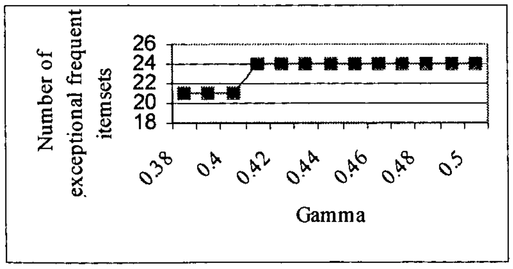


Figure 2.2.1. Number of global exceptional frequent itemsets in $\{C_0, C_1, \ldots, C_9\}$ at $\alpha = 0.05$, and $\mu = 0.1$

We have observed that the number of global exceptional frequent itemsets do not vary much at different γs. In fact, there is only one change in both the graphs of Figures 2.2.1 and 2.2.2.

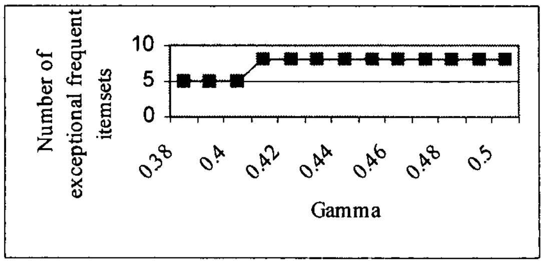


Figure 2.2.2. Number of global exceptional frequent itemsets in $\{R_0, R_1, ..., R_9\}$ at α= 0.02, and μ= 0.1

Also, we have studied the number of global exceptional frequent itemsets in multiple databases at different αs. We present experimental results in Figures 2.2.3 and 2.2.4. The number of global exceptional frequent itemsets in $\{C_0, C_1, ..., C_9\}$ remains fixed at 21 over different αs.

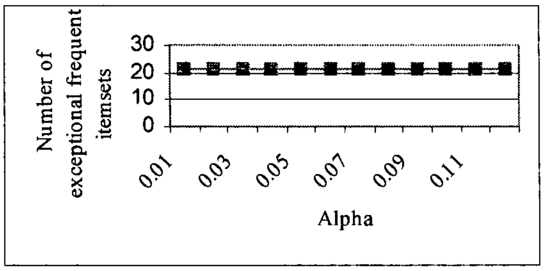


Figure 2.2.3. Number of global exceptional frequent itemsets in $\{C_0, C_1, ..., C_9\}$ at γ= 0.4, and μ= 0.1

But, we find a different trend with respect to the number of global exceptional frequent itemsets in $\{R_0, R_1, ..., R_9\}$. At lower and upper values of α, the number of global exceptional frequent itemsets is 0. Again, we get few global exceptional frequent itemsets for some middle values of α. Thus, there is no fixed relationship between the number of global exceptional frequent itemsets and α.

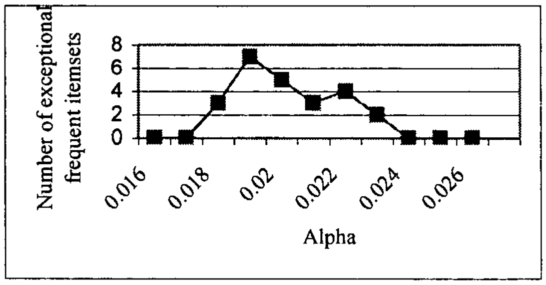


Figure 2.2.4. Number of global exceptional frequent itemsets in $\{R_0, R_1, ..., R_9\}$ at $\gamma = 0.4$, and $\mu = 0.1$

Also, we have calculated the error of the experiments. In Table 2.2.9, we present the error of the experiments at a given value of triplet (α, γ, μ).

Table 2.2.9. Errors of the experiments at a given value of triplet (α, γ, μ)

Experimental databases	α	γ	μ	(AE, Avg *NT*, *ALT*, Avg *NI*)	(ME, Avg *NT*, *ALT*, Avg *NI*)
$C_0, C_1, ..., C_9$	0.05	0.4	0.1	(0, 4, 3.025000, 8.4)	(0, 4, 3.025000, 8.4)
$R_0, R_1, ..., R_9$	0.02	0.4	0.1	(0.08359, 8816.2, 11.305755, 5882.1)	(0.085015, 8816.2, 11.305755, 5882.1)

2.2.6.1 Comparison with the existing algorithm

In this section, we make comparison between algorithms *IdentifyExPattern* [89] and *Exceptional-FrequentItemset-Synthesis* [2] experimentally. We analyze and compare these two algorithms on the basis of experiments conducted on the following two issues: (i) average error versus α, and (ii) synthesizing time versus number of databases.

2.2.6.1.1 *Average error*

We have calculated AEs at different αs to study the relationship between them. Experimental results are presented in Figures 2.2.5 and 2.2.6. We observe that there is no fixed relationship between AE and α.

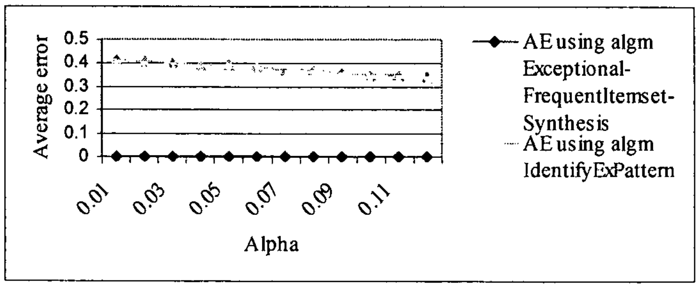


Figure 2.2.5. Average error versus α for *check* at $\gamma = 0.4$, and $\mu = 0.1$

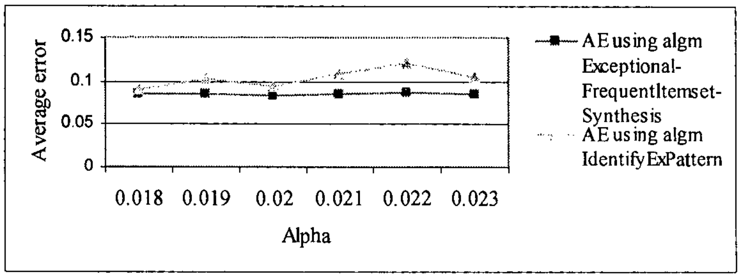


Figure 2.2.6. Average error versus α for *retail* at $\gamma = 0.4$, and $\mu = 0.1$

For both the databases, algorithm *ExceptionalFrequentItemsetSynthesis* performs better than algorithm *IdentifyExPattern.* In database *check*, the global exceptional frequent itemsets are not uniformly distributed. The global exceptional frequent itemsets appear only in few databases, while they remain absent in the remaining databases. Algorithm *ExceptionalFrequentItemsetSynthesis* finds average error 0 at different αs, since the error of synthesizing each global exceptional frequent itemset in $\{C_0, C_1, \ldots, C_9\}$ is 0.

2.2.6.1.2 *Synthesizing time*

Also, we have calculated the time for synthesizing global exceptional frequent itemsets by varying the number of databases. In Figures 2.2.7 and 2.2.8, we show time (in ms.) required to synthesize global exceptional frequent itemsets in multiple databases. In case of the experiment conducted on $\{C_0, C_1, \ldots, C_9\}$, we observe that the synthesizing time does not increase as the number of databases increases. This is due to the fact that the size of each of the databases is very small. In fact, the time required to synthesize global exceptional frequent itemsets in $\{C_0, C_1, \ldots, C_9\}$ is 0 ms., for both the algorithms.

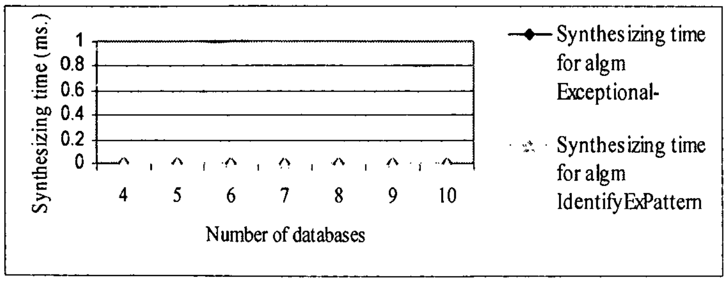


Figure 2.2.7. Synthesizing time versus number of databases obtained from *check* at $\gamma = 0.4$, and $\mu = 0.1$

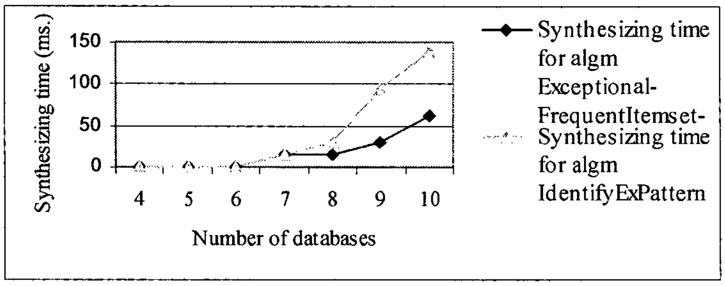


Figure 2.2.8. Synthesizing time versus number of databases obtained from *retail* at $\alpha = 0.02$, $\gamma = 0.4$, and $\mu = 0.1$

Considering the results presented in Figures 2.2.7 and 2.2.8, one could conclude that algorithm *Exceptional-FrequentItemset-Synthesis* executes faster than algorithm *IdentifyExPattern.* Also, this observation matches with the theoretical results presented. In general, the time for synthesizing global exceptional frequent itemsets either remains the same or, increases as the number of databases increases.

2.2.7 Conclusion

Synthesis of global exceptional patterns is an important component of a multi-database mining system. Many corporate decisions of a multi-branch company would depend on global exceptional patterns in branch databases. Though previous work discussed issues related to this problem, we believe that we have presented a better analysis.

We have identified the short-comings of the existing concepts and the algorithm to identify global exceptional patterns. We have proposed a definition of a global exceptional frequent itemset. Also, we have introduced the notion of exceptional sources for a global exceptional frequent itemset. The proposed algorithm identifies global exceptional frequent itemsets and their exceptional sources in multiple databases. We have also compared our algorithm with the èxisting algorithm. Our algorithm performs better than the existing algorithm on the following issues: (i) error of the experiment, and (ii) execution time. We have observed that our algorithm executes faster than the existing algorithm when the number of databases increases. Also, we have shown theoretically that our algorithm executes faster than the existing algorithm. The solution presented here is simple and effective in synthesizing global exceptional frequent itemsets in multiple databases.

Chapter 2.3

Synthesizing heavy association rules from different real data sources

Improved communication technology has been a major influential factor to rapid industrial growth and business activities. Also, many countries across the globe are adopting slowly a liberal economic policy. Due to the influence of a number of such factors, many countries are experiencing rapid economic growth. As a result, the number of companies is increasing over time. Many large companies have multiple branches. They operate from different branches located at different places. Some of these branches are fully operational and collect transactional data continuously. Consider the shopping malls owned by a company. These malls are open at least 12 hours a day. All the transactions made in a mall are stored locally. Thus, the company possesses multiple databases. Most of the pieces of previous data mining work are based on a single database. Thus, it is necessary to study data mining on multiple databases.

Many corporate decisions could be taken effectively by incorporating knowledge inherent in data across the branches. But, the effective management of multiple large databases becomes a challenging issue. It creates not only opportunities but also risks. The risks might involve significant amount investment on hardware and software to deal with a large volume of data. Our objective is to provide good quality of knowledge by minimizing the risks. The first question comes to our mind whether a traditional data mining technique could deal with multiple large databases. To apply a traditional data mining technique we need to amass all the databases together. A single computer might take unreasonable amount of time to process the entire database. Sometimes, it might not be feasible to carry out the data mining task using a single computer. Another solution to this problem would be to employ parallel machines. It might require high investment on hardware and software. One needs to make a cost-benefit analysis before implementing such a decision. In many situations, it might not be an acceptable solution to the management of the company. Moreover, it might be difficult to find regional patterns when a traditional data mining technique is applied on the entire database. Thus, the traditional data mining techniques are not suitable in this situation. So, it is a different problem. Hence, it is required to be dealt with in a different way. In this situation, we would employ the model of local pattern analysis [91] to deal with multiple large databases. Under this model, the branches are required to forward their local patterns instead of original databases to the central office for synthesis of global patterns.

Association rule mining has received a lot of attention to KDD community. An association rule becomes more interesting if it possesses higher support and higher confidence. In this chapter, we present the notion of heavy association rule. Heavy association rules are sometimes more useful than high-frequent association rules [81]. Many corporate decisions could be influenced by heavy association rules. Thus, it is important to mine heavy association rules in multiple databases. It could be difficult to extract heavy association rules in the union of all branch databases by employing a traditional data mining technique. Therefore, we synthesize heavy association rules from the association rules in local databases. We present an algorithm to synthesize heavy association rules from local association rules. We have extended the algorithm to notify whether a heavy association rule is high-frequent.

Also, we introduce the notion of exceptional association rule in multiple databases. We have also extended the algorithm to notify whether a heavy association rule is exceptional. Thus, our extended algorithm not only synthesizes heavy association rules, but also it notifies whether a heavy association rule is high-frequent or exceptional in multiple databases.

The rest of the chapter is organized as follows. We state the problem in Section 2.3.2. We discuss work related to the problem in Section 2.3.3. In Section 2.3.4, we present an extended model of synthesizing global patterns from local patterns in different databases. In Section 2.3.5, we discuss a method of synthesizing an association rule in multiple databases. We present an algorithm for synthesizing heavy association rules in multiple databases. The algorithm also reports whether a heavy association rule is high-frequent or exceptional in multiple databases. We have defined error of an experiment in Section 2.3.6. The experimental results on three real databases are presented in Section 2.3.7.

2.3.2 Problem statement

Consider a large company that transacts from n branches. Let D_i be the transactional database corresponding to the i-th branch of the multi-branch company, for $i = 1, 2, \ldots, n$. Also, let D be the union of all branch databases. We present an algorithm for synthesizing heavy association rules in D. The algorithm also notifies the high-frequency and exceptionality statuses of heavy association rules in D. Heavy association rules, high-frequent association rules and exceptional association rules are specific types of association rules. It might be required to discuss some other concepts before we define them.

Association rule mining is based on support (supp)-confidence (conf) framework established by Agrawal et al. [11]. Let I be set of items in D. An association rule r has been expressed symbolically as $c \rightarrow d$, where $c = \{c_1, c_2, \ldots, c_p\}$, and $d = \{d_1, d_2, \ldots, d_q\}$; $c_i, d_j \in I$, for $i = 1, 2, \ldots, p$, and $j = 1, 2, \ldots, q$. It expresses an association between the itemsets c and d, called the antecedent and consequent of r, respectively. The meaning attached to this implication could be expressed as follows: If the items in the itemset c are purchased by a customer then the items in the itemset d are likely to be purchased by the same customer at the same time. The interestingness of an association rule could be expressed by its support and confidence. Let E be a Boolean expression of items in database DB. Support of E in DB could be defined as the fraction of transactions in DB

such that the Boolean expression E is true for each of these transactions. We denote support of E in DB as $supp_a(E, DB)$. Then the support and confidence of association rule r could be expressed as follows: $supp_a(r, DB) = supp_a(c \cap d, DB)$, and $conf_a(r, DB) = supp_a(c \cap d, DB) / supp_a(c, DB)$. Later, we shall be dealing with synthesized support and synthesized confidence of an association rule. Thus, it is required to differentiate between actual support / confidence and their synthesized versions. The subscript a in the notation of support / confidence refers to actual support / confidence of an association rule. On the other hand, the subscript s in the notation of support / confidence refers to synthesized support / confidence of an association rule. We shall discuss how to get synthesized support and confidence of an association rule later. An association rule r in database DB is *interesting* if $supp_a(r, DB) \geq$ *minimum support* (α), and $conf_a(r, DB) \geq$ *minimum confidence* (β), for $i = 1, 2, \ldots, n$. The values of α and β are user-defined. The collection of association rules extracted from a database at the given α and β is called a *rulebase*. Let RB_i and SB_i be the rulebase and suggested rulebase corresponding to database D_i, respectively, for $i = 1, 2, \ldots, n$. An association rule $r \in RB_i$, if $supp_a(r, D_i) \geq \alpha$, and $conf_a(r, D_i) \geq \beta$, for $i = 1, 2, \ldots, n$. An association rule $r \in SB_i$, if $supp_a(r, D_i) \geq \alpha$, and $conf_a(r, D_i) < \beta$, for $i = 1, 2, \ldots, n$. There is a tendency of a suggested association rule in a database to become an association rule in another database. Apart from the association rules, we also consider the suggested association rules for synthesizing heavy association rules in D. The reasons for considering suggested association rules are given as follows. Firstly, we could synthesize support and confidence of an association rule in D more accurately. Secondly, we could synthesize high-frequent association rules in D more accurately. Thirdly, the experimental results (as reported in Table 2.3.5) have shown that the number of suggested association rules could be significant for some databases. In general, the accuracy of synthesizing an association rule increases as the number of extractions of the association rule increases. Thus, we consider suggested association rules also for synthesizing heavy association rules in D. In addition, the number of transactions in a database would be required for synthesizing an association rule. We define *size* of database DB as the number of transactions in DB, denoted by $size(DB)$. In the following, we define the concept of heavy association rule in multiple databases. First, we define a heavy association rule in a single database. Afterwards, we shall define a heavy association rule in multiple databases.

Definition 2.3.1. An association rule r in database DB is heavy if $supp_a(r, DB) \geq \mu$, and $conf_a(r, DB) \geq \nu$, where μ ($> \alpha$) and ν ($> \beta$) are the user defined thresholds of high-support and high-confidence for identifying heavy association rules in DB, respectively. •

If an association rule is heavy in a local database then it might not be heavy in D. An association rule in D might have different statuses in different local databases. For example, it might be a heavy association rule, or an association rule, or a suggested association rule, or absent in a local database. Thus, we need to synthesize an association rule for determining its overall status in D. The method of synthesizing an association rule is discussed in Section 2.3.5. After synthesizing an association rule, we get synthesized support and synthesized confidence of the association rule in D. Let $supp_s(r, DB)$ and $conf_s(r, DB)$ denote synthesized support and synthesized confidence of association rule r in DB, respectively. Now, we define a heavy association rule in D as follows.

Definition 2.3.2. Let D be the union of local databases under consideration. An association rule r in D is heavy if $supp_s(r, D) \geq \mu$, and $conf_s(r, D) \geq \nu$, where μ and ν are the user-defined thresholds of high-support and high-confidence for identifying heavy association rules in D, respectively. •

Apart from synthesized support and synthesized confidence of an association rule, the frequency of an association rule is an important issue in multi-database mining. We define *frequency* of an association rule as the number of extractions of the association rule from different databases. If an association rule is extracted from k out of n databases then the frequency of the association rule is k, for $0 \le k \le n$. An association rule may be high-frequent or, low-frequent, or neither high-frequent nor low-frequent in multiple databases. We could arrive in such a conclusion only if we have user-defined thresholds of low-frequency (γ_1), and high-frequency (γ_2) of an association rule, for $0 < \gamma_1 \le \gamma_2 \le 1$. A low-frequent association rule is extracted from less than $n \times \gamma_1$ databases. On the other hand, a high-frequent association rule is extracted from at least $n \times \gamma_2$ databases. In multi-database mining using local pattern analysis, we define a high-frequent association rule and a low-frequent association rule as follows:

Definition 2.3.3. Let an association rule be extracted from k out of n databases. Then the association rule is low-frequent if $k < n \times \gamma_1$, where γ_1 is the user-defined threshold of low-frequency. •

Definition 2.3.4. Let an association rule be extracted from k out of n databases. Then the association rule is high-frequent if $k \ge n \times \gamma_2$, where γ_2 is the user defined-threshold of high-frequency. •

While synthesizing heavy association rules in multiple databases, it may be worth noting the other attributes of a synthesized association rule. For example, high-frequency, low-frequency, and exceptionality are interesting as well as important attributes of a synthesized association rule. We have already defined high-frequent association rule and low-frequent association rule in multiple databases. We define an exceptional association rule in multiple databases as follows.

Definition 2.3.5. A heavy association rule in multiple databases is exceptional if it is low-frequent. •

It may be worth contrasting between a heavy association rule, a high-frequent association rule and an exceptional association rule in multiple databases.

- An exceptional association rule is also a heavy association rule.
- A high-frequent association rule is not an exceptional association rule, and vice versa.

- A high-frequent association rule is not necessarily be a heavy association rule.
- There may exist heavy association rules that are neither high-frequent nor exceptional.

The proposed problem could be stated as follows:

Let there are n distinct databases D_1, D_2, ..., D_n. Let RB_i and SB_i be the set of association rules and suggested association rules in D_i, respectively, for $i = 1, 2, ..., n$. Synthesize heavy association rules in the union of all databases (D) based on RB_i and SB_i, for $i = 1, 2, ..., n$. Also, notify whether each heavy association rule is high-frequent or exceptional in D.

2.3.3 Related work

Association rule mining finds interesting association between two itemsets in a database. The notion of association rule is introduced by Agrawal et al. [11]. The authors have proposed an algorithm to mine frequent itemsets in a database. Many algorithms have been reported to extract association rules in a database. In the following, we mention a few interesting algorithms for extracting association rules in a database. Agrawal and Srikant [13] have proposed apriori algorithm that uses breadth-first search strategy to count the supports of itemsets. The algorithm uses an improved candidate generation function, which exploits the downward closure property of support and makes it more efficient than earlier algorithm. Han et al. [39] have proposed data mining method FP-growth (frequent pattern growth) which uses an extended prefix-tree (FP-tree) structure to store the database in a compressed form. FP-growth adopts a divide-and-conquer approach to decompose both the mining tasks and databases. It uses a pattern fragment growth method to avoid the costly process of candidate generation and testing. Savasere et al. [66] have introduced partition algorithm. The database is scanned only twice. In the first scan the database is partitioned and in each partition support is counted. Then the counts are merged to generate potential frequent itemsets. In the second scan, the potential frequent itemsets are counted to find the actual frequent itemsets.

In the context of pattern synthesis, Viswanath et al. [78] have proposed a novel pattern synthesis method, called partition based pattern synthesis, which can generate an artificial training set of exponential order when compared with that of the given original training set.

Multi-database mining has been recently recognized as an important research topic in KDD community. In the following, we mention a few important contributions in multi-database mining. Liu et al. [55] have proposed multi-database mining technique that searches only the relevant databases. Otherwise, the mining process could be lengthy, aimless and ineffective. A measure of relevance is thus proposed for mining tasks with an objective to find patterns or regularities about certain attributes. Wu and Zhang [81] have proposed a weighting model for synthesizing high-frequent association rules from different databases. Zhang et al. [89] have proposed an algorithm to identify global exceptional frequent itemsets in multiple databases. Zhang [88], Zhang et al. [93] studied various strategies for mining different databases. Wu et at. [83] have proposed a database clustering technique for multi-database mining. Yin and Han [86] have proposed a new strategy for relational heterogeneous database classification. Zhang and Zaki [92] have edited a book on multi-database mining. Aronis et al. [18] introduced a system, called WoRLD, that uses spreading activation to enable inductive learning from multiple tables in multiple databases spread across the network.

Existing parallel mining techniques [12], [26], [28] could also be used to deal with multi-databases.

In the context of other applications of data mining, Hong and Weiss [42] have examined a few successful application areas and their technical challenges to show how the demand for data mining of massive data warehouses has fuelled advances in automated predictive methods.

2.3.4 An extended model of synthesizing global patterns from local patterns in different databases

Many multi-branch companies deal with multiple databases. As the number of such companies increases, we need to prepare ourselves to develop various applications based on patterns in multiple databases. There are following types of patterns in multiple databases: local pattern, global patterns, and patterns that are neither local nor global. A pattern based on a branch database is called a *local pattern*. On the other hand, a *global pattern* is based on all the databases under consideration. Zhang et al. [91] designed a local pattern analysis for synthesizing global patterns in multiple databases. We present here an extension to this model and the extended model [5] is shown in Figure 2.3.1. The extended model has a set of interfaces and a set of layers. Each interface is a set of operations that produces database(s) (or, knowledge) based on the database(s) at the next lower layer. There are four interfaces of the extended model for synthesizing global patterns from local patterns in different databases. The functions of the interfaces are described below.

Interface 2/1 applies different operations on data at the lowest layer. By applying these operations, we get a processed database from a local (original) database. These operations are performed on each branch database. Interface 3/2 applies a filtering algorithm on each processed database to separate relevant data from outlier data [20]. In particular, if we are interested in studying the durable items then the transactions containing only non-durable items could be treated as outlier transactions. Different interesting criteria could be set to filter data at this stage. Also, it loads data into the respective data warehouse. Interface 4/3 mines local patterns in each local data warehouse. There are two types of local patterns: local patterns and suggested local patterns. A suggested local pattern is close but fails to satisfy the requisite interestingness criteria. The reasons for considering suggested patterns are given as follows: Firstly, we could synthesize patterns more accurately. Secondly, due to the stochastic nature of the transactions, the number of suggested patterns could be significant in some databases. Thirdly, there is a tendency that a suggested pattern of one database to become a local pattern in another database. Thus, the correctness of synthesizing global patterns would increase as the number of local patterns increases. Consider a multi-branch company having *n* databases. Let LPB_i and SPB_i be the local pattern base and suggested local pattern base for the *i*-th branch, respectively, for $i = 1, 2, \ldots, n$. Interface 5/4 synthesizes global patterns, or analyses local patterns to meet real life challenges.

At the lowest layer, all the local databases are kept. We need to process these databases as they may not be at the right state for the mining task: Various data preparation techniques [65] like data cleaning, data transformation, data integration, and data reduction are applied to data in the local databases. We get the processed database PD_i corresponding to original database D_i, for $i = 1, 2, \ldots, n$. Then, we retain all the data that are relevant to the data mining applications. Using a relevance analysis, one could detect outlier data [51] from processed database. A relevance analysis is dependent on the context, and varies from one application to another application. Let OD_i be the outlier database corresponding to the *i*-th branch, for $i = 1, 2, \ldots, n$. Sometimes these databases are also used in some other applications. After removing outlier data from the processed database we get desired data warehouse, and the data in a data warehouse become ready for data mining task. Let W_i be the data warehouse corresponding to the *i*-th branch, for $i = 1, 2, \ldots, n$. Local patterns for the *i*-th branch are extracted from W_i, for $i = 1, 2, \ldots, n$. Finally, the local patterns are forwarded to the central office for synthesizing global patterns, or analyzing local patterns. Many data mining applications could be developed based on the local patterns in different databases. Figure 2.3.1 illustrates a model of synthesizing global patterns from local patterns in different databases.

In particular, if we are interested in synthesizing global frequent itemsets then a frequent itemset might not get extracted from all the databases under consideration. It might be required to estimate the support of a frequent itemset in a database that fails to report it. Thus, a global frequent itemset synthesized from local frequent itemsets is approximate in nature. If any one of the local databases is too large to apply a traditional data mining technique then this model would fail. In this situation, one could apply an appropriate sampling technique to reduce the size of a local database. Otherwise, the database could be partitioned into sub-databases. As a result, the error of data analysis would increase.

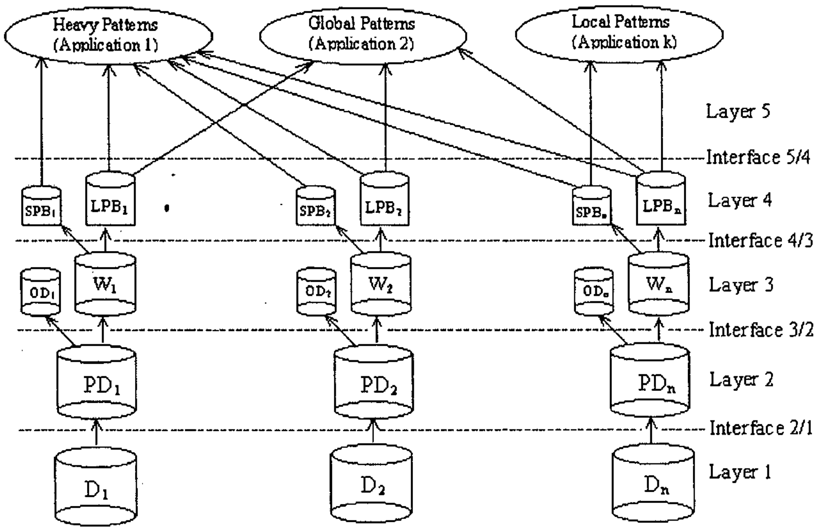


Figure 2.3.1. A model of synthesizing global patterns from local patterns in different databases

Though the above model introduces many layers and interfaces for synthesizing global patterns, but in a real life application, many of these layers and interfaces might be absent.

2.3.5 Synthesizing an association rule

Our technique of synthesizing heavy association rules is suitable for real databases, where the trend of the customers' behaviour exhibited in one database is usually present in other databases. In particular, a frequent itemset in one database is usually present in some transactions of other databases even if it does not get extracted. Our estimation procedure captures such trend and estimates the support of a missing association rule in a database. Let $E_1(r, DB)$ be the amount of error in estimating support of a missing association rule r in database DB. Also, let $E_2(r, DB)$ be the amount of error in assuming support as 0 for the missing association rule in *DB*. Then, $E_1(r, DB)$ is usually less than $E_2(r, DB)$. The estimated support and confidence of a missing association rule usually reduce the error of synthesizing heavy association rules in different databases. Thus, we would like to estimate the support and confidence of a missing association rule rather than missing its presence. If an association rule fails to get extracted from database *DB*, then we assume that *DB* contributes some amount of support and confidence for the association rule. The support and confidence of an association rule *r* in database *DB* satisfy the following inequality: $0 \leq supp_a(r, DB) \leq conf_a(r, DB) \leq 1$ (2.3.1)

At a given $\alpha = \alpha_0$, we observe that the confidence of an association rule *r* varies over the interval $[\alpha_0, 1]$.

Example 2.3.1. Let $\alpha = 0.333$. Assume that database D_1 contains the following transactions: $\{a1, b1, c1\}$, $\{a1, b1, c1\}$, $\{b2, c2\}$, $\{a2, b3, c3\}$, $\{a3, b4\}$ and $\{c4\}$. The support and confidence of association rule *r*: $\{a1\} \rightarrow \{b1\}$ in D_1 are 0.333 and 1.0 (highest) respectively. Assume that database D_2 contains the following transactions: $\{a1, b1, c1\}$, $\{a1, b1\}$, $\{a1, c1\}$, $\{a1\}$, $\{a1, b2\}$ and $\{a1, b3\}$. The support and confidence of *r* in D_2 are 0.333 and 0.333 (lowest) respectively. •

As the support of an association rule is the lower bound of its confidence, the confidence goes up as support increases. The support of an association rule is distributed over [0, 1]. If an association rule is not extracted from a database, then the support falls in $[0, \alpha)$, since the suggested association rules are also considered for synthesizing association rules. We would be interested in estimating the support of such rules. Assume that the association rule *r*: $\{c\} \rightarrow \{d\}$ has been extracted from *m* databases, for $1 \leq m \leq n$. Without loss of generality, we assume that the association rule *r* has been reported from the first *m* databases. We shall use the average behaviour of the customers of the first *m* branches to estimate the average behaviour of the customers in remaining branches. Let $D_{i,j}$ denote the union of databases $D_i, D_{i+1}, \ldots, D_j$, for $1 \leq i \leq j \leq n$. Then, $supp_a(\{c, d\}, D_{1,m})$ could be viewed as the average behaviour of customers of the first *m* branches for purchasing

2.3.5.1 Algorithm design

In this section, we present an algorithm for synthesizing heavy association rules in *D*. The algorithm also notifies whether a heavy association rule is high-frequent or exceptional in *D*. Let *N* and *M* be the number of association rules and the number of suggested association rules in different local databases, respectively. The association rules and suggested association rules are kept in arrays *RB* and *SB*, respectively. A rule in a local database could be described by following attributes: *ant*, *con*, *did*, *supp* and *conf*. The attributes *ant*, *con*, *did*, *supp* and *conf* represent antecedent, consequent, database identification, support, and confidence of an association rule, respectively. An attribute *x* of *i*-th association rule of *RB* could be accessed using notation *RB*(*i*).*x*, for $i = 1, 2, \ldots, |RB|$. All the synthesized rules are kept in array *SR*. A synthesized rule could be described by following attributes: *ant*, *con*, *did*, *ssupp* and *sconf*. The attributes *ssupp* and *sconf* represent synthesized support and synthesized confidence of a synthesized association rule, respectively. In the context of the work presented here, an association rule in *D* has the following additional attributes: *heavy*, *highFreq*, *lowFreq* and *except*. The attributes *heavy*, *highFreq*, *lowFreq* and *except* are used to indicate whether an association rule is heavy, high-frequent, low-frequent and exceptional in *D*, respectively. An attribute *y* of *i*-th synthesized association rule of *SR* could be accessed using notation *SR*(*i*).*y*, for $i = 1, 2, \ldots, |SR|$. An algorithm for synthesizing heavy association rules is presented below:

Algorithm 2.3.1. Synthesize heavy association rules in *D*. Also, indicate whether a heavy association rule is high-frequent or exceptional in *D*.

```
procedure Association-Rule-Synthesis ( n, RB, SB, $\mu$, $\nu$, size, $\gamma_1$, $\gamma_2$)
Inputs:
n: number of databases
RB: array of association rules
SB: array of suggested association rules
$\mu$: threshold of high-support for determining heavy association rules
```

ν: threshold of high-confidence for determining heavy association rules

size: array of total number of transactions in different databases

γ_1: threshold of low-frequency for determining low-frequent association rules

γ_2: threshold of high-frequency for determining high-frequent association rules

Outputs:

Heavy association rules along with their high-frequency and exceptionality statuses

```
01: copy rules of $RB$ and $SB$ into array $R$;
02: sort rules of $R$ based on attributes $ant$ and $con$ of a rule;
03: calculate total number of transactions in different databases into $totalTrans$;
04: let $nSynRules = 1$;
05: let $curPos = 1$;
06: while ( $curPos \leq |R|$ ) do
07:     calculate the number of occurrences of current rule $R(curPos)$ to $nExtractions$;
08:     let $SR(nSynRules).highFreq$ = false;
09:     if $((nExtractions / n) \geq \gamma_2)$ then
10:         $SR(nSynRules).highFreq$ = true;
11:     end if
12:     let $SR(nSynRules).lowFreq$ = false;
13:     if $((nExtractions / n) < \gamma_1)$ then
14:         $SR(nSynRules).lowFreq$ = true;
15:     end if
16:     calculate $supp_s(R(curPos), D)$ using formula (2.3.5);
17:     calculate $conf_s(R(curPos), D)$ using formula (2.3.8);
18:     let $SR(nSynRules).heavy$ = false;
19:     if $((supp_s(SR(nSynRules), D) \geq \mu)$ and $(conf_s(SR(nSynRules), D) \geq \nu))$ then
20:         $SR(nSynRules).heavy$ = true;
21:     end if
22:     let $SR(nSynRules).except$ = false;
```

```
23:   if ((SR(nSynRules) is low-frequent) and (SR(nSynRules) is heavy)) then
24:     SR(nSynRules).except = true;
25:   end if
26:   update index curPos for processing the next association rule;
27:   increase index nSynRules by 1;
28: end while
29: for each synthesized association rule $\tau$ in SR do
30:   if $\tau$ is heavy then
31:     display $\tau$ along with its high-frequency and exceptional statuses;
32:   end if
33: end for
end procedure
```

The above algorithm works as follows. The association rules and suggested association rules are copied into array *R*. All the rules in *R* are sorted on the pair of attributes *ant* and *con*, so that the same rule extracted from different databases becomes consecutive. Thus, it would help synthesizing one rule at a time. The synthesizing process is kept in the while-loop at line 6. Based on the number of extractions of a rule, we could determine its high-frequency and low-frequency statuses. Number of extractions of current association rule has been determined at line 7. The high-frequency status of current association rule is determined using lines 8-11. Also, the low-frequency status of current association rule is determined using lines 12-15. We synthesize support and confidence of current association rule based on formula (2.3.5) and (2.3.8), respectively. Once the synthesized support and synthesized confidence are calculated, one could identify the heavy and exceptional statues of current association rule. The heavy status of current association rule is determined using lines 18-21. Also, the exceptional status of current association rule is determined using lines 22-25. At line 26, one determines the next association rule in *R* for the synthesizing process. Heavy association rules are displayed along with their high-frequent and exceptionality statuses using lines 29 - 33. The shaded regions have been added to report the high-frequency and exceptionality statuses of heavy association rules.

Theorem 2.3.1. *The time complexity of procedure Association-Rule-Synthesis is maximum*{ $O((M + N) \times log(M + N))$, $O(n \times (M + N))$}, *where N and M are the number of association rules and the number of suggested association rules extracted from n databases.*

Proof. The lines 1 and 2 take time in $O(M + N)$ and $O((M + N) \times \log(M + N))$ respectively, since there are $M + N$ rules in different local databases. The while-loop at line 6 repeats maximum $M + N$ times. Line 7 takes $O(n)$ time, since each rule is extracted maximum n number of times. Lines 8-15 take $O(1)$ time. Using formula (2.3.2), one could calculate the average behaviour of customers of the first m databases in $O(n)$ time. Also, each of lines 16 and 17 takes $O(n)$ time. Lines 18-25 take $O(1)$ time. Line 26 could be executed during execution of line 7. Thus, the time complexity of while-loop 6-28 is $O(n \times (M + N))$. The time complexity of lines 29-33 is $O(M + N)$, since the number of synthesized association rules is less than or equal to $M + N$. Thus, the time complexity of procedure *Association-Rule-Synthesis* is *maximum*{ $O((M + N) \times \log(M + N))$, $O(n \times (M + N))$, $O(M + N)$} = *maximum*{ $O((M + N) \times \log(M + N))$, $O(n \times (M + N))$}. •

Wu and Zhang [81] have proposed an algorithm for synthesizing high-frequent association rules in different databases. This algorithm is based on the weights of the different databases. Again, the weight of a database would depend on the association rules extracted from the database. The proposed algorithm executes in $O(n^4 \times maxNosRules \times totalRules^2)$ time, where n, *maxNosRules*, and *totalRules* are the number of data sources, the maximum among the numbers of association rules extracted from different databases, and the total number of association rules in different databases, respectively. Algorithm *Association-Rule-Synthesis* could synthesize heavy association rules, high-frequency association rules, and exceptional association rules in *maximum* { $O(totalRules \times \log(totalRules))$, $O(n \times totalRules)$} time. Thus, the algorithm takes much less time than the existing algorithm that synthesizes only high-frequent association rules. Moreover, our algorithm is simple and direct in nature. We take an example to illustrate the our algorithm.

Example 2.3.2. Let D_1, D_2 and D_3 be three databases of sizes 4000 transactions, 3290 transactions, and 10200 transactions, respectively. Let D be the union of the databases D_1, D_2, and D_3. Assume that $\alpha = 0.2$, $\beta = 0.3$, $\gamma_1 = 0.4$, $\gamma_2 = 0.7$, $\mu = 0.3$ and $\nu = 0.4$. The following association rules have been extracted from the given databases. r_1: $\{H\} \rightarrow \{C, G\}$, r_2: $\{C\} \rightarrow \{G\}$, r_3: $\{G\} \rightarrow \{F\}$, r$_4$: $\{H\} \rightarrow \{E\}$, r_5: $\{A\} \rightarrow \{B\}$. The rulebases are given as follows: $RB_1 = \{r_1, r_2\}$, $SB_1 = \{r_3\}$; $RB_2 = \{r_4\}$, $SB_2 = \{r_1\}$; $RB_3 = \{r_1, r_5\}$, $SB_3 = \{r_2\}$. The supports and confidences of the association rules are given as follows. $supp_a(r_1, D_1) = 0.22$, $conf_a(r_1, D_1) = 0.55$; $supp_a(r_1, D_2) = 0.25$, $conf_a(r_1, D_2) = 0.29$; $supp_a(r_1, D_3) = 0.20$, $conf_a(r_1, D_3) = 0.52$; $supp_a(r_2, D_1) = 0.69$, $conf_a(r_2, D_1) = 0.82$; $supp_a(r_2, D_3) = 0.23$, $conf_a(r_2, D_3) = 0.28$; $supp_a(r_3, D_1) = 0.22$, $conf_a(r_3, D_1) = 0.29$; $supp_a(r_4, D_2) = 0.40$, $conf_a(r_4, D_2) = 0.45$; $supp_a(r_5, D_3) = 0.86$, $conf_a(r_5, D_3) = 0.92$. Also, let $supp_a(\{A\}, D_3) = 0.90$, $supp_a(\{C\}, D_1) = 0.80$, $supp_a(\{C\}, D_3) = 0.40$, $supp_a(\{G\}, D_1) = 0.29$, $supp_a(\{H\}, D_1) = 0.31$, $supp_a(\{H\}, D_2) = 0.33$, and $supp_a(\{H\}, D_3) = 0.50$.

Table 2.3.1. Heavy association rules in the union of databases given in Example 2.3.2

r: $ant \rightarrow con$	ant	con	$supp_s(r, D)$	$conf_s(r, D)$	heavy	highFreq	except
r_2	C	G	0.305466	0.664523	true	false	false
r_5	A	B	0.573235	0.899829	true	false	true

The association rules r_2 and r_5 have synthesized support greater than or equal to 0.3 and synthesized confidence greater than or equal to 0.4. So, r_2 and r_5 are heavy association rules in D. The association rule r_5 is exceptional, since it is heavy and low-frequent. But, the association rule r_2 is neither high-frequent nor exceptional. Though the association rule r_1 is high-frequent but it is not heavy, since $supp_s(r_1, D) = 0.213980$ and $conf_s(r_1, D) = 0.483589$. •

2.3.5.2 Finding expected lower bound of the number of suggested association rules

A frequent itemset X might fail to generate an association rule under some conditions, for $|X| \geq 2$. Let $\{c, d\}$ be a frequent itemset in database D. The association rule r: $\{c\} \rightarrow \{d\}$ fails to get extracted from D if the following conditions are satisfied.

$$supp_a(\{c, d\}, D) \geq \alpha, \text{ and } \frac{supp_a(\{c, d\}, D)}{supp_a(\{c\}, D)} < \beta \tag{2.3.9}$$

i.e., $\alpha \leq supp_a(\{c, d\}, D) < \beta \times supp_a(\{c\}, D)$. The frequent itemset $\{c, d\}$ fails to produce an association rule if $\beta \times supp_a(\{c\}, D) - supp_a(\{c, d\}, D) > 0$. Consider a large database containing items c and d. We assume that $supp_a(\{c\}, D) = 0.07$ and $supp_a(\{c, d\}, D) = 0.02$. Let us consider the equation $f(\beta) = 0.07 \times \beta - 0.02$. Now, $f(\beta) = 0$ implies $\beta = 0.28571$. Thus, the association rule r gets extracted if $\beta > 0.28571$.

The following theorem provides a lower bound of the expected number of suggested association rules in a database at given α and β.

Theorem 2.3.2. *Let m be the number of frequent itemsets of size greater than or equal to 2 in database DB. Let X and Y be any two disjoint frequent itemsets in DB. Assume that $supp_a(X, DB) \geq supp_a(Y, DB)$, if $|X| < |Y|$. Then, the expected lower bound of the number of suggested association rules in DB is given by*

$$2 \times m \times \left[\frac{(\beta^2 - 3 \times \beta + 3) \times \alpha^2 + \beta \times (\beta - 3) \times \alpha + \beta^2}{3 \times (1 - \alpha)} \right] \tag{2.3.10}$$

where, α and β are use-defined minimum support and minimum confidence, respectively.

Proof. Let I be a frequent itemset in DB, for $|I| \geq 2$. Also, let the itemsets J and K be proper subsets of I, such that $\{J, K\}$ forms a partition [53] of I. The itemset I fails to generate any of the two association rules $J \rightarrow K$ and $K \rightarrow J$, if

$$\frac{supp_a(I, DB)}{supp_a(J, DB)} < \beta, \text{ and } \frac{supp_a(I, DB)}{supp_a(K, DB)} < \beta$$

i.e., if $supp_a(I, DB) < \beta \times minimum\ \{\ supp_a(J, DB),\ supp_a(K, DB)\ \}$ (2.3.11)

Without loss of generality, let $|J| > |K|$. Assume that $supp_a(J, DB) \leq supp_a(K, DB)$, for $|J| = |K|$. Then, $supp_a(J, DB) \leq supp_a(K, DB)$, by the assumption of the theorem.

Thus, the condition (2.3.11) is true if $supp_a(I, DB) < \beta \times supp_a(J, DB)$

or, $\beta \times supp_a(J, DB) - supp_a(I, DB) > 0$ (2.3.12)

Let $y = supp_a(I, DB)$ and $z = minimum\ \{\ supp_a(J, DB): J \subset I\ \}$. Consider the function $f(z, y) = \beta \times z - y$, such that $1 \geq z \geq y \geq \alpha$. Let $A_1 = \{\ (z, y): f(z, y) > 0$ and $1 \geq z \geq y \geq \alpha\ \}$, and $A_2 = \{\ (z, y): 1 \geq z \geq y \geq \alpha\ \}$. If $(z, y) \in A_1$ then the corresponding itemset I does not generate an association rule in DB.

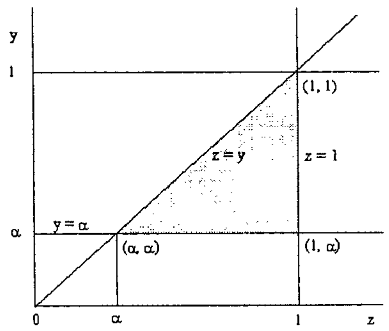


Figure 2.3.2. Region A_2 (shaded area)

$$A_1 = \int_{z=\alpha}^{1} \int_{y=\alpha}^{\beta\times z} (\beta \times z - y)\, dydz \quad (2.3.13)$$

$$= \frac{(1-\alpha)}{6}\left(\left(\beta^2 - 3\times\beta+3\right)\times\alpha^2 + \beta\times(\beta-3)\times\alpha+\beta^2\right) \quad (2.3.14)$$

$$A_2 = (1-\alpha)^2/2 \quad (2.3.15)$$

Let E be the event that a frequent itemset X does not generate an association rule in DB, for $|X| \geq 2$. The event E is equivalent to the event that a frequent itemset X generates a suggested association rule in DB, for $|X| \geq 2$. Then, $supp_a(E, D)$ could be expressed as follows.

$$supp_a(E,D) = \frac{\text{Area } A_1}{\text{Area } A_2} = \left[\frac{(\beta^2 - 3\times\beta+3)\times\alpha^2 + \beta\times(\beta-3)\times\alpha+\beta^2}{3\times(1-\alpha)}\right] \quad (2.3.16)$$

Also, each frequent itemset X could generate at least two suggested association rules, for $|X| \geq 2$. Thus, Theorem 2.3.2 follows. •

2.3.6 Error calculation

To evaluate the proposed technique of synthesizing heavy association rules we have measured the amount of error occurred in the experiments. Error of an experiment is relative to the number of transactions, number of items, and the length of a transaction in the databases. Thus, the error of an experiment needs to be expressed along with the *ANT*, *ALT*, and *ANI* in the given databases, where *ANT*, *ALT*, and *ANI* denote the average number of transactions, the average length of a transaction, and the average number of items in a database, respectively. There are several ways one could define error of an experiment. The definition of error of an experiment is based on the frequent itemsets generated from heavy association rules. Let r: $\{c\} \rightarrow \{d\}$ be a heavy association rule. The frequent itemsets generated from association rule r are $\{c\}$, $\{d\}$, and $\{c, d\}$. Let $\{X_1, X_2, \ldots, X_m\}$ be set of frequent itemsets generated from all the heavy association rules in D. We define following two types of error of an experiment.

1. Average Error (AE)

$$\text{AE}(D, \alpha, \mu, \nu) = \frac{1}{m}\sum_{i=1}^{m}\left|supp_a(X_i, D) - supp_s(X_i, D)\right| \qquad (2.3.17)$$

2. Maximum Error (ME)

$$\text{ME}(D, \alpha, \mu, \nu) = maximum\left\{\left|supp_a(X_i, D) - supp_s(X_i, D)\right|, i = 1, 2, \ldots, m\right\} \qquad (2.3.18)$$

$supp_a(X_i, D)$ and $supp_s(X_i, D)$ are actual (i.e., apriori) support and synthesized support of the itemset X_i in D, respectively.

Example 2.3.3. With reference to the Example 2.3.2, r_2: $C \rightarrow G$ and r_5: $A \rightarrow B$ are heavy association rules in D. The frequent itemsets generated from r_2 and r_5 are A, B, C, G, AB and CG. For the purpose of finding the error of an experiment, we need to find the actual support of the itemsets generated from the heavy association rules. The actual support of an itemset generated from a heavy association rule could be obtained by mining all the databases D_1, D_2, and D_3 together. Thus, AE(D, 0.2, 0.3, 0.4) = $\frac{1}{6}\{|supp_a(A, D) - supp_s(A, D)| + |supp_a(B, D) - supp_s(B, D)| + |supp_a(C, D) - supp_s(C, D)| + |supp_a(G, D) - supp_s(G, D)| + |supp_a(AB, D) - supp_s(AB, D)| + |supp_a(CG, D) - supp_s(CG, D)|\}$.

ME(D, 0.2, 0.3, 0.4) = *maximum* $\{|supp_a(A, D) - supp_s(A, D)|, |supp_a(B, D) - supp_s(B, D)|, |supp_a(C, D) - supp_s(C, D)|, |supp_a(G, D) - supp_s(G, D)|, |supp_a(AB, D) - supp_s(AB, D)|, |supp_a(CG, D) - supp_s(CG, D)|\}$. •

2.3.7 Experiments

We have carried out several experiments to study the effectiveness of our approach. All the experiments have been implemented on a 1.6 GHz Pentium processor with 256 MB of memory using visual C++ (version 6.0) software. We present the experimental results using three real databases. The database *retail* [34] is obtained from an anonymous Belgian retail supermarket store. The databases *BMS-Web-Wiew-1* and *BMS-Web-Wiew-2* can be found from KDD CUP 2000 [34]. We present some characteristics of these databases in Table 2.3.2. We use notations *DB*, *NT*, *AFI*, *ALT*, and *NI* to denote a database, the number of transactions, the average frequency of an item, the average length of a transaction, and the number of items in the corresponding database, respectively.

Table 2.3.2. Database characteristics

Database	*NT*	*ALT*	*AFI*	*NI*
retail	88,162	11.30576	99.67380	10000
BMS-Web-Wiew-1	1,49,639	2.00000	155.71176	1922
BMS-Web-Wiew-2	3,58,278	2.00000	7165.56000	100

Each of the above databases is divided into 10 databases for the purpose of carrying out experiments. The databases obtained from *retail*, *BMS-Web-Wiew-1*, and *BMS-Web-Wiew-2* are named as R_i, B_{1i} and B_{2i}, respectively, for $i = 0, 1, \ldots, 9$. The databases R_j and B_{ij} are called branch databases, for $i = 1, 2$, and $j = 0, 1, \ldots, 9$. Some characteristics of these branch databases are presented in Table 2.3.3.

Outputs of three experiments using the Algorithm 2.3.1 are presented in Table 2.3.4. The choice of different parameters is an important issue. We have chosen different α and β for different databases. But, they are kept same for branch databases obtained from the same database. For example, α and β are the same for branch databases R_i , for $i = 0, 1, \ldots, 9$.

After mining a single branch database from a group of branch databases using a reasonably low α and β, one could fix α and β for the purpose data mining task. If α and β are smaller, the multi-database mining application would produce more correct result. As we are constraint with the computing resources, we could choose α and β in such a way that all the patterns could be handled effectively.

The choice of μ and ν are context dependent and subjective. Also, if μ and ν are kept fixed then some databases might not report heavy association rules, while other databases might report many heavy association rules. While generating association rule one could estimate the average synthesized support and confidence based on the generated association rules. Thus, it gives an idea of thresholds for high-support and high-confidence for synthesizing heavy association rules in different databases. Also, the choice of γ_1 and γ_2 are also context dependent and subjective. Good values of γ_1 and γ_2 could lie in the interval [0.3, 0.4] and [0.6, 0.7], respectively. We have taken $\gamma_1 = 0.35$, and $\gamma_2 = 0.60$ for synthesizing heavy association rules.

Table 2.3.3. Branch database characteristics

DB	*N T*	*ALT*	*AFI*	*NI*	*DB*	*N T*	*ALT*	*AFI*	*NI*
R_0	9000	11.24389	12.07001	8384	R_5	9000	10.85578	16.70977	5847
R_1	9000	11.20922	12.26541	8225	R_6	9000	11.20011	17.41552	5788
R_2	9000	11.33667	14.59657	6990	R_7	9000	11.15511	17.34554	5788
R_3	9000	11.48978	16.66259	6206	R_8	9000	11.99711	18.69032	5777
R_4	9000	10.95678	16.03953	6148	R_9	7162	11.69199	15.34787	5456
B_{10}	14000	2.00000	14.94130	1874	B_{15}	14000	2.00000	280.00000	100
B_{11}	14000	2.00000	280.00000	100	B_{16}	14000	2.00000	280.00000	100
B_{12}	14000	2.00000	280.00000	100	B_{17}	14000	2.00000	280.00000	100
B_{13}	14000	2.00000	280.00000	100	B_{18}	14000	2.00000	280.00000	100
B_{14}	14000	2.00000	280.00000	100	B_{19}	23639	2.00000	472.78000	100
B_{20}	35827	2.00000	1326.92590	54	B_{25}	35827	2.00000	716.54000	100
B_{21}	35827	2.00000	1326.92590	54	B_{26}	35827	2.00000	716.54000	100
B_{22}	35827	2.00000	716.54000	100	B_{27}	35827	2.00000	716.54000	100
B_{23}	35827	2.00000	716.54000	100	B_{28}	35827	2.00000	716.54000	100
B_{24}	35827	2.00000	716.54000	100	B_{29}	35835	2.00000	716.70000	100

Table 2.3.4. First five heavy association rules reported from different databases (sorted in non-increasing order on synthesized support).

Data base	α	β	μ	ν	Heavy assoc rules	Syn supp	Syn conf	High freq	Excep tional
$\bigcup_{i=0}^{9} R_i$	0.05	0.2	0.1	0.5	{48}→{39}	0.33055	0.67629	Yes	No
					{39}→{48}	0.33055	0.56333	Yes	No
					{41}→{39}	0.12910	0.62535	Yes	No
					{38}→{39}	0.11734	0.66069	Yes	No
					{41}→{48}	0.10208	0.51495	Yes	No
$\bigcup_{i=0}^{9} B_{1i}$	0.01	0.2	0.007	0.1	{1}→{5}	0.00858	0.13422	No	No
					{5}→{1}	0.00858	0.10873	No	No
					{7}→{5}	0.00828	0.11503	No	No
					{5}-->{7}	0.00828	0.10843	No	No
					{3}→{5}	0.00746	0.12376	No	No
$\bigcup_{i=0}^{9} B_{2i}$	0.006	0.01	0.01	0.1	{3}→{1}	0.02145	0.14431	Yes	No
					{1}→{3}	0.02145	0.14295	Yes	No
					{7}→{1}	0.02096	0.14039	Yes	No
					{1}→{7}	0.020956	0.13999	Yes	No
					{5}→{1}	0.020758	0.14081	Yes	No

The experiments conducted on three databases result no exceptional association rule. Normally, exceptional association rules are rare. Also, we have not found any association rule which is heavy as well as high-frequent in multiple databases obtained from *BMS-Web-Wiew-1*.

In many applications, the suggested association rules are significant. While synthesizing the association rules from different databases we might need to consider the suggested association rules for the correctness of synthesizing association rules. We have observed that the number of suggested association rules in the set of databases $\{R_0, R_1, \ldots, R_9\}$ and $\{B_{10}, B_{11}, \ldots, B_{19}\}$ are significant. But, the set of databases $\{B_{20}, B_{21}, \ldots, B_{29}\}$ do not generate any suggested association rule. We present the number of association rules and the number of suggested association rules in these sets of databases in Table 2.3.5.

Table 2.3.5. Number of association rules and suggested association rules extracted from multiple databases

Database	α	β	Number of association rules (N)	Number of suggested association rules (M)	$M / (N + M)$
$\bigcup_{i=0}^{9} R_i$	0.05	0.2	821	519	0.387313
$\bigcup_{i=0}^{9} B_{1i}$	0.01	0.2	50	96	0.657534
$\bigcup_{i=0}^{9} B_{2i}$	0.006	0.01	792	0	0.000000

The error of synthesizing association rules in a database is relative to the following parameters: the number of transactions, the number of items, and the length of transactions in the given databases. If the number of transactions in a database increases the error of synthesizing association rules increases, provided other two parameters remain constant. If the length of a transaction of a database increases the error of synthesizing association rules is likely to increase, provided other two parameters remain constant. Lastly, if the number of items increases the error of synthesizing association rules is likely to decrease, provided other two parameters remain constant. Thus, the error of an experiment needs to be expressed along with the *ANT*, *ALT*, and *ANI* for the given databases. The errors of different experiments are presented in Table 2.3.6.

Table 2.3.6. Error of synthesizing the heavy association rules

Database	α	β	μ	ν	(AE, *ANT*, *ALT*, *ANI*)	(ME, *ANT*, *ALT*, *ANI*)
$\bigcup_{i=0}^{9} R_i$	0.05	0.2	0.1	0.5	(0.002503, 8816.2, 11.305755, 5882.1)	(0.003612, 8816.2, 11.305755, 5882.1)
$\bigcup_{i=0}^{9} B_{1i}$	0.01	0.2	0.007	0.1	(0.000365, 14963.9, 2.0, 277.4)	(0.000759, 14963.9, 2.0, 277.4)
$\bigcup_{i=0}^{9} B_{2i}$	0.006	0.01	0.01	0.1	(0.000118, 35827.8, 2.0, 90.8)	(0.000285, 35827.8, 2.0, 90.8)

2.3.7.1 Comparison with existing algorithm

In this section we make a detailed comparison between the part of the proposed algorithm that synthesizes only high-frequent association rules and the algorithm *RuleSynthesizing* [81]. Let the part of the proposed algorithm be *High-Frequency-Rule-Synthesis* that synthesizes only high-frequent association rules in different databases. We conduct experiments for comparing algorithms *High-Frequency-Rule-Synthesis* and *RuleSynthesizing*. We compare these two algorithms on the basis of the following two issues: (i) Average error, and (ii) Execution time.

2.3.7.1.1 *Analysis of average error*

Both the definitions of average errors are similar and use the same set of synthesized frequent itemsets. But, the methods of synthesizing frequent itemsets for these two approaches are different. Thus, the amount of error incurred in these two approaches might differ. In *RuleSynthesizing* algorithm, if an itemset fails to get extracted from a database then the support of the itemset is assumed as 0. But, in *Association-Rule-Synthesis* algorithm, if an itemset fails to get extracted from a database then the support of the itemset is estimated. Thus, the synthesized support of an itemset in the union of concerned databases for these two approaches might differ. As the number of databases increases the relative presence of a rule normally decreases. Thus, the error of synthesizing a rule normally increases. So, the AE of an experiment is likely to increase if the number of databases increases. We observe such phenomenon in Figures 2.3.3 and 2.3.4.

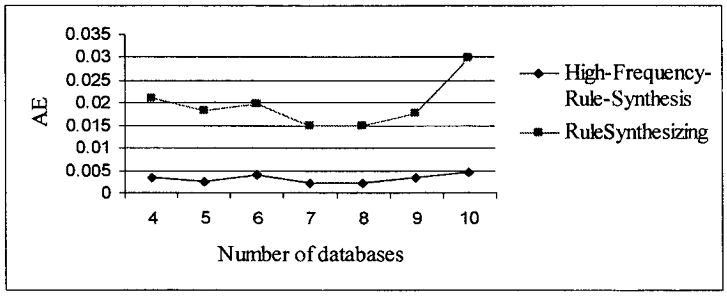


Figure 2.3.3. AE versus number of databases from *retail* at (α, β, γ) = (0.05, 0.2, 0.6)

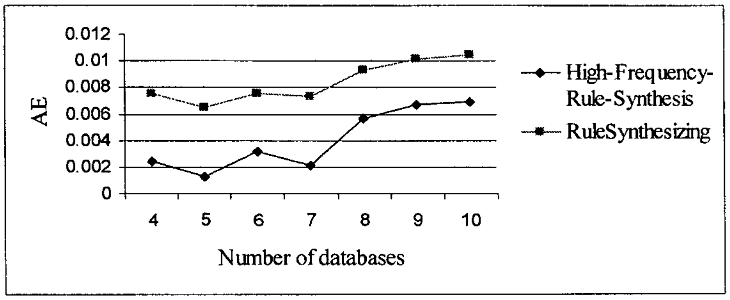


Figure 2.3.4. AE versus number of databases from *BMS-Web-Wiew-1* at (α, β, γ) = (0.005, 0.1, 0.3)

The proposed algorithm follows direct approach in identifying high-frequent association rules as opposed to *RuleSynthesizing* algorithm. Here, we study AE of the experiments for these two approaches. In Figures 2.3.3 and 2.3.4, we observe that AE error of an experiment conducted using *High-Frequency-Rule-Synthesis* algorithm is less than that of *RuleSynthesizing* algorithm.

2.3.7.1.2 *Analysis of execution time*

We have also conducted experiments to study the execution time by varying the number of databases. The number of synthesized frequent itemsets increases as the number of databases increases. Thus, the execution time normally increases with the increase of number of databases. We observe such phenomenon in Figures 2.3.5 and 2.3.6.

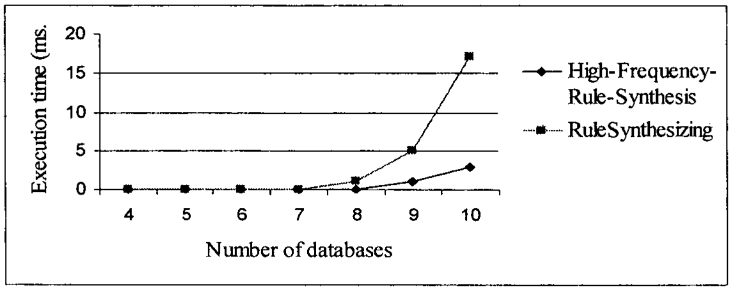


Figure 2.3.5. Execution time versus number of databases from *retail* at (α, β, γ) = (0.05, 0.2, 0.6)

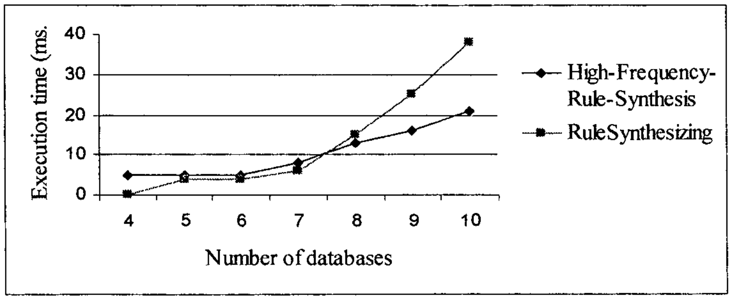


Figure 2.3.6. Execution time versus number of databases from *BMS-Web-Wiew-1* at (α, β, γ) = (0.005, 0.1, 0.3)

RuleSynthesizing algorithm might be faster than *High-Frequency-Rule-Synthesizing* algorithm for less number of databases. As the number of databases increases, *High-Frequency-Rule-Synthesizing* algorithm executes faster than *RuleSynthesizing* algorithm.

2.3.8 Conclusions

Synthesizing heavy association rule is an important component of a multi-database mining system. In this chapter, we present the notions of two new patterns in multiple databases viz., heavy association rule and exceptional association rule. Also, we presents an algorithm for synthesizing three important patterns in multiple databases viz., heavy association rules, high-frequent association rules, and exceptional association rules. It also provides a better solution for synthesizing high-frequent association rules in multiple databases. The algorithm presented here is simple and effective for synthesizing heavy association rules in multiple real databases.

Chapter 2.4

Clustering frequent items in multiple databases

Due to a liberal economic policy adopted by many countries across the globe, the number of branches of a multi-national company as well as the number of multi-national companies is increasing over time. Moreover, the economies of many countries are growing at a faster rate. As a result the number of multi-branch companies within a country is also increasing. Many of these companies collect a huge amount of data through different branches. Consider a multi-branch company that transacts from multiple branches. Each branch maintains a separate database for the transactions made at the branch. Thus, the company deals with multiple transactional databases. Data mining and knowledge discovery from large database is often considered as the basis of many decision-support applications. But, the most of the previous pieces of data mining work are based on a single database. Thus, it is necessary to study data mining on multiple databases.

An *itemset* is a collection of items in a database. Each itemset in a database is associated with a statistical measure called *support* [11]. Support of an itemset X in database D is the fraction of transactions in D containing X, denoted by $S(X, D)$. In general, let $S(E, D)$ be the support of a Boolean expression E defined on the transactions in database D. An itemset X is called *frequent* in D if $S(X, D) \geq \alpha$, where α is user defined level of *minimum support*. If X is frequent then $Y \subseteq X$ is also frequent, since $S(Y, D) \geq S(X, D)$, for $Y \neq \phi$. Thus, each item of a frequent itemset is also frequent. Itemset could be considered as a basic type of pattern in a transactional database. The collection of frequent itemsets determines major characteristics of a database. Many interesting algorithms [13], [39], [66] have been proposed to mine frequent itemsets in a database.

Thus, there are many implementations [32] for extracting frequent itemsets from a database. Itemset patterns influence heavily current KDD research. We observe the influence of itemset patterns on KDD research in the following ways: Firstly, many algorithms have been reported on mining frequent itemsets in a database. Secondly, many patterns are based on the itemset patterns in a database. Thus, they could be called as derived patterns in a database. For example, positive association rule [11] and high-frequent association rule [81] are examples of some derived patterns. Considerable amount of work have been reported on mining / synthesizing derived patterns in a database [13], [39], [66], [89]. Finally, solutions of many problems could be based on the analysis of patterns in a database [79], [83]. Such applications process patterns in a database for the purpose of making some decisions. Frequent items are the ingredients of most of the interesting patterns. Thus, the analysis and synthesis of frequent items is an interesting as well as important issue. In multi-database environment, local frequent items are interesting as well as important issue. They are used to construct the global patterns in multiple databases. Thus, clustering of frequent items in multiple databases is an important knowledge for a multi-branch company. Many important decisions could be based on clustering of frequent items in multiple databases. In the following, we mention a few such applications.

- Some of the frequent items (products) could be high profit making. Naturally, the company would like to promote them. There are various ways one could promote an item. An indirect way of promoting an item *P* is to promote items that are positively associated with it. The implication of positive association between *P* and another item *Q* is that if *Q* is purchased by a customer then *P* is likely to be purchased by the same customer at the same time. Thus, *P* is indirectly promoted. Clustering of frequent items could help identifying other items that promote a specific frequent item.
- Some frequent items could be of high standard. Thus, they bring goodwill for the company. They help promoting other items. Thus, it is important to know how the sales of these items affect the other items. Before making such analyses, one may need

to cluster the frequent items.

- Again, some of the frequent items could be low-profit making. Thus, it is important to know how they promote the sales of other items. Otherwise, the company could stop dealing with such items. Clustering of frequent items could help identifying items that do not promote other items.

Many corporate decisions could be taken effectively by incorporating knowledge inherent in data across the branches. But, the effective management of multiple large databases becomes a challenging issue. It creates not only opportunities but also risks. The risks might involve significant amount of investment on hardware and software to deal with the large volume of data. Our objective is to provide good solutions by minimizing the risks.

In this chapter, we synthesize highly extracted itemsets based on local itemsets. Highly extracted itemsets are defined in Section 2.4.4. We measure association among items in synthesized highly extracted itemsets. We cluster frequent items based on associations among items in highly extracted itemsets. Highly associated items could be put in the same class. The motivation of proposed clustering technique is given as follows.

Wu et al. [83] have proposed a technique for clustering a set of databases. The principle of clustering could be stated as follows. It finds association between every pair of objects (databases) using a measure of association. A set of m arbitrary objects form a class, if ${}^{m}C_2$ association values corresponding to ${}^{m}C_2$ pairs of objects are close. The level of association among the objects in this class is assumed as the minimum of ${}^{m}C_2$ association values. One could apply such technique for clustering frequent items in multiple transactional databases. If the number of items in a class is more than two, then we observe that this technique might fail to estimate the association among the items in the class correctly. Then accuracy of the entire clustering process becomes low. The proposed clustering technique follows a different approach and it clusters frequent items with higher degree of accuracy as compared to the existing technique.

Initially, we overview the existing measures association among a set of items in a database. Afterwards, we overview the existing techniques of mining multiple databases. In the context of mining multiple databases, we introduce the concept of highly extracted itemsets. We design an algorithm for synthesizing support of each highly extracted itemset. The algorithm also synthesizes association among items in a highly extracted itemset. Based on the synthesized associations corresponding to different highly extracted itemsets, we propose an algorithm for finding the best clustering of frequent items in multiple databases. Finally, we present experimental results to show the effectiveness of the proposed clustering technique.

The rest of the chapter is organized as follows. In Section 2.4.2, we study the existing measures of association, and the existing techniques for mining multiple databases. We also study different clustering techniques and other related issues. In Section 2.4.3, we discuss some results. We propose an algorithm for synthesizing supports of highly extracted itemsets in Section 4.2.4. The algorithm also synthesizes association among items in a highly extracted itemset of size greater than one. In Section 2.4.5, we propose an algorithm for clustering frequent items in multiple databases. Finally, experimental results are presented in Section 2.4.6 to show the effectiveness of the proposed clustering technique.

2.4.2 Problem statement

Consider a multi-branch company that operates from n branches. Let D_i be the database corresponding to the i-th branch, for $i = 1, 2, \ldots, n$. Also, let D be the union of these databases. In the context of clustering frequent items in multiple databases, first we discuss work related to this issue.

2.4.2.1 Related work

Our clustering technique is based on itemset patterns in multiple databases. In this context, one needs a technique for mining itemset patterns in multiple databases. Afterwards, association among items in an itemset is studied using a measure of association. Finally, one needs a clustering algorithm to cluster frequent items in multiple databases. Thus, we divide related work broadly into three areas: measures of association, techniques of mining multiple databases, and clustering algorithms.

2.4.2.1.1 *Measures of association*

A survey of different measures of association is provided in the first and second sections of Chapter 1.4. Also, we have explained why the existing measures are not able to capture association among a set of items in a database accurately. In Chapter 1.4, we have also presented two generalized measures of association A_1 and A_2. We use measure A_2 for clustering frequent items in multiple databases.

2.4.2.1.2 *Multi-database mining techniques*

The first question comes to our mind whether a traditional data mining technique could deal with multiple large databases. To apply a traditional data mining technique one needs to amass all the databases together. A single computer might take unreasonable amount of time to process the entire database. Sometimes, it might not be feasible to mine large volume of data using a single computer. Another solution to this problem would be to employ parallel machines. It might require high investment on hardware and software. One needs to make a cost-benefit analysis before implementing such a decision. In many situations, it might not be an acceptable solution to the management of the company. Moreover, it might be difficult to find local patterns when a mining technique is applied to the entire database. Thus, the traditional data mining techniques are not suitable in this situation. So, it is a different problem. Hence, it is required to be dealt with in a different way. In this situation, one could employ the model of local pattern analysis [91] to deal with multiple large databases. In this case, each branch is required to forward local patterns instead of original database to the central office for synthesis and analysis of local patterns. But, local pattern analysis might return approximate global patterns.

For the purpose of mining multiple databases, one could apply partition algorithm proposed by Savasere et al. [66]. The algorithm was designed to mine a very large database by partitioning. The algorithm works as follows. It scans the database twice. The database is divided into disjoint partitions, where each partition is small enough to fit in memory. In a first scan, the algorithm reads each partition and computes locally frequent itemsets in each partition using apriori algorithm [13]. In the second scan, the algorithm counts the supports of all locally frequent itemsets toward the complete database. In this case, each local database could be considered as a partition. Though partition algorithm mines frequent itemsets exactly, it is an expensive solution to mining multiple large databases, since each database is required to scan twice.

For mining multiple databases, there are three situations: (i) Each local database is small, so that a single database mining technique (SDMT) could mine the union of all databases. (ii) At least one of the local databases is large, so that a SDMT could mine every local database, but fail to mine the union of all local databases. (iii) At least one of the local databases is very large, so that a SDMT fails to mine every local database. We face challenges to handle the cases (ii) and (iii). The challenges posed to us are due to large size of some of the local databases.

A multi-database mining technique (MDMT) using local pattern analysis could be viewed as a two-step process M+S, explained as follows.

- Mine each local database using a SDMT by following a model M (Step 1)
- Synthesize patterns using an algorithm S (Step 2)

We use notation MDMT: M+S to represent above multi-database mining technique.

In the context of Step 1 of a MDMT using local pattern analysis, Zhang et al. [89] have proposed algorithm *IdentifyExPattern* (IEP) for identifying global exceptional patterns in multi-databases. Every local database is mined separately at *random order* (RO) using a SDMT for synthesizing global exceptional patterns.

In the context of Step 2 of a MDMT using local pattern analysis, Zhang et al. [89] have proposed a technique for synthesizing global patterns. In algorithm *IdentifyExPattern*, a pattern in a local database is assumed as nonexistent, if it does not get reported. Let $supp_a(p, DB)$ and $supp_s(p, DB)$ be the actual (i.e, apriori) support and synthesized support of pattern p in database DB. Support of pattern p in D has been synthesized as follows.

$$supp_s(p, D) = \frac{1}{num(p)} \sum_{i=1}^{num(p)} \frac{supp_a(p, D_i) - \alpha}{1 - \alpha} \tag{2.4.1}$$

where, $num(p)$ is the number of databases that report p at a given minimum support level (α).

Adhikari and Rao [5] have proposed *Association-Rule-Synthesis* (ARS) algorithm for synthesizing association rules in multiple real databases. For multiple real databases, the trend of the customers' behaviour exhibited in one database is usually present in other databases. In particular, a frequent itemset in one database is usually present in some transactions of other databases even if it does not get extracted. The estimation procedure captures such trend and estimates the support of a missing association rule. Without loss of generality, let the itemset X be extracted from first m databases, for $1 \le m \le n$. Then trend of X in first m databases could be expressed as follows.

$$trend^{1,m}(X \mid \alpha) = \frac{1}{\sum_{i=1}^{m} |D_i|} \times \sum_{i=1}^{m} [supp_a(X, D_i) \times |D_i|] \tag{2.4.2}$$

One could use the trend of X in first m databases for synthesizing support of X in D. We estimate support of X in each of the remaining databases by $\alpha \times trend^{1,n}(X \mid \alpha)$, for $j = k + 1, k + 2, \ldots, n$. Thus, the synthesized support of X could be computed as follows.

$$supp_s(X, D) = \frac{trend^{1,m}(X \mid \alpha)}{\sum_{i=1}^{n} |D_i|} \times \left[(1 - \alpha) \times \sum_{i=1}^{m} |D_i| + \alpha \times \sum_{i=1}^{n} |D_i|\right] \tag{2.4.3}$$

In synthesizing high-frequent association rule, Wu and Zhang [81] have proposed *RuleSynthesizing* (RS) algorithm for synthesizing high-frequent association rule in multiple databases. Based on the association rules in different databases, the authors have estimated weights of different databases. Let w_i be the weight of i-th database, for $i = 1$, 2, ..., n. Without loss of generality, let the association rule r be extracted from first m databases, for $1 \le m \le n$. $supp_a(r, D_i)$ has been assumed as 0, for $i = m + 1, m + 2, \ldots, n$. Then support of r in D has been synthesized as follows.

$supp_s(r,D) = w_1 \times supp_a(r,D_1) + \ldots + w_m \times supp_a(r,D_m)$ (2.4.4)

Adhikari and Rao [8] have proposed *pipelined feedback model* (PFM) for mining multiple databases. Let $W_1, W_2, \ldots, W_n$ be n local data warehouses. In PFM, W_1 is mined using a SDMT and local pattern base LPB_1 is extracted. While mining W_2, all the patterns in LPB_1 are extracted irrespective of their values of interestingness measures like, minimum support and minimum confidence. Apart from these patterns, some new patterns that satisfy user-defined threshold values of interestingness measures are also extracted. In general, while mining W_i, all the patterns in W_{i-1} are mined irrespective of their values of interestingness measures and some new patterns that satisfy user-defined threshold values of interestingness measures, for $i = 2, 3, \ldots, n$. Due to this nature of mining each data warehouse, the technique is called a feedback model. Thus, $|LPB_{i-1}| \le |LPB_i|$, for $i = 2, 3, \ldots, n$. There are $n!$ arrangements of pipelining for n databases. All arrangements of data warehouses would not produce the same mining result. If the number of local patterns increases, we get more accurate global patterns and a better analysis of local patterns. An arrangement of data warehouses would produce near optimal result if $|LPB_n|$ is a maximal. Let $size(W_i)$ be the size of W_i (in bytes), for $i = 1, 2, \ldots, n$. We shall follow the following rule of thumb regarding the arrangements of data warehouses for the purpose of mining. The number of patterns in W_{i-1} is greater than or equal to the number of patterns in W_i, if $size(W_{i-1}) \ge size(W_i)$, for $i = 2, 3, \ldots, n$. For the purpose of increasing number of local patterns, W_{i-1} precedes W_i in the pipelined arrangement of mining data warehouses if $size(W_{i-1}) \ge size(W_i)$, for $i = 2, 3, \ldots, n$. Finally, we analyze the patterns in $LPB_1, LPB_2, \ldots$, and LPB_n for synthesizing global patterns, or analyzing local patterns.

For synthesizing global patterns in D we discuss here a simple pattern synthesizing (SPS) algorithm. Without loss of generality, let the itemset X be extracted from first m

databases, for $1 \le m \le n$. Then synthesized support of X in D could be obtained as follows.

$$supp_s(X, D) = \frac{1}{\sum_{i=1}^{n} |D_i|} \times \sum_{i=1}^{m} [supp_a(X, D_i) \times |D_i|] \qquad (2.4.5)$$

There are two benefits of the PFM model. Firstly, it improves significantly the accuracy of mining multiple large databases as compared to local pattern analysis. Secondly, it scans each local database only once. Several experiments [8] have been conducted using MDMT: PFM+SPS, and we have observed that MDMT: PFM+SPS outperforms other MDMTs. Thus, for the purpose of clustering frequent items in multiple databases, we shall use MDMT: PFM+SPS for mining multiple databases.

Liu et al. [55] have proposed multi-database mining technique that searches only the relevant databases. Identifying relevant databases is based on selecting the relevant tables (relations) that contain specific, reliable and statistically significant information pertaining to the query. Zhang [88], Zhang et al. [93] studied various strategies for mining multiple databases.

Existing parallel mining techniques [12], [26], [28] could also be used to deal with multiple databases.

2.4.2.1.3 *Clustering techniques*

Zhang et al. [90] have proposed an efficient and scalable data clustering method BIRCH based on in-memory data structure called CF-tree. Estivill-Castro and Yang [30] have proposed an algorithm that remains efficient, generally applicable, multi-dimensional but is more robust to noise and outliers. Jain et al. [44] have presented an overview of pattern clustering methods from a statistical pattern recognition perspective, with a goal of providing useful advice and references to fundamental concepts accessible to the broad community of clustering practitioners. In this chapter, we present an algorithm based on local patterns. Thus, the above algorithms might not be suitable in this situation.

Ali et al. [15] have proposed a partial classification technique using association rules. The clustering of frequent items using local association rules might not be a good idea. The number of frequent itemsets obtained from a set of association rules might be much less than the number of frequent itemsets extracted using apriori algorithm [13]. Thus, efficiency of the clustering process might be low.

2.4.2.2 Our approach

Before we state our problem formally, we define few notations. Let $FI(D \mid \alpha)$ and $FIS(D \mid \alpha)$ be the set of frequent items and set of frequent itemsets in database D at a given α, respectively. Let $FI(1, n, \alpha)$ be equal to $\cup_{i=1}^{n} FI(D_i \mid \alpha)$ and $FIS(1, n, \alpha)$ be equal to $\cup_{i=1}^{n} FIS(D_i \mid \alpha)$. We apply measure of association A_2 and multi-database mining technique MDMT: PFM+SPS for the purpose of clustering frequent items in multiple databases. The proposed problem could be stated as follows.

There are n different databases D_i, for $i = 1, 2, \ldots, n$. Find the best non-trivial partition (if it exists) of $FI(1, n, \alpha)$ induced by $FIS(1, n, \alpha)$.

A partition [53] is a specific type of clustering. Formal definition of a non-trivial partition is given in Section 2.4.5.

2.4.3 Measuring association among items

For clustering frequent items in multiple databases, one needs to measure association among items in a database. We use measure A_2 in our clustering algorithm. For computing A_2, we make use of Lemma 1.4.7 and it is restated as follows.

Lemma 2.4.1. *Let $X = \{x_1, x_2, \ldots, x_m\}$ be an itemset in database D, for $m \geq 2$. Then*

$$A_2(X, D) = \frac{\sum_{i=1}^{m}\left[\sum_{j=1;\, j\neq i}^{m} S(\{x_i\}\cap\{x_j\}, D) - \sum_{j,k=1;\, j,k\neq i}^{m} S(\{x_i\}\cap\{x_j\}\cap\{x_k\}, D) + \ldots \pm S(\{x_1\}\cap\ldots\cap\{x_m\}, D)\right]}{m \times \left[\sum_{i=1}^{m} S(\{x_i\}, D) - \sum_{i,j=1;\, i<j}^{m} S(\{x_i\}\cap\{x_j\}, D) + \ldots \pm S(\{x_1\}\cap\{x_2\}\cap\ldots\cap\{x_m\}, D)\right]} \tag{2.4.6}$$

Proof. Please refer Lemma 1.4.7.

2.4.4 Synthesizing support of an itemset

In PFM, let the itemset X be extracted from k out of n databases, for $0 \le k \le n$. Let γ be the minimum threshold of number of extractions of an itemset, for $0 < \gamma \le 1$. We would be interested about the itemset if it has been extracted from minimum of $n \times \gamma$ databases. We call such itemsets as *highly extracted itemsets* (*HEIS*s). If an itemset X is highly extracted then an itemset $Y \subseteq X$ is also a highly extracted itemset, for $Y \neq \phi$. We define a highly extracted itemset as follows.

Definition 2.4.1. Let there are n databases. Let X be an itemset extracted from k databases. Then X is highly extracted if $k / n \ge \gamma$. •

A highly extracted itemset might not be frequent in all the databases under consideration. After applying PFM model of mining multiple databases, we synthesize supports of *HEIS*s using formula (2.4.5). In Example 2.4.1, we illustrate the procedure for synthesizing support of a *HEIS*.

Example 2.4.1. Consider a multi-branch company that has four branches. Let D_i be the database corresponding to the i-th branch, for i = 1, 2, 3, 4. The branch databases are given as follows. $D_1 = \{\{a, b\}, \{a, b, c\}, \{a, b, c, d\}, \{c, d, e\}, \{c, d, f\}, \{c, d, i\}\}$; $D_2 = \{\{a, b\}, \{a, b, g\}, \{g\}\}$; $D_3 = \{\{a, b, d\}, \{a, c, d\}, \{c, d\}\}$, $D_4 = \{\{a\}, \{a, b, c\}, \{c, d\}, \{c, d, i\}\}$. Assume that $\alpha = 0.4$, and $\gamma = 0.6$. Let $X(\eta)$ denotes the fact that the itemset X has support η in the corresponding database. We sort databases in non-increasing order on database size (in bytes). The sorted databases are given as follows: D_1, D_4, D_3, D_2. Applying PFM, the itemsets in different local databases are given as follows:

$LPB(D_1, \alpha) = \{\{a\}(0.5), \{b\}(0.5), \{c\}(0.833), \{d\}(0.667), \{a, b\}(0.5), \{c, d\}(0.667)\}$, $LPB(D_4, \alpha) = \{\{a\}(0.667), \{b\}(0.25), \{c\}(0.75), \{d\}(0.25), \{a, b\}(0.333), \{c, d\}(0.667)\}$, $LPB(D_3, \alpha) = \{\{a\}(0.667), \{b\}(0.333), \{c\}(0.667), \{d\}(1.0), \{a, b\}(0.333)$,

$\{c, d\}(0.667)\}$, and $LPB(D_2, \alpha) = \{\{a\}(0.667), \{b\}(0.667), \{c\}(0.0), \{d\}(0.0), \{a, b\}(0.667), \{c, d\}(0.0), \{g\}(0.667)\}$.

Let $D = \bigcup_{i=1}^{4} D_i$. Synthesized *HEIS*s in *D* are given as follows: $SHEIS(D, 0.4, 0.6) = \{ \{a\}(0.563), \{b\}(0.438), \{c\}(0.563), \{d\}(0.563), \{a, b\}(0.438), \{c, d\}(0.5)\}$. •

The collection of *SHEIS*s of size greater than 1 forms the basis of the proposed clustering technique. We present below an algorithm to obtain synthesized association among items in each *SHEIS* of size greater than 1. Let *N* be the number of itemsets in *n* databases. Let *AIS* be a two dimensional array such that *AIS*(*i*) is the array of itemsets extracted from D_i, for $i = 1, 2, \ldots, n$. Also, let *IS* be the set of all itemsets in *n* databases. An itemset could be described by the following attributes: *itemset*, *supp*, and *did*. Here, *itemset*, *supp* and *did* represent the itemset, support and database identification of itemset, respectively. All the synthesized itemsets are kept in the array *SIS*. Each synthesized itemsets has the following attributes: *itemset*, *ss*, and *sa*. Here, *ss* and *sa* represent synthesized support and synthesized association of the itemset, respectively. In the following algorithm, we synthesize association among items of each *SHEIS*.

Algorithm 2.4.1. Synthesize association among items of each *SHEIS* of size greater than 1.

```
procedure SynthesizeAssociation (n, AIS, size, γ)
Input:
n: number of databases
AIS: two dimensional array of itemsets extracted during mining multiple databases
size: array of number of transactions in input databases
γ: threshold for minimum number of extractions of an itemset
Output:
Synthesized association among items of each SHEIS
01:  collect all local itemsets into array IS;
02:  sort itemsets of IS based on itemset attribute;
03:  add sizes of all branch databases into variable totalTransactions;
```

```
04: let nSynItemSets = 0; let i = 1;
05: if ( $i \leq |IS|$ ) then
06:   let j = i; let count = 0;
07:   while ($j \leq i + n$) do
08:     if (IS(j).itemset = IS(i).itemset) then
09:       process support of IS(i);
10:       increase count by 1; increase j by 1;
11:     else go to line 14;
12:     end if
13:   end while
14:   synSupp = $supp_s$(IS(i).itemset, D) using formula (2.4.5) and totalTransactions;
15:   if ($count / n \geq \gamma$) then
16:     SIS(nSynItemSets). ss = synSupp;
17:     SIS(nSynItemSets). itemset = IS(i).itemset;
18:   end if
19:   update i by j;
20:   increase nSynItemSets by 1;
21:   go to line 5;
22: end if
23: initialize synthesized association to $\delta$ for each itemset in SIS;
24: for j = 1 to nSynItemSets do
25:   if (|SIS(nSynItemSets). itemset| $\geq$ 2) then
26:     SIS(nSynItemSets).sa = $A_2$(SIS(nSynItemSets).itemset, D) using formula (2.4.6);
27:   end if
28: end for
end procedure
```

We sort itemsets of *IS*, so that processing of itemsets becomes easier. We find total number of transactions in different databases into variable *totalTransactions*. The variables *nSynItemSets* and *i* keep track of the number synthesized itemsets and the current itemset of *IS*, respectively. The algorithm segment in lines 5 - 22 is repeated *N* times. An itemset gets processed at each iteration. An itemset occurs maximum *n* times. Thus, the while-loop in lines 7-13 repeats maximum *n* times. The variable *count* keeps track of number of times an itemset is extracted. Based on variable *count* one could determine whether an itemset is highly extracted. If an itemset is highly extracted then we store the details into array *SIS* and increase *nSynItemSets* by 1. We update variable *i* by *j* for processing the next itemset. We go back to line 5 for processing the next itemset. Using lines 24-28, we calculate synthesized association using formula (2.4.6), for each synthesized itemset of size greater than 1. In the next paragraph, we determine the time complexity of above algorithm.

Line 1 takes $O(N)$ time, since there are N itemsets in n databases. Line 2 takes $O(N \times \log(N))$ time to sort N itemsets. Line 3 takes $O(n)$ time, since there are n databases. The while-loop at line 7 repeats maximum n times. The if-statement at line 5 repeats N times. Thus, time complexity of program segment in lines 5-22 is $O(n \times N)$. Line 23 takes $O(N)$ time. Let the average size of a class be p. The time complexity for searching an itemset in *IS* is $O(N)$. The time-complexity for computing association of an itemset is $O(N \times p^2)$, and hence, the time complexity of program segment in lines 24-28 is $O(N^2 \times p^2)$ time. Thus, the time complexity of procedure *SynthesizeAssociation* is equal to *maximum* $\{O(N^2 \times p^2), O(N \times \log(N)), O(n \times N)\} = O(N^2 \times p^2)$, since $N > \log(N)$, and $N > n$.

2.4.5 Clustering of frequent items

Existing technique [83] for clustering multiple databases works as follows. A measure of similarity between two databases is proposed. Let there are m databases to be clustered. Then the similarities for ${}^{m}C_2$ pairs of databases are computed. Based on a level of similarity, the databases are clustered into different classes. For the purpose of clustering databases, the following measure of similarity between two databases has been proposed [83].

$$sim_2(D_1, D_2) = \frac{I(D_1) \cap I(D_2)}{I(D_1) \cup I(D_2)} \tag{2.4.7}$$

In the context of similarity between two items in a database, we have observed in Corollary 2.4.1 that a measure sim_2 could be obtained from similarity measure A_2, for $X = \{x_1, x_2\}$. It could also be used to cluster frequent items in a database. In the following example, we shall show that association among items of an itemset could not be determined accurately using this approach. In particular, association among items of $\{a, b, c\}$ could not be correctly estimated by associations among items of $\{a, b\}$, $\{a, c\}$, and $\{b, c\}$. We explain this issue in Example 2.4.2.

Example 2.4.2. Let $D_5 = \{ \{a, b, c, d\}, \{a, b, c, e\}, \{a, b, d\}, \{a, e, f\}, \{b, c, e\}, \{d, e, g\}, \{d, f, g\}, \{e, f, g\}, \{e, f, h\}, \{g, h, i\} \}$. Also, let α be 0.2. The supports of relevant frequent itemsets are given as follows. $S(\{a\}, D_5) = 0.4$, $S(\{b\}, D_5) = 0.4$, $S(\{c\}, D_5) = 0.3$, $S(\{a, b\}, D_5) = 0.3$, $S(\{a, c\}, D_5) = 0.2$, $S(\{b, c\}, D_5) = 0.3$, $S(\{a, b, c\}, D_5) = 0.2$. Now, $sim_2(\{a, b\}, D_5) = 0.6$, $sim_2(\{a, c\}, D_5) = 0.4$, $sim_2(\{b, c\}, D_5) = 0.75$. Using sim_2, the items a, b, and c could be put in the same class at the level of similarity 0.4, i.e., $minimum\{0.6, 0.4, 0.75\}$. Using A_2, we have $A_2(\{a, b, c\}, D_5) = 0.66667$. Thus, the items a, b, and c could be put in the same class at the level 0.66667. We observe that the subset of transactions $\{ \{a, b, c, d\}, \{a, b, c, e\}, \{a, b, d\}, \{a, e, f\}, \{b, c, e\}\}$ of D_1 results in the amount of association among a, b, and c. Two out of five transactions contain two items of $\{a, b, c\}$. Two out of five transactions contain all the items of $\{a, b, c\}$. The more items of $\{a, b, c\}$ occur together, higher is the association among items of $\{a, b, c\}$. Thus, we observe that the amount of association among the items of $\{a, b, c\}$ is close to 0.66667 rather than 0.4. Thus, we fail to measure association correctly among the items of $\{a, b, c\}$ based on the similarities between items of $\{a, b\}$, $\{a, c\}$, and $\{b, c\}$. •

The above example shows that the existing clustering technique might cluster a set of frequent items with low accuracy. Thus, we have the following observation.

Observation 2.4.1. *Let $X = \{x_1, x_2, ..., x_m\}$ be an itemset in database D. The existing clustering technique* [83] *puts items of X in a class at the level of association minimum*

$\{sim_2(x_i, x_j, D): 1 \leq i < j \leq m\}$. The proposed clustering technique puts items of X in a class at the level of association $A_2(\{x_1, x_2, ..., x_m\}, D)$. •

A clustering of items results in a set of classes of items. A class of frequent items over $FI(1, n, \alpha)$ could be defined as follows.

Definition 2.4.2. A class $class^{\delta}$ formed at a level of association δ under the measure of association A_2 over $FI(1, n, \alpha)$ in database D is defined as $X \subseteq FI(1, n, \alpha)$ such that $A_2(X, D) \geq \delta$, and one of the following conditions is satisfied: (i) $X \in SHEIS(1, n, \alpha, \gamma)$, for $|X| \geq 2$, and (ii) $X \in FI(1, n, \alpha)$, for $|X| = 1$. •

Definition 2.4.3 enables us to define a clustering of frequent items over $FI(1, n, \alpha)$ as follows.

Definition 2.4.3. Let π^{δ} be a clustering of frequent items over $FI(1, n, \alpha)$ at level of association δ under the measure of association A_2. Then, $\pi^{\delta} = \{X$: X is a class of type $class^{\delta}$ over $FI(1, n, \alpha)$ $\}$. •

We symbolize the i-th class of π^{δ} as $CL_i^{\delta,\alpha}$, for $i = 1, 2, ..., |\pi^{\delta}|$. A clustering may not include all the frequent items in local databases. One might be interested in clustering of all frequent items under consideration. A complete clustering of frequent items over $FI(1, n, \alpha)$ is defined as follows.

Definition 2.4.4. A clustering $\pi^{\delta} = \{CL_1^{\delta,\alpha}, CL_2^{\delta,\alpha}, ..., CL_m^{\delta,\alpha}\}$ is complete, if $\bigcup_{i=1}^{m} \mathrm{CL}_i^{\delta,\alpha} = FI(1, n, \alpha)$, where $CL_i^{\delta,\alpha}$ is a class of type $class^{\delta}$ over $FI(1, n, \alpha)$, for $i = 1, 2, ..., m$. •

Two classes in a clustering might not be mutually exclusive. One might be interested in finding out a mutually exclusive clustering. A mutually exclusive clustering over $FI(1, n, \alpha)$ could be defined as follows.

Definition 2.4.5. A clustering $\pi^{\delta} = \{CL_1^{\delta,\alpha}, CL_2^{\delta,\alpha}, ..., CL_m^{\delta,\alpha}\}$ is mutually exclusive if $CL_i^{\delta,\alpha} \bigcap CL_j^{\delta,\alpha} = \phi$, $CL_i^{\delta,\alpha}$ and $CL_j^{\delta,\alpha}$ are classes of type $class^{\delta}$ over $FI(1, n, \alpha)$, for $i \neq j$, i, $j = 1, 2, ..., m$. •

We are interested in finding out such a mutually exclusive and complete clustering. In fact, we are interested in finding the best non-trivial partition of frequent items. First, we define a partition of frequent items as follows.

Definition 2.4.6. A complete and mutually exclusive clustering is called a partition. •

A clustering is not necessarily be a partition. In most of the cases, a trivial partition might not be interesting to us. We define a non-trivial partition of frequent items as follows.

Definition 2.4.7. A partition π is non-trivial if $1 < |\pi| < n$. •

A partition is based on *SHEIS*s and associations among items in these itemsets. For this purpose, we need to synthesize association among items of every *SHEIS* of size greater than 1. We define synthesized association among items of a *SHEIS* as follows.

Definition 2.4.8. Let there are n different databases. Let $X \in SHEIS(D)$ such that $|X| \geq 2$. Synthesized association among the items of X is obtained by formula (2.4.19), denoted by $SA(X, D \mid \alpha, \gamma)$. •

To find goodness of a partition, we need to measure dispersion among items of a 2-item *SHEIS*. We define synthesized dispersion *SD* of an itemset of size 2 as follows.

Definition 2.4.9. Let there are n different databases. Let $X \in SHEIS(D)$ such that $|X| = 2$. Synthesized dispersion *SD* among items of X is given by $SD(X, D \mid \alpha, \gamma) = 1 - SA(X, D \mid \alpha)$. • (2.4.8)

We calculate synthesized associations corresponding to all *SHEIS*s of size greater than 1. In Example 2.4.3, we calculate *SA*s of itemsets in *SHEIS*(*D*).

Example 2.4.3. We continue here the discussion of Example 2.4.1. Synthesized associations among items of relevant *SHEIS*s are given as follows: $SA(\{a, b\}, D) = 0.77798$, $SA(\{c, d\}, D) = 0.79872$. We arrange *SHEIS*s of size greater than one in non-increasing order on synthesized association. The arranged *SHEIS*s are given as follows: $\{c, d\}$, $\{a, b\}$. Also, $FI((1, n, \alpha)) = \{a, b, c, d, g\}$. There exist two non-trivial partitions. They are given as follows: $\pi^{0.79872} = \{ \{a\},\{b\}, \{g\}, \{c, d\}\}$, and $\pi^{0.77798} = \{ \{g\}, \{a, b\}, \{c, d\}\}$. •

A frequent but not highly extracted item forms a singleton class. In above partitions, *g* is an example of such item.

2.4.5.1 Finding the best non-trivial partition

In Example 2.4.3, we observe the existence of two non-trivial partitions. At the levels of association 0.79872 and 0.77798, we get two non-trivial partitions. We would like to find the best partition among available non-trivial partitions. The best partition is based on the principle of maximizing the intra-class association and maximizing inter-class dispersion. Intra-class association and inter-class dispersion are defined as follows.

Definition 2.4.10. The intra-class association of a partition π at the level of association δ under the measure of synthesized association *SA* is defined as follows.

$$intra-class\ association(\pi^{\delta}) = \sum_{C\in\pi,\ |C|\geq 2} SA(C \mid \alpha, \gamma). \bullet \qquad (2.4.9)$$

Definition 2.4.11. The inter-class dispersion of a partition π at the level of association δ under the measure of synthesized dispersion *SD* is defined as follows.

$$inter-class\ dispersion(\pi^{\delta}) = \sum_{C_p, C_q \in \pi;\ p\neq q} \quad \sum_{a\in C_p, b\in C_q;\ \{a,b\}\in SHEIS} SD(\{a,b\} \mid \alpha, \gamma). \bullet \qquad (2.4.10)$$

We would like to define goodness measure of a partition for the purpose of finding the best partition among available non-trivial partitions. We define goodness measure of a partition as follows.

Definition 2.4.12. *goodness* (π^{δ}) = *intra- class association*(π^{δ}) + *inter-class dispersion*(π^{δ}) - $|\pi^{\delta}|$.• (2.4.11)

Better partition is obtained at higher goodness value. In Example 2.4.4, we calculate the goodness values of partitions obtained in Example 2.4.3.

Example 2.4.4. For the first partition, we get *intra-class association* $(\pi^{0.79872})$ = 0.79872, *inter-class dispersion* $(\pi^{0.79872})$ = 0.22202, and *goodness* $(\pi^{0.79872})$ = 1.02074. For the second partition, we get *intra-class association* $(\pi^{0.77798})$ = 1.57670, *inter-class dispersion*

($\pi^{0.77798}$) = 0.0, and *goodness* ($\pi^{0.77798}$) = 1.57670. The goodness value of the second partition is more than that of the first partition. Thus, the best non trivial partition of $FI(1, n, \alpha)$ is { {a, b}, {c, d}, {g} }, and obtained at level 0.77798. •

Let us look back at the databases in Example 2.4.1. In most of the transactions, whenever one item of {a, b} is there, then other item is also present. Also, items c and d appear together in most of the cases in a transaction. Thus, we find that partition $\pi^{0.77798}$ matches the ground reality better than partition $\pi^{0.79872}$ and the output of clustering is consistent with the transactions in the databases. It validates the clustering technique presented in this chapter. In the following lemma, we provide a set of necessary and sufficient conditions for the existence of a non-trivial partition.

Lemma 2.4.6. *Let there are n different databases. There exits a non-trivial partition of $FI(1, n, \alpha)$ if and only if there exists an itemset $X \in SHEIS(1, n, \alpha, \gamma)$ such that* (i) $|X| \geq 2$, *and* (ii) $SA(Y, D) \neq SA(Z, D)$, *for all* $Y, Z \in SHEIS(1, n, \alpha, \gamma)$, *and* $|Y|, |Z| \geq 2$.

Proof. We sort *SHEIS*s in non-increasing order on synthesized association. Let $SA(M, D)$ = *maximum* $\{SA(X, D): X \in HEIS(1, n, \alpha, \gamma)$, and $|X| \geq 2$ }. Before the itemset M, there does not exist any *SHEIS*. Thus, itemset M is trivially mutually exclusive with the previous *SHEIS*s. Due to condition (ii), there exists a partition at the level $SA(M, D)$. In addition, the partition is non-trivial due to condition (i). This non-trivial partition contains a single class M such that $|M| \geq 2$. The remaining classes of this partition are singleton. •

At two different levels of association δ_1, and δ_2 ($\neq \delta_1$), we may get the same partition.

Definition 2.4.13. Let $C \subseteq FI(1, n, \alpha)$, and $C \neq \phi$. Two partitions π^{δ_1} and π^{δ_2} are the same if the following statement is true: $C \in \pi^{\delta_1}$ if and only if $C \in \pi^{\delta_2}$, for $\delta_1 \neq \delta_2$. •

There are $2^{|S|}$ elements in the power set of S. Also, there are two trivial partitions for a non-null set. Thus, the number of distinct non-trivial partitions of a non-null set is always less than or equal to $2^{|S|} - 2$, for a non-null set S. In Lemma 2.4.7, we find the upper bound of number of non-trivial partitions.

Lemma 2.4.7. *Let there are n different databases. Then the number of distinct non-trivial partitions is less than or equal to* $|\{X: |X| \geq 2 \text{ and } X \in SHEIS(D)\}|$. *Equality holds if and only if the following conditions are true.*

(i) *There does not exist a* $X \in SHEIS(D)$, *for* $|X| \geq 3$.

(ii) $Y \cap Z = \phi$, *for all* $Y, Z \in SHEIS(D)$, *and* $|Y|, |Z| \geq 2$.

(iii) $SA(Y, D) \neq SA(Z, D)$, *for all* $Y, Z \in SHEIS(D)$, *and* $|Y|, |Z| \geq 2$.

Proof. We arrange *SHEIS*s in non-increasing order based on synthesized association, for all *SHEIS* of size greater than 1. Let the arranged *SHEIS*s be $X_1, X_2, \ldots, X_m$, for integer $m \geq 1$. There exists a partition at $SA(X_1)$, if conditions (ii) and (iii) are satisfied at $Y = X_1$ (as mentioned in Lemma 2.4.6). In general, there exists another partition at $SA(X_k, D)$, if conditions (ii) and (iii) are satisfied for $X_1, X_2, \ldots, X_{k-1}$. If X is a *SHEIS* then $Y \subset X$ is also a *SHEIS*, for $Y \neq \phi$. Two partitions could not exist at levels $SA(X, D)$ and $SA(Y, \text{D})$, since $Y \subset X$. Thus, condition (i) is necessary at the equality. The lemma follows. •

Corollary 2.4.3. *Let there are n different databases. The set of all non-trivial partitions of* $FI(1,n, \alpha)$ *is* $\{\pi^{SA(X, D)}: X \in SHEIS(D), |X| \geq 2, \text{ and } \pi^{SA(X, D)} \text{ exists}\}$. •

Based on Observation 2.4.1 and Corollary 2.4.3, one could obtain the difference in similarity between the proposed clustering technique and the existing technique as follows.

Definition 2.4.14. Let π be clustering of $FI(1, n, \alpha)$ at level δ. Let $X = \{x_1, x_2, \ldots, x_m\} \in SHEIS(D)$ such that $\delta = SA(X, D)$, for $|X| \geq 2$. The difference in similarity using measure A_2 is given by $DS(X, D) = A_2(\{x_1, x_2, \ldots, x_m\}, D) - minimum\{sim_2(x_i, x_j, D) : 1 \leq i < j \leq m\}$. • (2.4.12)

Using Algorithm 2.4.1, we obtain the synthesized association of each *SHEIS* of size greater than 1. We use this information for finding the best non-trivial partition of frequent items in multiple databases.

Algorithm 2.4.2. Best non-trivial partition (if it exists) of frequent items in multiple databases.

```
procedure BestPartition (m, S)
Input:
m: number of SHEISs of size greater than 1
S: array of SHEISs of size greater than 1
Output:
Best non-trivial partition (if it exists) of frequent items in multiple databases
01: arrange the elements of S in non-increasing order on synthesized association;
02: let $S(m+1) = \phi$; let $SA(S(m+1), D) = 0$;
03: if ($m = 1$) then
04:   form a class using items in $S(1)$;
05:   for each item in ($FI(1, n, \alpha) - S(1)$) do
06:     form a singleton class;
07:   end for
08:   a partition is formed at level $SA(S(1), D)$;
09:   return the partition;
10: end if
11: let $temp = \phi$; let $mExclusion = 0$;
12: for $i = 1$ to $m$ do
13:   if ($temp \cap S(i) \neq \phi$) then $mExclusion = 1$; end if
14:   if (($mExclusion = 0$) and ($SA(S(i), D) \neq SA(S(i+1), D)$)) then
15:     for $j = 1$ to $i$ do
16:       items in $S(j)$ form a class;
17:       $temp = temp \cup S(j)$;
18:     end for
19:     for each item in ($FI(1, n, \alpha) - temp$) do
20:       form a singleton class;
```

```
21:         end for
22:         store the classes formed and the level of partition as $SA(S(i), D)$;
23:       else if ($mExclusion$ = 1) then go to line 26; end if
24:       end if
25:    end for
26:    return the partition having maximum goodness;
end procedure
```

If *m* is equal to 1 then we have only one non-trivial partition. All the items of *SHEIS* form a class and each of the remaining frequent items forms a singleton class. The partition is formed at the level of synthesized association among items in *SHEIS*. The variable *temp* accumulates all the items in previous *SHEIS*s. The variable *mExclusion* is used to check the mutually exclusiveness among the current SHEIS and all the previous *SHEIS*s. Also, we need to check another condition whether the synthesized association of current *SHEIS* different than the synthesized association among items of the next *SHEIS*. The conditions for existence of a partition are checked at line 14. If a partition exists at the current level then the items in each of the previous *SHEIS*s form a class. Each of the remaining items forms a singleton class. If the current *SHEIS* is not mutually exclusive with each of the previous *SHEIS*s of *S* then no more partition exists. Some more useful explanations are given in Lemma 2.4.8.

Line 1 takes $O(m\times\log(m))$ time. The for-loop at line 5 takes $O(|FI|)$ time. The for-loop at line 12 repeats *m* times. Let the average size of a class be *p*. The for-loop at line 15 takes $O(m \times p)$ time. The for-loop at line 19 takes $O(|FI|)$ time. Each of the statements at line 13 and 17 takes $O(p \times |FI|)$ time for a single execution of line 11. Thus, the time complexity of the program segment lines 15-18 is $O(m \times p \times |FI|)$, for each iteration of for-loop at line 12. Also, the for-loop at 19 takes $O(|FI|)$ time. Thus, the time complexity of program segment 12-25 is $O(m^2 \times p \times |FI|)$. Therefore, the time complexity of algorithm *BestPartition* is $O(m^2 \times p \times |FI|)$.

Lemma 2.4.8. *Algorithm BestPartition works correctly.*

Proof: We arrange *SHEIS*s of size greater than one in non-increasing order on synthesized association. Existence of only one SHEIS of size greater than one implies the existence of only one non-trivial partition (as mentioned in Lemma 2.4.6). By default, it will be the best non-trivial partition.

Let there are m *SHEIS*s of size greater than one, for an integer $m \geq 2$. Let the arranged *SHEIS*s be $X_1, X_2, \ldots, X_m$. Then we need to check for the existence of partitions only at levels $SA(X_i, D)$, for $i = 1, 2, \ldots, m$ (as mentioned in Corollary 2.4.3). So, we have used a for loop at line 12 to check for partitions at m discrete levels $SA(X_i, D)$, for $i = 1, 2, \ldots, m$. At the j-th iteration of for-loop at line 12, we check the mutually exclusiveness of the itemset X_j with itemset X_k, for $k = 1, 2, \ldots, j-1$. If each of $X_1, X_2, \ldots$, and X_{j-1} is mutually exclusive with X_j and $SA(X_j, D) = SA(X_{j+1}, D)$ then the current partition is not recorded. At this point, we are not sure whether X_{j+1} is mutually exclusive with X_k, for $k = 1, 2, \ldots, j$. At the next iteration for $i = j+1$, the partition is recorded (if it exists) and it contains the itemsets $X_1, X_2, \ldots, X_{j+1}$ as classes and each of the remaining frequent items forms a singleton class, provided $SA(X_{j+1}, D) > SA(X_{j+2}, D)$. At the j-th iteration of for-loop at line 12, if each of $X_1, X_2, \ldots$, and X_{j-1} is not mutually exclusive with X_j then no more partition exists. Thus, the algorithm works correctly. •

2.4.5.2 Analysis of error

To evaluate our proposed clustering technique we have measured the amount of error occurred in an experiment. The clustering is based on *SHEIS*s extracted from D_i, for $i = 1, 2, \ldots, n$. Let the number of *SHEIS*s be m. Supports of all *SHEIS*s have been synthesized during the clustering process. There are several ways one could define error of an experiment. We have defined following two types of error of an experiment.

Average Error: $$\mathrm{AE}(D, \alpha, \gamma) = \frac{1}{m} \sum_{i=1}^{m} |SS(X_i, D) - S(X_i, D)| \qquad (2.4.13)$$

Maximum Error: $$\mathrm{ME}(D, \alpha, \gamma) = maximum\{|SS(X_i, D) - S(X_i, D)|, \text{for } i = 1, 2, \ldots, m\} \qquad (2.4.14)$$

Notations $SS(X, D)$ and $S(X, D)$ denote synthesized support and (apriori) support of X in D. Error of an experiment is relative to the number of transactions, number of items, and the length of a transaction in local databases. Thus, the error of the experiment needs to be expressed along with the average number of transactions (*ANT*), average number of items (*ANI*), and the average length of a transaction (*ALT*) in D.

2.4.6 Experimental results

We have carried out several experiments to study the effectiveness of our approach. All the experiments have been implemented on a 2.8 GHz Pentium D dual processor with 512 MB of memory using visual C++ (version 6.0) software. We present the experimental results using three real databases. The database *retail* [34] is obtained from an anonymous Belgian retail supermarket store. The database *mushroom* is also available in [34]. The database *ecoli* is a subset of *ecoli database* [77] and has been processed for the purpose of conducting experiments. For this purpose, we have omitted non-numeric fields of *ecoli database*. We present some characteristics of these databases in Table 2.4.1.

Table 2.4.1. Database characteristics

Database (*DB*)	*NT*	*ALT*	*AFI*	*NI*
retail	88,162	11.305755	99.673800	10,000
mushroom	8,124	24.000000	1624.800000	120
ecoli	336	7.000000	25.565217	92

Let *NT*, *ALT*, *AFI*, and *NI* denote the number of transactions, average length of a transaction, average frequency of an item and number of items in the data source, respectively. Each of the above databases is divided into 10 databases for the purpose of conducting experiments. The databases obtained from *retail*(R), *mushroom*(M) and *ecoli*(E) are named as R_i, M_i and E_i, respectively, for $i = 0, 1, \ldots, 9$. The databases R_i, M_i

and E_i are called input databases, for $i = 0, 1, \ldots, 9$. Some characteristics of these input databases are presented in Table 2.4.2. We have carried out several experiments to study the effectiveness of our approach for clustering the frequent items. We present results of the experiments based on the above databases. We observe that a partition of frequent items might not exist for some combination of input databases, α, and γ.

2.4.6.1 Overall output

(a) *For experiment with retail*: The set of frequent items in different databases are given as follows: *FI*(0, 9, 0.1) = { {0}, {1}, {2}, {3}, {4}, {5}, {6}, {7}, {8}, {9}, {32}, {38}, {39}, {41}, {48}}. *SHEIS*s of size greater than 1 along with their synthesized associations are given as follows: {39, 48} (0.443690), {39, 41, 48} (0.393977), {39, 41} (0.263936), {41, 48} (0.251072), {38, 39} (0.181348). The best non-trivial partition is given as follows: $\pi^{0.44369}$ = {{0}, {1}, {2}, {3}, {4}, {5}, {6}, {7}, {8}, {9}, {32}, {38}, {41}, {39, 48}}.

(b) *For experiment with mushroom*: The set of frequent items in different databases are given as follows : *FI*(0, 9, 0.5) = {{1}, {2}, {3}, {6}, {7}, {9}, {10}, {11}, {23}, {24}, {28}, {29}, {34}, {36}, {37}, {38}, {39}, {48}, {52}, {53}, {54}, {56}, {58 }, {59}, {61}, {63}, {66}, {67}, {76}, {85}, {86}, {90}, {93}, {94}, {95}, {99}, {101}, {102 }, {110}, {114}, {116}, {117}, {119} }. Top 10 *SHEIS*s of size greater than 1 along with their synthesized associations are given as follows: {34, 90} (0.999957), {34, 86} (0.999458), {34, 85} (0.995638), {34, 36, 85} (0.989711), {34, 36, 90} (0.987932), {34, 85, 90} (0.980130), {34, 36, 86} (0.977787), {34, 86, 90} (0.977439), {34, 85, 86} (0.968257), {85, 86} (0.962741).

Table 2.4.2. Input database characteristics

DB	*NT*	*ALT*	*AFI*	*NI*	*DB*	*NT*	*ALT*	*AFI*	*NI*
R_0	9000	11.24389	12.07001	8384	R_5	9000	10.85578	16.70977	5847
R_1	9000	11.20922	12.26541	8225	R_6	9000	11.20011	17.41552	5788
R_2	9000	11.33667	14.59657	6990	R_7	9000	11.15511	17.34554	5788
R_3	9000	11.48978	16.66259	6206	R_8	9000	11.99711	18.69032	5777
R_4	9000	10.95678	16.03953	6148	R_9	7162	11.69199	15.34787	5456
M_0	812	24.00000	295.27272	66	M_5	812	24.00000	221.45454	88
M_1	812	24.00000	286.58823	68	M_6	812	24.00000	216.53333	90
M_2	812	24.00000	249.84615	78	M_7	812	24.00000	191.05882	102
M_3	812	24.00000	282.43478	69	M_8	812	24.00000	229.27058	85
M_4	812	24.00000	259.84000	75	M_9	816	24.00000	227.72093	86
E_0	33	7.00000	4.62000	50	E_5	33	7.00000	3.91525	59
E_1	33	7.00000	5.13333	45	E_6	33	7.00000	3.50000	66
E_2	33	7.00000	5.50000	42	E_7	33	7.00000	3.91525	59
E_3	33	7.00000	4.81250	48	E_8	33	7.00000	3.39706	68
E_4	33	7.00000	3.39706	68	E_9	39	7.000000	4.55000	60

The transactions in different databases are highly similar, in the sense that two transactions in a database have many common items. The best non-trivial partition is given as follows. $\pi^{0.999957}$ = {{1}, {2}, {3}, {6}, {7}, {9}, {10}, {11}, {23}, {24}, {28}, {29}, {36}, {37}, {38}, {39}, {48}, {52}, {53}, {54}, {56}, {58 }, {59}, {61}, {63}, {66}, {67}, {76}, {85}, {86}, {93}, {94}, {95}, {99}, {101}, {102 }, {110}, {114}, {116}, {117}, {119}, {34, 90}}.

(c) *For experiment with ecoli*: The frequent items in different databases are given as follows. *FI*(0, 9, 0.1) = {{0}, {20}, {23}, {24}, {25}, {26}, {27}, {28}, {29}, {30}, {31}, {32}, {33}, {34}, {35}, {36}, {37}, {38}, {39}, {40}, {41}, {42}, {43}, {44}, {45}, {46}, {47}, {48}, {49}, {50}, {51}, {52}, {54}, {56}, {57}, {58}, {59}, {61}, {63}, {64}, {65}, {66}, {67}, {68}, {69}, {70}, {71}, {72}, {73}, {74}, {75}, {76}, {77}, {78}, {79}, {80}, {81}, {92}, {100}}. Top 10 *SHEIS*s of size greater than 1 along with their synthesized associations are given as follows. {48, 50}(0.803571), {37, 48, 50}(0.232143), {48, 50, 52}(0.229167), {40, 48, 50}(0.226190), {44, 48, 50}(0.226190), {46, 48, 50}(0.193452), {37, 48}(0.190476), {44, 50}(0.190476), {48, 50, 51}(0.190476), {40, 48}(0.184524). The best non-trivial partition is given as follows. $\pi^{0.803571}$ = {{0}, {20}, {23}, {24}, {25}, {26}, {27}, {28}, {29}, {30}, {31}, {32}, {33}, {34}, {35}, {36}, {37}, {38}, {39}, {40}, {41}, {42}, {43}, {44}, {45}, {46}, {47}, {49}, {51}, {52}, {54}, {56}, {57}, {58}, {59}, {61}, {63}, {64}, {65}, {66}, {67}, {68}, {69}, {70}, {71}, {72}, {73}, {74}, {75}, {76}, {77}, {78}, {79}, {80}, {81}, {92}, {100}, {48, 50}}.

2.4.6.2 Synthesis of highly extracted itemsets

(a) *For experiment with retail*: In Table 2.4.3, we present errors in synthesizing *HEIS*s. *HEIS*s {32}, {38}, {39}, {48}, {38, 39} and {39, 48} are extracted from every branch database. Thus, the error in synthesizing support of each of the above HEISs is zero.

Table 2.4.3. Error in synthesizing supports of HEISs at α = 0.1 and γ = 0.6

HEIS *X*	$\lvert SS(X, R) - S^R(X, R)\rvert$	*HEIS* *X*	$\lvert SS(X, R) - S(X, R)\rvert$	*HEIS* *X*	$\lvert SS(X, R) - S(X, R)\rvert$	*HEIS X*	$\lvert SS(X, R) - S(X, R)\rvert$
{0}	0.002925	{5}	0.001948	{32}	0.000000	{38, 39}	0.000000
{1}	0.002064	{6}	0.001949	{38}	0.000000	{39, 41}	0.003616
{2}	0.001945	{7}	0.002006	{39}	0.000000	{39, 48}	0.000000
{3}	0.002030	{8}	0.002016	{41}	0.003470	{41, 48}	0.003721
{4}	0.002034	{9}	0.001654	{48}	0.000000	{39, 41, 48}	0.003781

(b) *For experiment with mushroom*: The transactions in local databases are highly similar, in the sense that two transactions in a database have many common items. Thus, we get many frequent itemsets in local databases even at a high value of α. The errors in synthesizing some *HEIS*s are presented in Table 2.4.4.

Table 2.4.4. Error in synthesizing supports of selected *HEIS*s at $\alpha = 0.5$ and $\gamma = 0.7$ (top 15)

HEIS X	$\lvert SS(X, M) - S(X, M)\rvert$	*HEIS X*	$\lvert SS(X, M) - S(X, M)\rvert$	*HEIS X*	$\lvert SS(X, M) - S(X, M)\rvert$
{24, 85}	0.003304	{85, 90}	0.001278	{34, 39}	0.0010866
{24, 86}	0.003282	{53}	0.001207	{67}	0.001049
{24, 90}	0.002204	{53, 90}	0.001107	{34, 67}	0.001049
{85, 86}	0.001295	{53, 85}	0.001107	{39}	0.001081
{86, 90}	0.001290	{53, 86}	0.001007	{34, 90}	0.001098

(c) *For experiment with ecoli*: For the local databases generated from *ecoli*, the average size of databases, the average length of transactions in different databases are smaller. Thus, the error of synthesizing a *HEIS* is comparatively higher. The errors in synthesizing some *HEIS*s are presented in Table 2.4.5.

Table 2.4.5. Error in synthesizing supports of selected *HEIS*s at $\alpha = 0.1$ and $\gamma = 0.6$ (top 15)

HEIS X	$\lvert SS(X, E) - S(X, E)\rvert$	*HEIS X*	$\lvert SS(X, E) - S(X, E)\rvert$	*HEIS X*	$\lvert SS(X, E) - S(X, E)\rvert$
{48,50, 51}	0.003695	{44}	0.002381	{48, 51}	0.001488
{46,48, 50}	0.003471	{46, 48}	0.002380	{37}	0.001390
{40,48, 50}	0.003273	{44, 48, 50}	0.002083	{52}	0.001387
{37,48, 50}	0.002976	{48, 50, 52}	0.001785	{40, 50}	0.001291
{50, 51}	0.002381	{40}	0.001785	{50, 52}	0.001261

2.4.7.3 Error of the experiment

The errors of different experiments are presented in Table 2.4.6. If the average number of transactions in different databases increases then the average error of synthesizing *HEIS*s is likely to decrease, provided other two parameters remain constant. If the average length of transactions in different databases increases then the average error of synthesizing *HEIS*s is likely to increase, provided other two parameters remain constant. Lastly, if the average number of items in different databases increases then the error of synthesizing *HEIS*s is likely to decrease, provided other two parameters remain constant.

Table 2.4.6. Error of the experiments

DB	α	γ	AE (*ANT*, *ALT*, *ANI*)	ME (*ANT*, *ALT*, *ANI*)
$\bigcup_{i=0}^{9} R_i$	0.1	0.7	0.00120 (8816.2, 11.30576, 5882.1)	0.00293 (8816.2, 11.30576, 5882.1)
$\bigcup_{i=0}^{9} M_i$	0.5	0.7	0.00125 (812.4, 24.00000, 80.7)	0.003304 (812.4, 24.00000, 80.7)
$\bigcup_{i=0}^{9} E_i$	0.5	0.7	0.00133 (33.6, 7.00000, 56.5)	0.00370 (33.6, 7.00000, 56.5)

2.4.7.4 Average error versus γ

We have conducted experiments to study the behaviour of AE over different γs. In general, we find that AE decreases as γ increases. The purpose of the experiment would be lost if we keep γ at a high value, since the number of *HEIS*s also decreases as γ increases. Thus, a decision based on *HEIS*s would have low validity at a high value of γ. In Figures 2.4.1, 2.4.2 and 2.4.3, we present graphs of AE versus γ for three experiments. From the figures presented below, we find that the value of γ around 0.7 would have been a good choice for clustering frequent items in different databases.

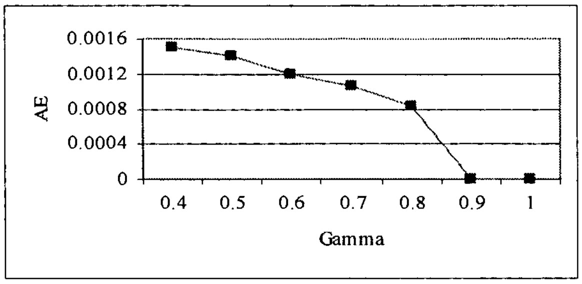


Figure 2.4.1. AE versus γ at $\alpha = 0.1$ (*retail*)

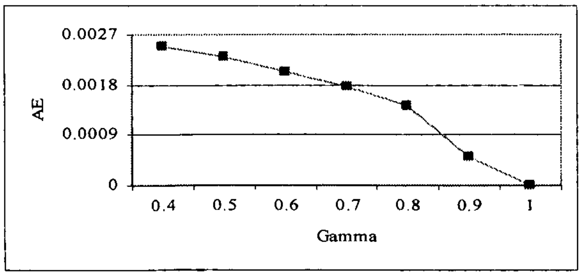


Figure 2.4.2. AE versus γ at $\alpha = 0.5$ (*mushroom*)

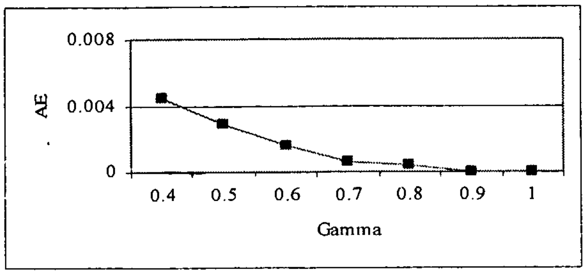


Figure 2.4.3. AE versus γ at $\alpha = 0.1$ (*ecoli*)

2.4.7.5 Average error versus α

We have also conducted experiments to study the behaviour of AE over different αs. In general, we find that AE increases as α increases. As α increases, the more number of databases would fail to extract an itemset. Thus, the error of synthesizing an itemset is likely to increase as α increases. In Figures 2.4.4, 2.4.5 and 2.4.6, we present graphs of AE versus α for three experiments.

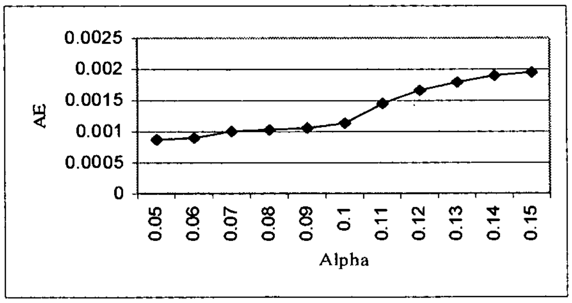


Figure 2.4.4. AE versus α at $\gamma = 0.7$ (*retail*)

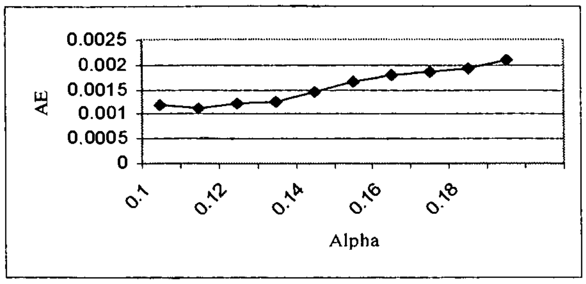


Figure 2.4.5. AE versus α at $\gamma = 0.7$ (*mushroom*)

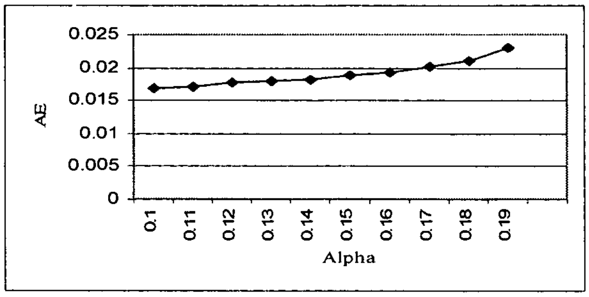


Figure 2.4.6. AE versus α at $\gamma = 0.7$ (*ecoli*)

2.4.7.6 Clustering time versus number of databases

We have also studied the behaviour of clustering time required over the number of databases used in an experiment. As the number of databases increases, the number of frequent itemsets also increases. In general, we find that clustering time increases as the number of databases increases. In Figures 2.4.7, 2.4.8 and 2.4.9, we present graphs of clustering time versus number of databases for three experiments.

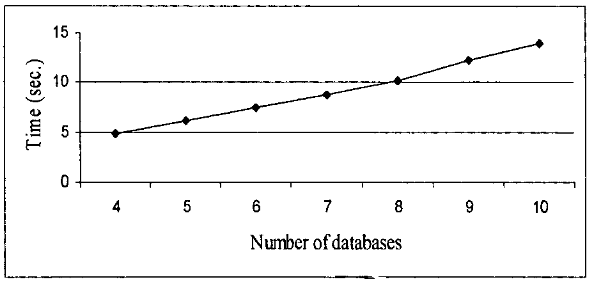


Figure 2.4.7. Clustering time versus number of databases at $\alpha = 0.1$ and $\gamma = 0.7$ (*retail*)

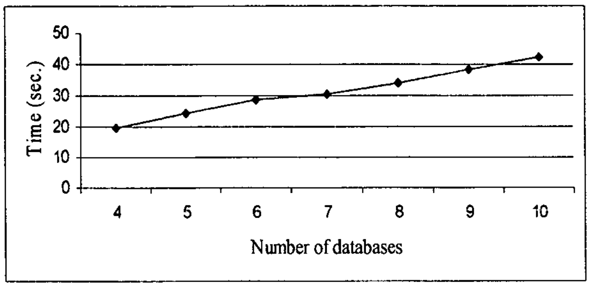


Figure 2.4.8. Clustering time versus number of databases at $\alpha = 0.5$ and $\gamma = 0.7$ (*mushroom*)

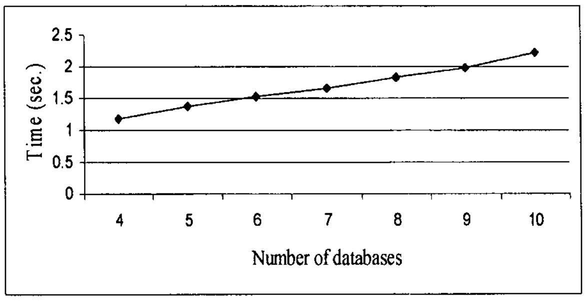


Figure 2.4.9. Clustering time versus number of databases at $\alpha = 0.5$ and $\gamma = 0.7$ (*ecoli*)

2.4.7.7 Comparison with existing technique

The proposed clustering technique is likely to enhance the accuracy of clustering process, if the clustering is performed at a level δ such that δ is a synthesized association of a class containing more than 2 frequent items. In each of the Tables 2.4.7, 2.4.8 and 2.4.9, we present an example of clustering that achieves higher level of accuracy.

Table 2.4.7. A sample clustering of frequent items in multiple databases (*retail*)

α	γ	Clustering	δ(existing approach)	δ(proposed approach)	*DS*
0.1	0.7	$\{\{0\}, \{1\}, \{2\}, \{3\}, \{4\}, \{5\}, \{6\}, \{7\}, \{8\}, \{9\}, \{32\}, \{38, 39\}, \{39, 41, 48\}\}$	0.251072	0.393977	0.142905

Table 2.4.8. A sample clustering of frequent items in multiple databases (*mushroom*)

α	γ	clustering	δ(existing approach)	δ(proposed approach)	*DS*
0.5	0.7	{{1}, {2}, {3}, {6}, {7}, {9}, {10}, {11}, {23}, {24}, {28}, {29}, {37}, {38}, {39}, {48}, {52}, {53}, {54}, {56}, {58 }, {59}, {61}, {63}, {66}, {67}, {76}, {86}, {90}, {93}, {94}, {95}, {99}, {101}, {102 }, {110}, {114}, {116}, {117}, {119}, {34, 90}, {34, 86}, {34, 36, 85} }	0.842935	0.989711	0.146776

Table 2.4.9. A sample clustering of frequent items in multiple databases (*ecoli*)

α	γ	clustering	δ(existing approach)	δ(proposed approach)	*DS*
0.1	0.7	{{0}, {20}, {23}, {24}, {25}, {26}, {27}, {28}, {29}, {30}, {31}, {32}, {33}, {34}, {35}, {36}, {38}, {39}, {40}, {41}, {42}, {43}, {44}, {45}, {46}, {47}, {49}, {51}, {52}, {54}, {56}, {57}, {58}, {59}, {61}, {63}, {64}, {65}, {66}, {67}, {68}, {69}, {70}, {71}, {72}, {73}, {74}, {75}, {76}, {77}, {78}, {79}, {80}, {81}, {92}, {100}, {37, 48, 50}}	0.178571	0.232143	0.053572

2.4.8 Conclusion

Clustering relevant objects is an important task of many decision support systems. We have observed that existing clustering technique might cluster frequent items in multiple databases with low accuracy. We propose a new technique for clustering frequent items in multiple databases. It clusters frequent items in multiple databases with higher degree of accuracy.

The main problem with existing clustering technique is that it might not be able to estimate similarity among items in a class with high accuracy. Thus, it might fail to cluster frequent items with higher accuracy level. The experimental results show that the proposed clustering technique is effective and promising.

Chapter 2.5

Conclusion

Recognizing patterns in multiple databases is an important theme of activities. Many important decisions could be based on analysis of patterns in multiple databases. In Part 2, we have dealt with the following patterns in multiple databases: exceptional frequent itemset, heavy association rule, exceptional association rule, high-frequent association rule, and clustering items in multiple databases.

In Chapter 2.2, we have identified the shortcomings of existing concept of global exceptional pattern and proposed a definition of a global exceptional frequent itemset. We have designed an algorithm to identify global exceptional patterns in multiple databases. Also, we have introduced the notion of exceptional sources for a global exceptional frequent itemset. The proposed algorithm identifies global exceptional frequent itemsets and their exceptional sources in multiple databases.

In Chapter 2.3, we present an algorithm for synthesizing three important patterns in multiple databases viz., heavy association rule, high-frequent association rule, and exceptional association rule. It also provides a better solution for synthesizing high-frequent association rules in multiple databases.

In Chapter 2.4, we propose a new technique of clustering frequent items in multiple databases. It clusters frequent items in multiple databases with higher degree of accuracy as compared to the existing approaches. The main problem with an existing clustering technique is that it might not be able to estimate similarity among items in a class effectively. The experimental results show that the proposed clustering technique is effective and promising.

Part 3

Developing better multi-database mining applications

Chapter 3.1

Introduction

The sole purpose of Part 3 is to develop better multi-databases mining applications. We have discussed a number of strategies to enhance the efficiency of a multi-database mining system. The efficiency of a multi-database mining application could be enhanced by choosing an appropriate multi-database mining model, an appropriate pattern synthesizing technique, a better pattern representation technique, and an efficient algorithm for solving the problem. In Part 3, we have made the following contributions.

- We propose a new technique, called pipelined feedback model (PFM), for mining multiple databases.
- We propose a technique, called antecedent consequent pair (ACP) coding, for representing rulebases corresponding to different databases with space efficiency. It enables us to incorporate more association rules for synthesizing global patterns or decision-making activities.
- We propose an index structure to access the coded association rules conveniently.
- We prove that ACP coding represents rulebases using the least amount of storage space in comparison to any other rulebase representation technique.
- We propose a technique for storing rulebases corresponding to different databases in the secondary storage.
- We propose an algorithm for clustering frequent items in multiple databases based on measure of association A_2 and multi-database mining technique PFM.
- A model of mining global patterns of select items in multiple databases is proposed.
- A measure of overall association (OA), between two items in a database is proposed.

- An algorithm is designed based on *OA* for the purpose of grouping frequent items in multiple databases.

Chapter 3.2

Mining multiple large databases

Many large companies operate from a number of branches located at different geographical regions. Each branch might collect data continuously and data get stored locally. Thus, the collection of all branch databases might be large. Many decisions of a multi-branch company are based on data stored over the branches. The challenges involve in making good quality of decisions based on large volume of data distributed over the branches. It creates not only risks but also opportunities. One of the risks might be significant amount of investment on hardware and software to deal with multiple large databases. Our objective is to provide good quality of knowledge by minimizing the risks.

The first question comes to our mind whether a traditional data mining technique [13], [39] could provide a good solution in dealing with multiple large databases. To apply a traditional data mining technique one needs to amass all the branch databases together. A traditional data mining technique might not provide a good solution due to the following reasons.

- The company might have to invest heavily on hardware and software to deal with a large volume of data.
- A single computer might take unreasonable amount of time to mine a huge amount of data.
- It might be difficult to identify local patterns if a traditional data mining technique is applied on the entire database.

Thus, a traditional data mining technique might not be suitable in this situation. So, it is a different problem. Hence, it is required to be dealt with in a different way. In this situation local pattern analysis [91] could be a solution. Under this model of mining multiple databases, each branch requires to mine its database using a traditional data mining technique. Afterwards, each branch is required to forward the pattern base to the central office. Then the central office would process the pattern bases collected from different branches. Due to the following reason, the local pattern analysis alone might not be a judicious choice for mining multiple large databases.

- A synthesized global pattern might differ considerably from true global patterns.
- The process of mining current database is independent of patterns extracted from the previous databases.

For the purpose of mining multiple databases, one could apply *partition algorithm* (PA) proposed by Savasere et al. [66]. The algorithm was designed to mine a very large database by partitioning. The algorithm works as follows. It scans the database twice. The database is divided into disjoint partitions, where each partition is small enough to fit in memory. In a first scan, the algorithm reads each partition and computes locally frequent itemsets in each partition using apriori algorithm [13]. In the second scan, the algorithm counts the supports of all locally frequent itemsets toward the complete database. In this case, each local database could be considered as a partition. Though partition algorithm mines frequent itemsets exactly, it is an expensive solution to mining multiple large databases, since each local database is required to scan twice.

There are two benefits of PFM. Firstly, it improves significantly the accuracy of mining multiple large databases as compared to local pattern analysis. Secondly, it scans each local database only once.

For mining multiple databases, there are three situations: (i) Each of the local databases is small, so that a single database mining technique (SDMT) could mine the union of all databases. (ii) At least one of the local databases is large, so that a SDMT could mine every local database, but fail to mine the union of all local databases. (iii) At least one of the local databases is very large, so that a SDMT fails to mine it. We face challenges to handle the cases (ii) and (iii). The challenges are posed to us due to large size of some local databases.

A multi-database mining technique (MDMT) using local pattern analysis could be viewed as a two-step process M+S, explained as follows.

- Mine each local database using a SDMT by following a model M (Step 1)
- Synthesize patterns using an algorithm S (Step 2)

We use notation MDMT: M+S to represent above multi-database mining technique. We propose a MDMT that improves the quality of both synthesized patterns and analysis of local patterns. Our algorithm could handle the cases (ii) and (iii) reasonably well, and it requires mining each local database only once.

The rest of the chapter is organized as follows. We discuss related work and define the problem in Section 3.2.2. We propose a model for mining multiple databases in Section 3.2.3. We define error of an experiment in Section 3.2.4. In Section 3.2.5, we provide experimental results.

3.2.2 Problem definition

Consider a multi-branch company that operates from n branches. Let D_i be the database corresponding to the i-th branch, for $i = 1, 2, \ldots, n$. Let D be the union of all branch databases. Before presenting the proposed model, we shall first study work related to this issue.

3.2.2.1 Related work

In Section 2.4.2.1.2, we have provided a survey of existing multi-database mining techniques.

3.2.2.2 Our approach

We need to process local databases as they may not be at the right state for the mining task. Various data preparation techniques [65] like data cleaning, data transformation, data integration, and data reduction are applied to branch databases. We get processed database corresponding to (original) local database. Then we keep all the data that are relevant to data mining applications. Using a relevance analysis, one could detect outlier data [51] and store in a separate storage. After removing outlier data from a processed database we get desired data warehouse and the data in a data warehouse become ready for the mining task. Let W_i be the data warehouse corresponding to the i-th branch, for $i = 1, 2, \ldots, n$. Then the local patterns for the i-th branch are extracted from W_i, for $i = 1, 2, \ldots, n$. We mine each data warehouse using a SDMT. In Figure 3.2.1, we present a new model of mining multiple databases [8].

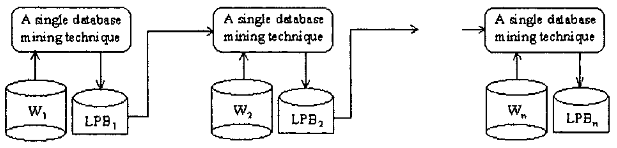

A single database mining technique
A single database mining technique
A single database mining technique
W1
LPB1
W2
LPB2
Wn
LPBn

Figure 3.2.1. Pipelined feedback model (PFM) of mining multiple databases

In PFM, W_1 is mined using a SDMT and local pattern base LPB_1 is extracted. While mining W_2, all the patterns in LPB_1 are extracted irrespective of their values of interestingness measures like, minimum support and minimum confidence. Apart from these patterns, some new patterns that satisfy user-defined threshold values of interestingness measures are also extracted. In general, while mining W_i, all the patterns in W_{i-1} are mined irrespective of their values of interestingness measures, and some new patterns that satisfy user-defined threshold values of interestingness measures, for $i = 2, 3, \ldots, n$. Due to this nature of mining each data warehouse, the technique is called a feedback model. Thus, $|LPB_{i-1}| \leq |LPB_i|$, for $i = 2, 3, \ldots, n$. There are $n!$ arrangements of pipelining for n databases. All the arrangements of data warehouses might not produce the same mining result. If the number of local patterns increases, we get more accurate global patterns and a better analysis of local patterns. An arrangement of data warehouses would produce near optimal result if $|LPB_n|$ is a maximal. Let $size(W_i)$ be the size of W_i (in bytes), for $i = 1, 2, \ldots, n$. We shall follow the following rule of thumb regarding the arrangements of data warehouses for the purpose of mining. The number of patterns in W_{i-1} is greater than or equal to the number of patterns in W_i, if $size(W_{i-1}) \geq size(W_i)$, for $i = 2, 3, \ldots, n$. For the purpose of increasing number of local patterns, W_{i-1} precedes W_i in the pipelined arrangement of mining data warehouses if $size(W_{i-1}) \geq size(W_i)$, for $i = 2, 3, \ldots, n$. Finally, we analyze the patterns in LPB_1, LPB_2, ..., and LPB_n for synthesizing global patterns, or analyzing local patterns.

For synthesizing global patterns in D we discuss here a simple pattern synthesizing (SPS) algorithm. Without loss of generality, let the itemset X be extracted from first m databases, for $1 \leq m \leq n$. Then synthesized support of X in D could be obtained as follows.

$$supp_s(X, D) = \frac{1}{\sum_{i=1}^{n} |D_i|} \times \sum_{i=1}^{m} [\, supp_a(X, D_i) \times |D_i| \,] \tag{3.2.1}$$

3.2.3 Mining multiple databases

In this section, we present a new algorithm for mining multiple databases [8]. The algorithm is based on the pipelined feedback model discussed in Section 3.2.2.

Algorithm 3.2.1. Mine multiple data warehouses using pipelined feedback model.

```
procedure PipelinedFeedbackModel ($W_1, W_2, \ldots, W_n$)
Input: $W_1, W_2, \ldots, W_n$
Output: local pattern bases
01: for $i = 1$ to $n$ do
02:   if $W_i$ does not fit in memory then
03:     partition $W_i$ into $W_{i1}, W_{i2}, \ldots$, and $W_{ip_i}$;
04:   else let $W_{i1} = W_i$;
05:   end if
06: end for
```

```
07: sort data warehouses on size in non-increasing order and the data warehouses are
    renamed as $DW_1, DW_2, \dots, DW_N$, where $N = \sum_{i=1}^{n} p_i$;
08: let $LPB_0 = \phi$;
09: for $i = 1$ to $N$ do
10:     mine $DW_i$ using a SDMT with input $LPB_{i-1}$;
11: end for
12: return $LPB_1, LPB_2, \dots, LPB_N$;
end procedure
```

In above algorithm, the usage of LPB_{i-1} during mining DW_i has been explained in Section 3.2.2.2. Once a pattern is extracted from a data warehouse, then it gets extracted from the remaining data warehouses. Thus, the algorithm *PipelinedFeedbackModel* improves synthesized patterns and analysis of local patterns significantly.

3.2.4 Error of an experiment

To evaluate MDMT: PFM+SPS, we need to measure the amount of error of the experiments. An experiment mines frequent itemsets in local databases using PFM, and then synthesizes global patterns using SPS algorithm. We need to find how the global synthesized support differs from the exact support of an itemset.

Let $LPB_n = \{X_1, X_2, \dots, X_m\}$. There are several ways one could define error of an experiment. We have defined following two types of error of an experiment.

1. Average Error (AE)

$$\text{AE}(D, \alpha) = \frac{1}{m} \sum_{i=1}^{m} \left| supp_a(X_i, D) - supp_s(X_i, D) \right| \tag{3.2.2}$$

2. Maximum Error (ME)

$$\text{ME}(D, \alpha) = maximum \left\{ \left| supp_a(X_i, D) - supp_s(X_i, D) \right|, i = 1, 2, \dots, m \right\} \tag{3.2.3}$$

$supp_a(X_i, D)$ is obtained by mining D using a traditional data mining technique, for $i = 1, 2, \dots, m$. $supp_s(X_i, D)$ is obtained by SPS, for $i = 1, 2, \dots, m$.

3.2.5 Experiments

We have carried out several experiments to study the effectiveness of our approach. All the experiments have been implemented on a 2.8 GHz Pentium D dual core processor with 512 MB of memory using visual C++ (version 6.0) software. We present experimental results using synthetic database *T10I4D100K* (*T*) [34] and two real databases *retail* (*R*) [34] and *BMS-Web-Wiew-1* (*B*) [34]. We present some characteristics of these databases in Table 3.2.1.

Let *NT*, *AFI*, *ALT*, and *NI* denote the number of transactions, average frequency of an item, average length of a transaction, and number of items in a database, respectively.

Table 3.2.1. Database characteristics

Database	*NT*	*ALT*	*AFI*	*NI*
T	1,00,000	11.10228	1276.12413	870
R	88,162	11.30575	99.67380	10000
B	1,49,639	2.00000	155.71176	1922

Each of the above databases is divided into 10 databases for the purpose of carrying out experiments. The databases obtained from *T*, *R* and *B* are named as T_i, R_i, and B_i, respectively, for i = 0, 1, …, 9. The databases T_i, R_i, and B_i are called input databases (DBs), for i = 0, 1, …, 9. Some characteristics of these input databases are presented in the Table 3.2.2.

Table 3.2.2. Input database characteristics

DB	*NT*	*ALT*	*AFI*	*NI*	*DB*	*NT*	*ALT*	*AFI*	*NI*
T_0	10000	11.05500	127.65588	866	T_5	10000	11.13910	128.62702	866
T_1	10000	11.13330	128.41176	867	T_6	10000	11.10780	128.56250	864
T_2	10000	11.06700	127.64705	867	T_7	10000	11.09840	128.45376	864
T_3	10000	11.12260	128.43649	866	T_8	10000	11.08150	128.55568	862
T_4	10000	11.13670	128.74797	865	T_9	10000	11.08140	128.10867	865
R_0	9000	11.24389	12.07001	8384	R_5	9000	10.85578	16.70977	5847
R_1	9000	11.20922	12.26541	8225	R_6	9000	11.20011	17.41552	5788
R_2	9000	11.33667	14.59657	6990	R_7	9000	11.15511	17.34554	5788
R_3	9000	11.48978	16.66259	6206	R_8	9000	11.99711	18.69032	5777
R_4	9000	10.95678	16.03953	6148	R_9	7162	11.69199	15.34787	5456
B_0	14000	2.00000	14.94130	1874	B_5	14000	2.00000	280.00000	100
B_1	14000	2.00000	280.00000	100	B_6	14000	2.00000	280.00000	100
B_2	14000	2.00000	280.00000	100	B_7	14000	2.00000	280.00000	100
B_3	14000	2.00000	280.00000	100	B_8	14000	2.00000	280.00000	100
B_4	14000	2.00000	280.00000	100	B_9	23639	2.00000	472.78000	100

In Table 3.2.3, we present some outputs to show that the proposed technique improves significantly the mining results. We have performed experiments using other MDMTs on these databases for the purpose of comparing with MDMT: PFM+SPS.

Table 3.2.3. Error of the experiments at a given α

Database	*T10I4D100K*		*retail*		*BMS-Web-Wiew-1*	
α	0.05		0.11		0.19	
Error type	AE	ME	AE	ME	AE	ME
MDMT: RO+IEP	0.01218	0.03730	0.00516	0.05825	0.04823	0.14490
MDMT: RO+RS	0.01017	0.03612	0.00502	0.05755	0.02319	0.13490
MDMT: RO+ARS	0.00719	0.03599	0.00491	0.05730	0.02102	0.10514
MDMT: PFM+SPS	0.00321	0.03583	0.00484	0.05725	0	0
MDMT: RO+PA	0	0	0	0	0	0

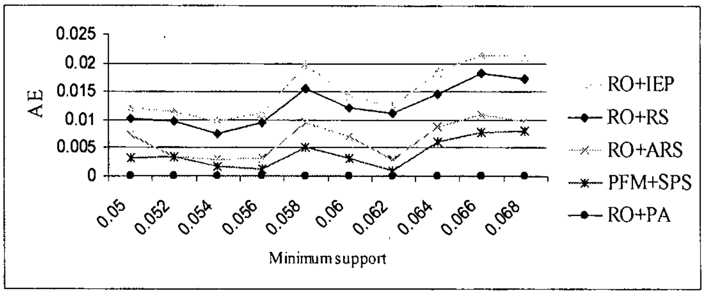


Figure 3.2.2. AE vs. α for experiments using database T

In the Figures 3.2.2, 3.2.3, and 3.2.4, we show average errors against different αs. From Figures 3.2.2, 3.2.3, and 3.2.4, one could conclude that AE normally increases as α increases. The number of databases reporting a pattern decreases as α increases. Thus, the AE of synthesizing patterns normally increases as α increases.

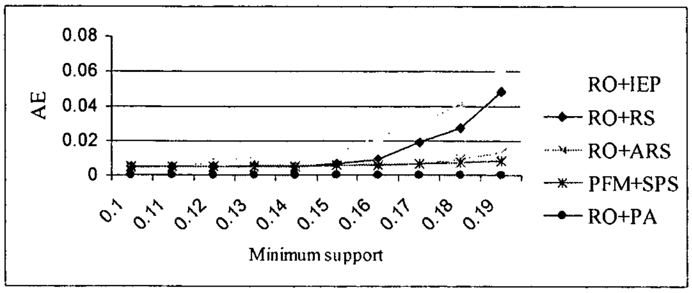


Figure 3.2.3. AE vs. α for experiments using database *R*

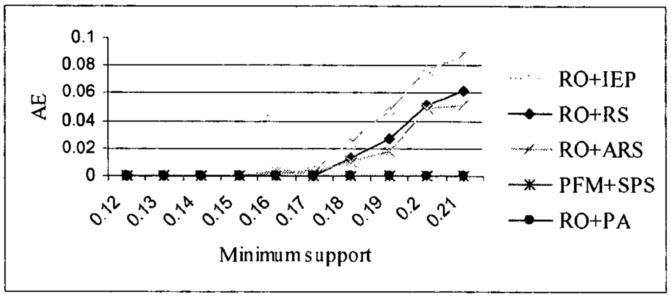


Figure 3.2.4. AE vs. α for experiments using database *B*

3.2.6 Conclusion

In this chapter, we present a new technique for mining multiple databases. It improves significantly the accuracy of mining multiple databases as compared to existing techniques that scan each database only once. The proposed technique could also be used for mining a large database by dividing it into sub-databases. MDMT: PFM+SPS is effective and promising.

Chapter 3.3

Enhancing quality of knowledge synthesized from multi-database mining

Many large organizations transact from multiple branches. The transactions made in a branch are stored locally. Thus, many multi-branch companies possess multiple databases. Consider a company that operates shopping malls from different places. These malls are open at least 12 hours a day. All the transactions made in a mall are stored locally. Thus, the company possesses multiple databases. A corporate decision based on data distributed over the branches requires handling multiple databases effectively. Most of the previous pieces of data mining work are based on a single database. Therefore, it is important to study data mining on multiple databases.

Consider a multi-branch company that operates from different locations. Each branch possesses a large database. Thus, the collection of all branch databases is very large. The first question comes to our mind whether a traditional data mining technique could deal with the multiple large databases. To apply a traditional data mining technique, one needs to amass all the branch databases together. A single computer might take unreasonable amount of time to process the entire database. Sometimes it might not be feasible to carry out the mining task. Another solution would be to employ parallel machines. But, it requires high investment on hardware and software. Moreover, it is difficult to identify local patterns if the mining technique is applied on the entire database. Thus, a traditional data mining technique is not suitable in this situation. So, it is a different problem. Hence, it is required to be dealt with in a different way. In this case, one could employ the model of local pattern analysis [91]. According to this model of mining, the branches are required to forward their local patterns to the central office for analysis and synthesis of patterns. The central office collects the local patterns and stores them for further analysis and synthesis. In particular, the company may need to identify the global association rulcs in the union of all databases. Let $X \rightarrow Y$ be an association rule extracted from a few databases. Then local pattern analysis might return approximate association rule $X \rightarrow Y$ in the union of all databases, since it may fail to get extracted from all the databases. Using a coding, discussed in this chapter, one could reduce further the values of minimum support and minimum confidence for extracting local association rules. Thus, the association rule $X \rightarrow Y$ might get extracted from more number of databases, since minimum support and minimum confidence are lowered further. In that case, the synthesized association rule $X \rightarrow Y$ might be more accurate than the earlier approach.

Multi-database mining has been recently recognized as an important research topic in the KDD community. Many multi-database mining applications often handle a large number of patterns. In multi-database mining applications, local patterns could be used in two ways. In the first category of applications, global patterns are synthesized from local patterns [5], [81], [89]. Synthesized global patterns could be used in various decision-making problems. In the second category of applications, various decisions are taken based on the local patterns in different databases [6], [83]. Thus, the available local patterns could play an important role in finding a solution to a problem. For a problem in the first category, the quality of a global pattern is influenced by the pattern synthesizing algorithm and the available local patterns. Also, we observe that a global pattern synthesized from local patterns might be approximate. At a given pattern synthesizing algorithm, one could enhance the quality of synthesized patterns by increasing the number of local patterns in a knowledge synthesizing process. For a problem in the second category, the quality of decision is based on the quality of measure used in the decision-making process. Again, the quality of measure is based on the correctncss of the measure and the available local patterns. For the purpose of clustering databases, Wu et al. [83] have proposed two such measures of similarity between two databases. For a given measure of decision-making, one could enhance the quality of decision by increasing the number of local patterns in the decision making process. Thus, the number of available local patterns plays a crucial role in building efficient multi-database mining applications. Thus, one could make a better decision if the available patterns are more. One could increase the number of local patterns by lowering further the user inputs, such as minimum support and minimum confidence, given to a data mining algorithm. More patterns could be stored in main memory by applying a space efficient pattern base representation technique. In this chapter, we present a coding, called ACP coding [4], for representing a set of association rules in different databases space efficiently. A similar technique [6] could also be applied for representing frequent itemsets in different databases space efficiently.

Consider a multi-branch company that transacts from n branches, for $n \geq 2$. Let D_i be the database corresponding to the i-th branch, for $i = 1, 2, ..., n$. Also, let D be the union of these databases. The data mining model adopted in this chapter for association rule is the support (supp)-confidence (conf) framework established by Agrawal et al. [11]. The set of association rules extracted from a database is called a *rulebase*. Before we present the problem, we introduce a few notations used frequently in this chapter. Let RB_i be the rulebase corresponding to database D_i at the *minimum support level* α and *minimum confidence level* β, for $i = 1, 2, ..., n$. Also, let RB be the union of rulebases corresponding to different databases. Many interesting algorithms have been reported on mining association rules in a database [13], [39], [66]. Let T be a technique for representing RB in main memory. Let φ and ψ denote the pattern synthesizing algorithm and computing resource used for a data mining application, respectively. Also, let $\xi(RB \mid T, \alpha, \beta, \varphi, \psi)$ denote the collection of synthesized patterns over RB at a given tuple $(T, \alpha, \beta, \varphi, \psi)$. The quality of synthesized patterns could be enhanced if the number of local patterns increases. Thus, quality of $\xi(RB \mid T, \alpha_1, \beta_1, \varphi, \psi) \leq$ quality of $\xi(RB \mid T, \alpha_2, \beta_2, \varphi, \psi)$, if $\alpha_2 < \alpha_1$ and $\beta_2 < \beta_1$. So, the problem of enhancing the quality of synthesized patterns boils down to the problem of designing a space efficient technique for representing rulebases corresponding to different databases.

As the frequent itemsets are the natural form of compression for association rules, the following reasons motivate us to compress association rules rather than frequent itemsets. Firstly, applications dealing with the association rules could be developed efficiently. Secondly, a frequent itemset might not generate any association rule at a given minimum confidence.

In this chapter, we present a space efficient technique to represent *RB* in main memory. Let $SP^{T}(RB \mid \alpha, \beta, \psi)$ and $SP^{T}_{min}(RB \mid \alpha, \beta, \psi)$ be the amount of space (in bits) and minimum amount of space (in bits) consumed by *RB* using a rulebase representation technique *T*, respectively. We observe that a rulebase representation technique might not represent *RB* at its minimum level because of the stochastic nature of the set of transactions contained in the database. In other words, a frequent itemset might not generate all the association rules. For example, the association rule $X \rightarrow Y$ might not get extracted from any one of the given databases, even if the itemset $\{X, Y\}$ is frequent in some databases. Thus, $SP^{T}_{min}(RB \mid \alpha, \beta, \psi) \leq SP^{T}(RB \mid \alpha, \beta, \psi)$, for a given tuple (α, β, ψ), where $0 < \alpha \leq \beta \leq 1$. Let Γ be the set of all techniques for representing a set of association rules. We are interested in finding out a technique $T_1 \in \Gamma$ for representing *RB*, such that $SP^{T_1}(RB \mid \alpha, \beta, \psi) \leq SP^{T}(RB \mid \alpha, \beta, \psi)$, for all $T \in \Gamma$. Let $SP_{\min}(RB \mid \alpha, \beta, \psi)$ = *minimum* $\{SP^{T}_{min}(RB \mid \alpha, \beta, \psi): T \in \Gamma\}$. The efficiency of *T* for representing *RB* is judged by comparing $SP^{T}(RB \mid \alpha, \beta, \psi)$ with $SP_{min}(RB \mid \alpha, \beta, \psi)$. We would like to design an efficient rulebase representation technique T_1 such that $SP^{T_1}(RB \mid \alpha, \beta, \psi) \leq SP^{T}(RB \mid \alpha, \beta, \psi)$, for $T \in \Gamma$.

Our work is based on RB_i, for $i = 1, 2, \ldots, n$. One could lower α and β further so that each RB_i represents the corresponding database reasonably well. The work is not concerned with mining branch databases. ACP coding reduces *RB* significantly, so that the coded *RB* becomes available in the main memory during the execution of pattern processing / synthesizing algorithm. The benefits of coding *RB* are given as follows. Firstly, the quality of processed / synthesized knowledge gets enhanced, since the number of local association rules participate in the pattern processing / synthesizing algorithm is more. Secondly, the pattern processing / synthesizing algorithm could access all the local association rules conveniently, since coded *RB* becomes available in the main memory. This arrangement might be possible, since coded *RB* is reasonably small. For the purpose of achieving latter benefit, we propose an index structure to access the coded association rules conveniently. Finally, the coded RB and the corresponding index table could be stored in the secondary storage for the usage of different multi-database mining applications. The following issues are discussed in this chapter.

- We present a technique, called ACP coding, for representing rulebases corresponding to different databases space efficiently. It enables us to incorporate more association rules for synthesizing global patterns or decision-making activities.
- We present an index structure to access the coded association rules conveniently.
- We prove that ACP coding represents *RB* using least amount of storage space in comparison to any other rulebase representation technique.
- We present a technique for storing rulebases corresponding to different databases in the secondary storage.
- We conduct experiments to judge the effectiveness of our approach.

The rest of the chapter is organized as follows. In Section 3.3.2, we discuss related work. A simple coding, called SBV coding, for representing different rulebases is presented in Section 3.3.3. In Section 3.3.4, we present ACP coding for representing rulebases space efficiently. Experimental results are presented in Section 3.3.5.

ferent databases. Thus, it is difficult to handle association rules in different databases effectively during postmining of rulebases corresponding to different databases. Ananthanarayana et al. [16] have proposed PC-tree to represent data completely and minimally in main memory. It is built by scanning database only once. It could be used to represent dynamic databases with the help of knowledge that is either static or changing. It is not suitable for storing and accessing association rules. PC-tree also lacks the capability of handling association rules in different databases during postmining of rulebases corresponding to different databases.

The proposed work falls under the third category of solutions to reducing storage of different rulebases. It is useful for handling association rules effectively during postmining of association rules in different databases. No work has been reported so far under this category.

In the context of mining good quality of knowledge from different data sources, Su et al. [74] have proposed a framework for identifying trustworthy knowledge from external data sources. Such framework might not be useful in this context.

Zhang and Zaki [92] have edited a book on various problems related to multi-database mining. Zhang [88], and Zhang et al. [93] studied various strategies for mining multiple databases. Kum et al. [50] have presented a novel algorithm, ApproxMAP, to mine approximate sequential patterns, called *consensus patterns*, from large sequence databases in two steps. First, sequences are clustered by similarity. Then, consensus patterns are mined directly from each cluster through multiple alignment.

3.3.3 Simple bit vector (SBV) coding

We need to process all the association rules in different local databases for synthesizing patterns, or decision-making applications. We shall use tuple (*ant*, *con*, *s*, *c*) to symbolize an association rule, where *ant*, *con*, *s*, and *c* represent antecedent, consequent, support and confidence of the association rule $ant \rightarrow con$, respectively. We present a situation in Example 3.3.1.

Example 3.3.1. A multi-branch company has four branches. Let $\alpha = 0.35$, and $\beta = 0.45$. Our discussion would be still complete without specification of the content of the branch databases. The rulebases corresponding to different databases are given below.

RB_1 = { (*A*, *C*, 1.0, 1.0), (*C*, *A*, 1.0, 1.0), (*A*, *B*, 0.41877, 0.41877), (*B*, *A*, 0.41877, 0.73740), (*B*, *C*, 0.40341, 0.71035), (*C*, *B*, 0.40341, 0.40341), (*A*, *BC*, 0.36107, 0. 36107), (*B*, *AC*, 0.36107, 0.63580), (*C*, *AB*, 0.36107, 0.36107), (*AB*, *C*, 0.36107, 0.74035), (*AC*, *B*, 0.36107, 0.36107), (*BC*, *A*, 0.36107, 0.89504) }; RB_2 = { (*A*, *C*, 0.66667, 0.66667), (*C*, *A*, 0.66667, 1.0) }; RB_3 = { (*A*, *C*, 0.66667, 0.66667), (*C*, *A*, 0.66667, 1.0), (*A*, *E*, 0.66667, 0.66667), (*E*, *A*, 0.66667, 1.0) }; RB_4 = { (*F*, *D*, 0.75000, 0.75000), (*D*, *F*, 0.75000, 1.0), (*F*, *E*, 0.50000, 0.50000), (*E*, *F*, 0.50000, 1.0), (*F*, *H*, 0.50000, 0.50000), (*H*, *F*, 0.50000, 1.0) }. •

One could represent an associating rule conveniently using an object (or, a record). A typical object representing an association rule consists of following attributes: database identification, number of items in the antecedent, items in the antecedent, number of items in the consequent, items in the consequent, support, and confidence. We calculate the space requirement of such an object using Example 3.3.2.

Example 3.3.2. The discussion of Example 3.3.1 is continued here. A typical compiler represents an integer and a real number using 4 bytes and 8 bytes, respectively. An item could be considered as an integer. Consider the association rule (*A*, *BC*, 0.36107, 0.36107) of RB_1. Each of the following components of an association rule could consume 4 bytes: database identification, number of items in the antecedent, item A, number of items in the consequent, item *B*, and item *C*. Support and confidence of an association rule could consume 8 bytes each. The association rule (*A*, *BC*, 0.36107, 0.36107) of RB_1 thus consumes 40 bytes. The association rule (*A*, *C*, 1.0, 1.0) of RB_1 could consume 36 bytes. Thus, the amount of space required to store four rulebases is equal to $(18 \times 36 + 6 \times 40)$ bytes, i.e. 7104 bits. A technique without optimisation (TWO) could consume 7104 bits to represent these rulebases. •

Let I be the set of all items in D. Let X, Y and Z be three itemsets such that $Y, Z \subseteq X$. Then $\{Y, Z\}$ forms a *2-itemset partition* of X if $Y \cup Z = X$, and $Y \cap Z = \phi$. We define *size* of itemset X as the number of items in X, denoted by $|X|$. Then, we have $2^{|X|}$ 2-itemset partitions of X. For example, $\{\{a\}, \{b, c\}\}$ is a 2-itemset partition of $\{a, b, c\}$. An association rule $Y \rightarrow Z$ corresponds to a 2-itemset partition of X, for $Y, Z \subseteq X$. The antecedent and consequent of an association rule are non-null. Thus, we have Lemma 3.3.1.

Lemma 3.3.1. *An itemset X can generate maximum $2^{|X|}$- 2 association rules, for $|X| \geq 2$.*

Let there are 10 items. The number of itemsets using 10 items is 2^{10}. Thus, 10 bits would be enough to represent an itemset. The itemset ABC, i.e. $\{A, B, C\}$ could be represented by the bit combination 1110000000. 2-itemset partitions of ABC are $\{\phi, ABC\}$, $\{A, BC\}$, $\{B, AC\}$, $\{C, AB\}$, $\{AB, C\}$, $\{AC, B\}$, $\{BC, A\}$, and $\{ABC, \phi\}$. Number of 2-itemset partitions of a set containing 3 items is 2^3. Every 2-itemset partition corresponds to an association rule, except the partitions $\{\phi, ABC\}$ and $\{ABC, \phi\}$. For example, the partition $\{A, BC\}$ corresponds to the association rule $A \rightarrow BC$. Thus, 3 bits are sufficient to identify an association rule generated from ABC. If the number of items is large, then this method might take significant amount of memory space to represent itemsets and the association rules generated from the itemsets. Thus, this technique is not suitable to represent association rules in databases containing large number of items.

3.3.3.1 Dealing with databases containing large number of items

We explain SBV coding with the help of Example 3.3.3.

Example 3.3.3. We continue here the discussion of Example 3.3.1. Let the number of items be 10000. One needs 14 bits to identify an item, since $2^{13} < 10000 \leq 2^{14}$. We assume that the support and confidence of an association rule are represented using 5 digits after the decimal point. Thus, support / confidence value 1.0 could be represented as 0.99999. We use 17-bit binary number to represent support / confidence, since $2^{16} < 99999 \leq 2^{17}$.

Let us consider the association rule (*A*, *BC*, 0.36107, 0. 36107) of RB_1. There are 4 databases viz., D_1, D_2, D_3, and D_4. We need 2 bits to identify a database, since $2^1 < 4 \leq 2^2$. Also, 4 bits could be enough to represent the number of items in an association rule. We put bit 1 at the beginning of binary representation of an item, if it appears in the antecedent of the association rule. We put bit 0 at the beginning of binary representation of an item, if it appears in the consequent of the association rule. Using this arrangement, the lengths of the antecedent and consequent do not required to be stored. The following bit vector could represent the above association rule.

$\underbrace{00}_{1}\underbrace{0011}_{2}\underbrace{1}_{3}\underbrace{0000000000000001}_{4}\underbrace{0}_{5}\underbrace{0000000000000010}_{6}$

$\underbrace{0}_{7}\underbrace{0000000000000011}_{8}\underbrace{01000110100001011}_{9}\ \underbrace{01000110100001011}_{10}$

The components of above bit vector are explained below.

Component 1 represents the first database (i.e., D_1)

Component 2 represents the number of items in the association rule (i.e., 3)

Component 3 (i.e., bit 1) implies that the current item (i.e., item *A*) belongs to antecedent

Component 4 represents item *A* (i.e., item number 1)

Component 5 (i.e., bit 0) implies that the current item (i.e., item *B*) belongs to consequent

Component 6 represents item *B* (i.e., item number 2)

Component 7 (i.e., bit 0) implies that the current item (i.e., item *C*) belongs to consequent

Component 8 represents item *C* (i.e., item number 3)

Component 9 represents support of association rule

Component 10 represents confidence of association rule

The number of bits required for an association rule containing two items and three items are 70 and 85, respectively. Therefore, the amount of storage space required to represent different rulebases is equal to $(18 \times 70 + 6 \times 85)$ bits, i.e., 1770 bits. A technique without optimization could consume 7104 bits (as mentioned in Example 3.3.2) to represent the same. Thus, SBV coding reduces the amount of storage space for representing different rulebases significantly. ∘

In the following section, we consider a special case of bit vector coding. It optimizes the storage space for representing different rulebases based on the fact that many association rules have the same antecedent-consequent pair. Before we move on to the next section, we consider the following lemma.

Lemma 3.3.2. *Let there are p items. Let m be the minimum number of bits required to represent an item. Then,* $m = \lceil log_2(p) \rceil$.

Proof. We have $2^{m-1} < p \leq 2^m$, for an integer m. Thus, we get $m < \log_2(p) + 1$, and $\log_2(p) \leq$ m, since $\log_k(x)$ is a monotonic increasing function of x, for $k > 1$. Combining these two inequalities we get, $\log_2(p) \leq m < \log_2(p) + 1$. ∘

3.3.4 Antecedent-consequent pair (ACP) coding

The central office generates sets of frequent itemsets from different rulebases. Let FIS_i be the set of frequent itemsets generated from RB_i, for $i = 1, 2, ..., n$. Also, let FIS be the union of all frequent itemsets reported from different databases. We symbolize a frequent itemset by a pair (itemset, support). The association rules (F, D, 0.75000, 0.75000) and (D, F, 0.75000, 1.0) of RB_4 generate the following frequent itemsets: (D, 0.75000), (F, 1.0) and (DF, 0.75000). In the following example, we generate FIS_i, for $i = 1, 2, ..., n$.

Example 3.3.4. The discussion of Example 3.3.1 is continued here. The sets of frequent itemsets generated by the central office are given below.

FIS_1 = { (A, 1.0), (C, 1.0), (B, 0.56790), (AC, 1.0), (AB, 0.41877), (BC, 0.40341), (ABC, 0.36107) }; FIS_2 = { (A, 1.0), (C, 0.66667), (AC, 0.66667) }; FIS_3 = { (A, 1.0), (C, 0.66667), (E, 0.66667), (AC, 0.66667), (AE, 0.66667) }; FIS_4 = { (D, 0.75000), (E, 0.50000), (F, 1.0), (H, 0.50000), (DF, 0.75000), (EF, 0.5000), (FH, 0.50000) }. ∘

The ACP coding is a special case of bit vector coding, where antecedent-consequent pairs of the associations rules arc coded in a specific order. The ACP coding is lossless

[67] and similar to Huffman coding [43]. ACP coding and the Huffman coding are not the same, in the sense that an ACP code may be a prefix of another ACP code. Then, how does a search procedure detect antecedent-consequent pair of an association rule correctly? The answer to this question would be available in Section 3.3.4.1.

Let X be a frequent itemset generated from an association rule. Also, let $f(X)$ be the number of rulebases that generate itemset X. Also, let $f_i(X) = 1$, if X is extracted from the i-th database, and $f_i(X) = 0$, otherwise; for $i = 1, 2, \ldots, n$. Then, $f(X) \leq \sum_{i=1}^{n} f_i(X)$. The central office sorts the frequent itemsets X using $|X|$ as the primary key and $f(X)$ as the secondary key, for $X \in FIS$, and $|X| \geq 2$. Initially, the itemsets are sorted on size in non-decreasing order. Then, the itemsets of the same size are sorted on $f(X)$ in non-increasing order. If $f(X)$ is high then the number of association rules generated from X is expected to be high. Therefore, we represent antecedent-consequent pair of such an association rule using a code of smaller size. We shall explain the coding with the help of Example 3.3.5.

Example 3.3.5. We continue here the discussion of Example 3.3.4. We sort all the frequent itemsets of size greater than or equal to 2. Sorted frequent itemsets are presented in Table 3.3.1.

Table 3.3.1. Sorted frequent itemsets of size greater than or equal to 2

X	AC	AB	AE	BC	DF	EF	FH	ABC
$f(X)$	3	1	1	1	1	1	1	1

The coding process is described as follows. Find an itemset that has a maximal f-value. Itemset AC has the maximum f-value. We code AC as 0. The maximum number of association rules could be generated from AC is two. Thus, we code association rules $A \rightarrow C$ and $C \rightarrow A$ as 0 and 1, respectively. Now, 1-digit codes are finished. Then, we find an itemset that has a second maximal f-value. We choose AB. We could have chosen any itemset from $\{AB, AE, BC, DF, EF, FH\}$, since every itemset in the set has the same size and the same f-value. We code AB as 00. The maximum number of association rules could be generated from AB is two. Thus, we code the association rules $A \rightarrow B$ and $B \rightarrow A$

as 00 and 01, respectively. We follow in the same way and code the association rules $A \rightarrow E$ and $E \rightarrow A$ as 10 and 11, respectively. Now, 2-digit codes are finished. Finally, we choose *ABC*. We code *ABC* as 0000. The association rules $A \rightarrow BC$, $B \rightarrow AC$, $C \rightarrow AB$, $AB \rightarrow C$, $AC \rightarrow B$, and $BC \rightarrow A$ get coded as 0000, 0001, 0010, 0011, 0100, and 0101, respectively. Each frequent itemset receives a code. We call it an *itemset code*. Also, antecedent-consequent pair of an association rule receives a code. We call it a *rule code*. •

Now, an association rule could be represented in the main memory using the following components: database identification number, ACP code, support, and confidence. Let n be the number of databases. Then we have $2^{k-1} < n \leq 2^k$, for an integer k. Thus, we need k bits to represent the database identification number. We represent support / confidence using p digits after the decimal point. If we represent a fraction f using an integer d and then f could be obtained by the formula: $f = d \times 10^{-p}$. We represent support / confidence by storing the corresponding integer. The following lemma determines the minimum number of binary digits required to store a decimal number.

Lemma 3.3.3. *A p-digit decimal number can be represented by a* $\lceil p \times \log_2 10 \rceil$ *-digit binary number.*

Proof. Let t be the minimum number of binary digits required to represent a p-digit decimal number x. Then we have $x < 10^p < 2^t$. So, $t > p \times \log_2 10$, since $\log_k(y)$ is a monotonic increasing function of y, for $k > 1$. Thus, the minimum integer t for which $x < 2^t$ is true is given by $\lceil p \times \log_2 10 \rceil$. •

The following lemma determines the minimum amount of storage space required to represent *RB* under some conditions.

Lemma 3.3.4. *Let M be the number of association rules having distinct antecedent-consequent pairs among N association rules extracted from n databases, where* $2^{m-1} < M \leq 2^m$, *and* $2^{p-1} < n \leq 2^p$, *for some positive integers m and p. Suppose the support and confidence of an association rule are represented by a fractions containing k digits after the decimal point. Assume that a frequent itemset X generates all possible association* *rules, for $X \in FIS$, and $|X| \geq 2$. Then the minimum amount of storage space required to represent RB in the main memory is given as follows.*

$SP^{ACP\ coding}_{min,main}(RB \mid \alpha, \beta, \psi) = M \times (m-1) - 2 \times (2^{m-1} - m) + N \times (p + 2 \times \lceil k \times log_2 10 \rceil)$ *bits,*

if $M < 2^m - 2$; *and*

$SP^{ACP\ coding}_{min,main}(RB \mid \alpha, \beta, \psi) = M \times m - 2 \times (2^m - m - 1) + N \times (p + 2 \times \lceil k \times log_2 10 \rceil)$ *bits,*

otherwise.

Proof. p bits are required to identify a database. The amount of memory required to represent database identifications of N association rules is equal to $P = N \times p$ bits. The minimum amount of memory required to represent both support and confidence of N association rules is equal to $Q = N \times 2 \times \lceil k \times \log_2 10 \rceil$ bits (as mentioned in Lemma 3.3.3). Let R be the minimum amount of memory required to represent ACPs of M association rules. The expression R could be obtained from the fact that 2^1 ACPs are of length 1, 2^2 ACPs are of length 2, and so on. The expression of R is given as follows.

$R = \sum_{i=1}^{m-2} i \times 2^i + (m-1) \times \left(M - \sum_{i=1}^{m-2} 2^i\right)$bits, if $\left(M - \sum_{i=1}^{m-2} 2^i\right) < 2^{m-1}$; and

$$R = \sum_{i=1}^{m-1} i \times 2^i + m \times \left(M - \sum_{i=1}^{m-1} 2^i\right) \text{ bits, if } \left(M - \sum_{i=1}^{m-2} 2^i\right) \geq 2^{m-1}. \quad (3.3.1)$$

R assumes second form of expression for a few cases. For example, if (M = 15) then R assumes second form of expression. The ACP codes are given as follows: 0, 1, 00, 01, 10, 11, 000, 001, 010, 011, 100, 101, 110, 111, 0000. Then, the minimum amount of storage space required to represent RB is equal to (P + Q + R) bits. Now, $\sum_{i=1}^{m-2} 2^i = 2^{m-1} - 2$, and $\sum_{i=1}^{m-2} i \times 2^i = (m-3) \times 2^{m-1} + 2$. Thus, the lemma follows. •

In the following example, we calculate the amount of storage space required for representing rulebases of Example 3.3.1.

Example 3.3.6. The discussion of Example 3.3.1 is continued here. The number of association rules in RB is 24. With reference to Lemma 3.3.4, we have N = 24, M = 20, and n = 4. Thus, m = 5, and p = 2. Assume that the support and confidence of an association rule are represented by fractions containing 5 digits after the decimal point.

Thus, $k = 5$. Then, the minimum amount of storage space required to represent *RB* is 922 bits. •

ACP coding may assign some codes for which there exist no associated rules. Let *ABC* be a frequent itemset extracted from some databases. Assume that the association rule $AC \rightarrow B$ is not extracted from any one of the databases that extract *ABC*. Let the itemset code corresponding to *ABC* is 0000. Then, the ACP code for $AC \rightarrow B$ is 0100, i.e., the 4-th association rule generated from *ABC*. Therefore, ACP coding does not always store rulebases at the minimum level.

All rule codes are ACP codes. But, the converse of the statement is not true. Some ACP codes do not have assigned association rules, since the assigned association rules do not get extracted from any one of the given databases. An ACP code *X* is *empty* if *X* is not a rule code.

Lemma 3.3.5. *Let $X \in FIS$ such that $|X| \geq 2$. We assume that X generates at least one association rule. Let m (≥ 2) be the maximum size of a frequent itemset in FIS. Let n_i be the number of distinct frequent itemsets in FIS of size i, for i = 2, 3, ..., m. Then, the maximum number of empty ACP codes is equal to* $\sum_{i=2}^{m}(2^i - 3) \times n_i$.

Proof. In the extreme case, only one association rule is generated for each frequent itemset *X* in *FIS*, such that $|X| \geq 2$. Using Lemma 3.3.1, a frequent itemset *X* could generate maximum $2^{|X|}$-2 association rules. In such a situation, $2^{|X|}$-3 ACP codes are empty for *X*. Thus, the maximum number of empty ACP codes for the frequent itemsets of size *i* is equal to $(2^i-3) \times n_i$. Hence the result follows. •

To search an association rule we maintain all the itemsets in *FIS* along with their itemset codes in an index table such that the size of an itemset is greater than one. We generate rule codes of the association rules from the corresponding itemset code. In Section 3.3.4.1, we discuss a procedure for constructing index table and accessing mechanism for the association rules.

3.3.4.1 Indexing rule codes

An index table contains the frequent itemsets of size greater than one and the corresponding itemset codes. These frequent itemsets are generated from different rule-bases. Example 3.3.2 illustrates the procedure of searching an association rule in the index table.

Example 3.3.7. We continue here the discussion of Example 3.3.5.

Table 3.3.2. Index table for searching an association rule

Itemset	*AC*	*AB*	*AE*	*BC*	*DF*	*EF*	*FH*	*ABC*
Code	0	00	10	000	010	100	110	0000

We construct the above index table as follows. The itemset code corresponding to AC is 0. The itemset code 0 corresponds to the set of association rules $\{A \rightarrow C, C \rightarrow A\}$. We would like to discuss the procedure for searching an association rule in the index table. Suppose we wish to search the association rule corresponding to rule code 111. We apply binary search technique to find code 111. The binary search technique is based on the length of an itemset code. The search might end up at the fourth cell containing itemset code 000. Now, we apply sequential search towards the right side of the forth cell, since *value*(000) < *value*(111). We find that 111 is not present in the index table. But, the code 111 lies between 110 and 0000, since $|111| < |0000|$ and *value*(111) > *value*(110). We define *value* of a code ω as the numerical value of the code, i.e. $value(\omega) = (\omega)_{10}$. For example, *value* (010) = 2. Thus, the sequential search stops at the cell containing itemset code 110. In general, for a rule code ω, we get a consecutive pair of itemset codes ($code_1$, $code_2$) in the index table, such that $code_1 \le \omega < code_2$. Then $code_1$ is the desired itemset code. Let Y be the desired itemset corresponding to the rule code ω. Then, ω corresponds to an association rule generated by Y. Thus, the itemset code corresponding to the rule ing) is significantly less than that of other techniques. We illustrate this issue in Example 3.3.8.

Example 3.3.8. The discussion of Example 3.3.7 is continued here. We have 8 frequent itemsets in the index table. Let there are 10000 items in the given databases. Therefore, 14 bits are required to identify an item. Thus, the amount of storage space would require for AC and ABC are equal to $2 \times 14 = 28$ bits, and $3 \times 14 = 42$ bits, respectively. The size of index file is the size of itemsets plus the size of itemset codes. In this case, the index table consumes $(28 \times 7 + 42 \times 1) + 21$ bits, i.e., 259 bits. The total space required (including the overhead of indexing) to represent RB is equal to $(259 + 922)$ bits (as mentioned in Example 3.3.6) = 1181 bits. Based on the running example, we compare the amounts of storage space required to represent RB using different rulebase representation techniques.

Table 3.3.3. Amounts of storage space required for representing RB using different rulebase representation techniques

Technique for representing RB	TWO	SBV	ACP
Amount of space (bits)	7104	1770	1181

We observe that ACP coding consumes the least amount of space to represent RB. Let $OI(T)$ be the overhead of maintaining index table using technique T. A technique without optimization (TWO) might not maintain index table separately. In this case, OI(TWO) = 0 bit. But, ACP coding performs better than a TWO, because ACP coding optimizes storage spaces for representing components of an association rule. •

We describe here the data structures used in the algorithm for representing rulebases using ACP coding. A frequent itemset could be described by the following attributes: database identification, itemset and support. The frequent itemset generated from RB_i are stored into array FIS_i, for $i = 1, 2, \ldots, n$. We keep all the generated frequent itemsets into array FIS. Also, we have calculated f-value for every distinct frequent itemset X in FIS

code 111 is 110. The frequent itemset corresponding to itemset code 110 is *FH*. Thus, the association rule corresponding rule code 111 is $H \rightarrow F$. •

Initially, the binary search procedure finds an itemset code of desired length. Then, it moves forward or backward sequentially till we get the desired itemset code. The algorithm for searching an itemset code is presented below.

Algorithm 3.3.1. Search for the itemset code corresponding to a rule code in the index table.

```
procedure itemset-code-search ($\omega$, $T$, $i$, $j$)
Inputs:
$\omega$: rule code (an ACP code)
$T$: index table
$i$: start index
$j$: end index
Outputs:
index of the itemset code corresponding to $\omega$
01: $x = |\omega|$;
02: $k$ = binary-search ($x$, $T$, $i$, $j$);
03: if (value($\omega$) $\geq$ value(($T(k)$.code)) then
04:     $q$ = forward-sequential-search ($\omega$, $T$, $k + 1$, $j$);
05: else
06:     $q$ = backward-sequential-search ($\omega$, $T$, $k - 1$, $i$);
07: end if
08: return($q$);
end procedure
```

The above algorithm is described as follows. The algorithm *itemset-code-search* [4] searches index table T between the i-th and j-th cells, and returns the index of the itemset code corresponding to the rule code ω. The procedure *binary-search* returns an integer k corresponding to rule code ω. If $value(\omega) \geq value((T(k).code)$ then we search sequentially in T from index $(k+1)$ to j. Otherwise, we search sequentially in T from index $(k-1)$ down to i. Let there are m cells in the index table. Then binary search makes maximum $\lfloor \log_2(m) \rfloor + 1$ comparisons [49]. The sequential search makes O(1) comparison in this case. Therefore, algorithm *itemset-code-search* takes O(log(m)) time.

Now, we need to find the association rule generated from the itemset corresponding to the itemset code returned by algorithm *itemset-code-search.* We shall take an example to illustrate the procedure for identifying association rule for a given rule code. Let us consider the rule code 0100. Using above technique, we determine that 0000 is the itemset code corresponding to rule code 0100. The itemset corresponding to the itemset code 0000 is ABC. The association rules generated from itemset ABC could be numbered as follows: 0-th association rule (i.e., $A \rightarrow BC$) has rule code 0000, 1-th association rule (i.e., $B \rightarrow AC$) has rule code 0001, and so on. Proceeding in this way, we find that the 4-th association rule (i.e., $AC \rightarrow B$) has rule code 0100.

We shall now find the association rule number corresponding to rule code ω. Let X be the itemset corresponding to rule code ω, and ν be the itemset code corresponding to X. Let $RB(X)$ be the set of all possible association rules generated by X. From Lemma 3.3.1, we have $|RB(X)| = 2^{|x|} - 2$, for $|X| \geq 2$. If $|\nu| = |\omega|$ then ω corresponds to $(\omega_{10} - \nu_{10})$-th association rule generated from X, where Y_{10} denote the decimal value corresponding to binary code Y. If $|\nu| < |\omega|$ then ω corresponds to $(2^{|\nu|} - \omega_{10} + \nu_{10})$-th association rule generated from X. In this case, $\nu = 0000$, $\omega = 0100$, and $X = ABC$. Thus, ω corresponds to 4-th association rule generated from X.

Algorithm 3.3.2. Find itemset and association rule number corresponding to a rule code.

```
procedure rule-generation (k, T, C, X)
Input:
k: index
T: index table
C: rule code (an ACP code)
Output:
itemset X corresponding to C
association rule number corresponding to C
01: let $X = T(k).itemset$;
02: if ( $|T(k).code| = |C|$ ) then
03:     return $(C_{10} - (T(k).code)_{10})$;
04: else
05:     return $(2^{|T(k).code|} - (C)_{10} + (T(k).code)_{10})$;
06: end if
end procedure
```

We assume that the algorithm *itemset-code-search* [4] returns *k* as the index of the itemset code corresponding to rule code *C*. Using index table *T* and *k*, the algorithm *rule-generation* returns the rule number and the itemset corresponding to rule code *C*. The itemset is returned through argument *X*, and rule number is returned through the return statement.

ACP coding maintains an index table in main memory. We shall take an example to verify that the amount of space consumed by a rulebase (including the overhead of indexing) is significantly less than that of other techniques. We illustrate this issue in Example 3.3.8.

Example 3.3.8. The discussion of Example 3.3.7 is continued here. We have 8 frequent itemsets in the index table. Let there are 10000 items in the given databases. Therefore, 14 bits are required to identify an item. Thus, the amount of storage space would require for *AC* and *ABC* are equal to $2 \times 14 = 28$ bits, and $3 \times 14 = 42$ bits, respectively. The size of index file is the size of itemsets plus the size of itemset codes. In this case, the index table consumes $(28 \times 7 + 42 \times 1) + 21$ bits, i.e., 259 bits. The total space required (including the overhead of indexing) to represent *RB* is equal to (259 + 922) bits (as mentioned in Example 3.3.6) = 1181 bits. Based on the running example, we compare the amounts of storage space required to represent *RB* using different rulebase representation techniques.

Table 3.3.3. Amounts of storage space required for representing *RB* using different rulebase representation techniques

Technique for representing *RB*	TWO	SBV	ACP
Amount of space (bits)	7104	1770	1181

We observe that ACP coding consumes the least amount of space to represent *RB*. Let *OI*(*T*) be the overhead of maintaining index table using technique *T*. A technique without optimization (TWO) might not maintain index table separately. In this case, *OI*(TWO) = 0 bit. But, ACP coding performs better than a TWO, because ACP coding optimizes storage spaces for representing components of an association rule. •

We describe here the data structures used in the algorithm for representing rulebases using ACP coding. A frequent itemset could be described by the following attributes: database identification, itemset and support. The frequent itemset generated from RB_i are stored into array FIS_i, for $i = 1, 2, \ldots, n$. We keep all the generated frequent itemsets into array *FIS*. Also, we have calculated *f*-value for every distinct frequent itemset *X* in *FIS* such that $|X| \geq 2$. The frequent itemsets and their *f*-values are stored into array *IS_Table*. We present below an algorithm for representing different rulebases using ACP coding [4].

Algorithm 3.3.3. Represent rulebases using ACP coding.

procedure *ACP-coding* (*n*, *RB*)

Input:

n: number of databases

RB: union of rulebases

Output:

Coded association rules

```
01: let $FIS = \phi$;
02: for $i = 1$ to $n$ do
03:     read $RB_i$ from secondary storage;
04:     generate $FIS_i$ from $RB_i$;
05:     $FIS = FIS \cup FIS_i$;
06: end for
07: let $j = 1$;
08: let $i = 1$;
09: while ( $i \leq |FIS|$ ) do
10:     if ( $|FIS(i).itemset| \geq 2$ ) then
11:         compute $f(X)$;
12:         $IS\_Table(j).itemset = X$;
13:         $IS\_Table(j).f(X) = f(X)$;
```

```
14:         increase index $j$ by 1;
15:         update index $i$ for processing the next frequent itemset in $FIS$;
16:     end if
17: end for
18: sort itemsets in $IS\_Table$ using $|X|$ as the primary key and $f(X)$ as the secondary key;
19: for $i = 1$ to $|IS\_Table|$ do
20:     $C$ = ACP code of $IS\_Table(i).itemset$;
21:     $T(i).itemset = IS\_Table(i).itemset$;
22:     $T(i).code = C$;
23: end for
end procedure
```

Using lines 1-6, we have generated frequent itemsets from different rulebases and stored them into array *FIS*. We compute f-value for every frequent itemset X and store it into *IS_Table* using lines 7-17, for $|X| \geq 2$. At line 18, we sort frequent itemsets in *IS_Table* for the purpose of coding. Index table T is constructed using lines 19-23.

We calculate time complexities of different statements except the shaded statement of the above algorithm, since it involves reading data from secondary storage. Let the maximum of $\{|FIS_i|: 1 \leq i \leq n\}$ be p. Then the total number of itemsets is $O(n \times p)$. Therefore, lines 7-17 take $O(n \times p)$ time. Line 18 takes $O(n \times p \times \log(n \times p))$ time to sort $O(n \times p)$ itemsets. Lines 19-23 take $O(n \times p)$ time to construct the index table.

3.3.4.2 Storing rulebases in secondary memory

An association rule could be stored in main memory using the following components: database identification, rule code, support, and confidence. Database identification, support and confidence could be stored using the method described in Section 3.3.3. Also, we need to maintain an index table in main memory to code / decode an association rule.

The rulebases corresponding to different databases could be stored in secondary memory using a bit sequential file *F*. The first line of *F* contains the number of databases. The second line of *F* contains the number of association rules in the first rulebase. The following lines of *F* contain the association rules in the first rulebase. After keeping all the association rules in the first rulebase, we keep number of association rules in the second rulebase, and the association rules in the second rulebase thereafter. We illustrate the proposed file structure in Example 3.3.9.

Example 3.3.9. Assume that there are 3 databases D_1, D_2, and D_3. Let the number of association rules extracted from these databases be 3, 4, and 2, respectively. The coded rulebases could be stored as follows.

<3><\n>

<3><\n>

$<r_{11}><s_{11}><c_{11}><\backslash n>$

$<r_{12}><s_{12}><c_{12}><\backslash n>$

$<r_{13}><s_{13}><c_{13}><\backslash n>$

<4><\n>

$<r_{21}><s_{21}><c_{21}><\backslash n>$

$<r_{22}><s_{22}><c_{22}><\backslash n>$

$<r_{23}><s_{23}><c_{23}><\backslash n>$

$<r_{24}><s_{24}><c_{24}><\backslash n>$

<2><\n>

$<r_{31}><s_{31}><c_{31}><\backslash n>$

$<r_{32}><s_{32}><c_{32}><\backslash n>$

'\n' stands for the new line character. While storing an association rule in the secondary memory, if it contains a bit combination as that of '\n', then we need to stuff one more '\n' after the occurrence of '\n'. We need not store the database identification along with an association rule, since the i-th set of association rules corresponds to the i-th database, for $i = 1, 2, 3$. Notations r_{ij}, s_{ij}, and c_{ij} denote the rule code, support, and confidence of j-th association rule reported from i-th database, respectively, for $j = 1, 2, \ldots, |RB_i|$, and $i = 1, 2, 3$. •

Lemma 3.3.6. *Let M be the number of association rules with distinct antecedent-consequent pairs among N association rules reported from n databases, where $2^{m-1} < M \leq 2^m$, for an integer m. Suppose the support and confidence of an association rule are represented by fractions containing k digits after the decimal point. Assume that a frequent itemset X in FIS generates all possible association rules, for $|X| \geq 2$. Then the minimum amount of storage space required to represent RB in secondary memory is given as follows.*

$SP^{ACP\ coding}_{min,\,secondary}(RB \mid \alpha, \beta) = 12 \times n + M \times (m - 1) + N \times (2 \times \lceil k \times \log_2 10 \rceil + 8) - 2 \times (2^{m-1} - m) + 12$ *bits, if* $M < 2^m - 2$, *and*

$SP^{ACP\ coding}_{min,\,secondary}(RB \mid \alpha, \beta) = 12 \times n + M \times m + N \times (2 \times \lceil k \times \log_2 10 \rceil + 8) - 2 \times (2^m - m - 1) + 12$ *bits, otherwise.*

Proof. We do not need to store the database identification in the secondary storage, as the rulebases are stored sequentially one after another. A typical compiler represents '\n' and an integer value using 1 byte and 4 bytes, respectively. The amount of memory required to represent the new line characters is equal to $P = 8 \times (N + n + 1)$ bits. The amount of memory required to store the number of databases and the number of association rules of each rulebase is equal to $Q = 4 \times (n + 1)$ bits. The amount of memory required to represent the both the support and confidence of N rules is equal to $R = N \times 2\times \lceil k \times \log_2 10 \rceil$ bits (as mentioned in Lemma 3.3.3). Let S be the minimum amount of memory required to represent the ACPs of M rules. Then, $S = M \times (m\text{-}1) - 2 \times (2^{m-1} - m)$ bits, if $M < 2^m - 2$, and $S = M \times m - 2 \times (2^m - m - 1)$ bits, otherwise (as mentioned in Lemma 3.3.4). Thus, the minimum amount of storage space required to represent RB in the secondary memory is equal to $(P + Q + R + S)$ bits. •

3.3.4.3 Space efficiency of our approach

The effectiveness of a rulebase representation technique requires to be validated by its storage efficiency. There are many ways one could define the storage efficiency of a rulebase representation technique. We use the following definition to measure the storage efficiency of a rulebase representation technique.

Definition 3.3.1. Let RB_i be the rulebase corresponding to database D_i at a given pair (α, β), for $i = 1, 2, \ldots, n$. Let RB be the union of rulebases corresponding to different databases. The space efficiency of technique T for representing RB is defined as follows.

$$\varepsilon(T, RB \mid \alpha, \beta, \psi) = \frac{SP_{min}(RB \mid \alpha, \beta, \psi)}{SP^T(RB \mid \alpha, \beta, \psi)}, \text{ for } T \in \Gamma$$

with respect to the symbols and notations used in Section 3.3.1. •

We note that $0 < \varepsilon \le 1$. A rulebase representation technique is good if ε is high. We would like to show that ACP coding stores rulebases at higher level of efficiency than that of any other representation technique.

Lemma 3.3.7. *Let RB_i be the set of association rules extracted from database D_i at a given pair (α, β), for $i = 1, 2, \ldots, n$. Let RB be the union of rulebases corresponding to different databases. Also, let Γ be the set of all rulebase representation techniques. Then, $\varepsilon(ACP\ coding, RB \mid \alpha, \beta, \psi) \ge \varepsilon(T, RB \mid \alpha, \beta, \psi)$, for $T \in \Gamma$.*

Proof. We shall show that ACP coding stores *RB* using minimum storage space at a given pair (α, β). A local association rule has the following components: database identification, antecedent, consequent, support, and confidence. We classify the above components into following three groups: {database identification}, {antecedent, consequent}, and {support, confidence}. Among these three groups, the item of group 1 is independent of the items of other groups. If there are *n* databases, we need minimum $\lceil \log_2 n \rceil$ bits to represent the item of group 1 (as mentioned in Lemma 3.3.2). Many association rules may have the same antecedent-consequent pair. If an antecedent-consequent pair appears in many association rules, then it receives a shorter code. Therefore, the antecedent-consequent pair of association rule having highest frequency is represented by a code of smallest size. ACP code starts from 0, and then follows the sequence 1, 00, 01, 10, 11, 000, 001, ... Therefore, no other technique would provide sizes of codes lesser than them. Therefore, the items of group 2 are expressed minimally using ACP codes. Again, the items of group 3 are related with the items of group 2. Suppose we keep *p* digits after the decimal point for representing an item of group 3. Then, the representation an item of group 3 becomes independent of items of group 2. We need minimum $2 \times \lceil p \times \log_2 10 \rceil$ bits to represent support and confidence of an association rule (as mentioned in Lemma 3.3.3). Thus, *minimum {representation of an association rule} = minimum {representation of items of group 1 + representation of items of group 2 + representation of items of group 3} = minimum {representation of items of group 1} + minimum {representation of items of group 2} + minimum{ representation of items of group 3}*.

Also, there will be an entry in the index table for the itemset corresponding to an association rule for coding / decoding process. Thus, *minimum {representation of index table} = minimum {representation of itemsets + representation of codes}*. If there are *p* items then an itemset of size *k* could be represented by $k \times \lceil \log_2 (p) \rceil$ bits (as mentioned in

Lemma 3.3.2). Also, ACP codes consume minimum space because of the way they have been designed. Thus, *minimum {representation of index table}* = *minimum {representation of itemsets}* + *minimum {representation of codes}*. Therefore, *minimum {representation of rulebases}* = $\sum_r${*representation of association rule r using ACP coding*} + *representation of index table used in ACP coding*. Hence, the lemma follows. •

Lemma 3.3.8. *Let RB_i be the set of association rules extracted from database D_i at a given pair (α, β), for $i = 1, 2, ..., n$. Let RB be the union of rulebases corresponding to different databases. Then, $SP_{min}(RB \mid \alpha, \beta, \psi) = SP_{min}^{ACP\ coding}(RB \mid \alpha, \beta, \psi)$.*

Proof. From Lemma 3.3.7, we conclude that ACP coding represents rulebases using lesser amount of storage space than that of any other technique. Thus, $SP_{min}^{ACP\ coding}(RB \mid \alpha, \beta, \psi) \le SP_{min}^{T}(RB \mid \alpha, \beta, \psi)$, for $T \in \Gamma$. We observe that a rulebase representation technique T might not represent rulebases at its minimum level because of the stochastic nature of the set of transactions contained in a database. In other words, a frequent itemset may not generate all the association rules in a database. For example, the association rule $X \rightarrow Y$ may not get extracted from some of the given databases, even if itemset $\{X, Y\}$ is frequent in the remaining databases. Thus, if ACP coding represents RB using minimum storage space, then it would be the minimum representation of RB at a given tuple (α, β, ψ). •

There are many ways one could define the quality of synthesized patterns. We define the quality of synthesized patterns as follows.

Definition 3.3.2. Let RB_i be the rulebase extracted from database D_i at a given pair (α, β), for $i = 1, 2, ..., n$. Let RB be the union of rulebases corresponding to different databases. We represent RB using a rulebase representation technique T. Let $\xi(RB \mid T, \alpha, \beta, \varphi, \psi)$ denote the collection of synthesized patterns over RB at a given tuple $(T, \alpha, \beta, \varphi, \psi)$. We define quality of $\xi(RB \mid T, \alpha, \beta, \varphi, \psi)$ as $\varepsilon(T, RB \mid \alpha, \beta, \varphi, \psi)$, with respect to the symbols and notations used in Section 3.3.1. •

Also, ε (*ACP coding, RB, α, β*) $\geq$ ε (*T, RB, α, β*), for $T \in \Gamma$ (as mentioned in Lemma 3.3.7). Thus, quality of $\xi(RB \mid ACP\ coding, \alpha, \beta, \varphi, \psi) \geq$ quality of $\xi(RB \mid T, \alpha, \beta, \varphi, \psi)$, for $T \in \Gamma$.

3.3.5 Experiments

We have carried out several experiments to study the effectiveness of ACP coding. All the experiments have been implemented on a 1.6 GHz Pentium processor with 256 MB of memory using visual C++ (version 6.0) software. The following experiments are based on the transactional databases T10I4D100K (T_1) [34], and T40I10D100K (T_2) [34]. These databases were generated using synthetic database generator from IBM Almaden Quest research group. We present some characteristics of these databases in Table 3.3.4.

Table 3.3.4. Database characteristics

Database	*NT*	*ALT*	*AFI*	*NI*
T_1	100000	11.10228	1276.12413	870
T_4	100000	40.40507	4310.51698	942

For the purpose of conducting the experiments, we divide each of these databases into 10 databases. We call these two sets of 10 databases as the input databases. The database T_i has been divided into 10 databases T_{ij} of size 10000 transactions each, for j = 0, 1, 2, …, 9, and i = 1, 4. We present the characteristics of the input databases in Table 3.3.5.

Table 3.3.5. Input database characteristics

Database	*ALT*	*AFI*	*NI*	Database	*ALT*	*AFI*	*NI*
T_{10}	11.05500	127.65588	866	T_{40}	40.56710	431.56489	940
T_{11}	11.13330	128.41176	867	T_{41}	40.58240	432.18743	939
T_{12}	11.06700	127.64705	867	T_{42}	40.63190	431.79489	941
T_{13}	11.12260	128.43649	866	T_{43}	40.62690	431.74176	941
T_{14}	11.13670	128.74797	865	T_{44}	40.66110	432.56489	940
T_{15}	11.13910	128.62702	866	T_{45}	40.50630	430.46014	941
T_{16}	11.10780	128.56250	864	T_{46}	40.74350	433.44148	940
T_{17}	11.09840	128.45376	864	T_{47}	40.62380	431.70882	941
T_{18}	11.08150	128.55568	862	T_{48}	40.52810	431.15000	940
T_{19}	11.08140	128.10867	865	T_{49}	40.57960	432.15761	939

The results of mining input databases are given in Table 3.3.6. The notations used in the above tables are explained as follows. *NT*, *ALT*, *AFI* and *NI* stand for number of transactions, average length of a transaction, average frequency of an item, and number of items in the data source, respectively. Some results on association rule mining are presented in Table 3.3.6.

Table 3.3.6. Result of data mining

Database	α	β	*N2IR*	*N3IR*	*NkIR* ($k > 3$)
$\bigcup_{i=1}^{10} T_{1i}$	0.01	0.2	136	29	0
$\bigcup_{i=1}^{10} T_{4i}$	0.05	0.2	262	0	0

In the above table, *NkIR* stands for the number of k-item association rules from different databases, for $k \geq 2$. We present comparison among different rulebase representation techniques in Table 3.3.7.

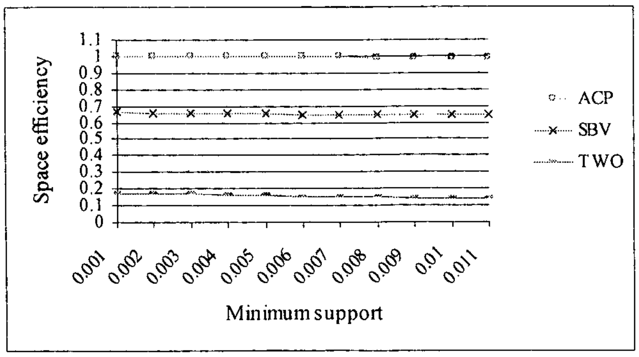
Table 3.3.7. Comparison among different rulebase representation techniques (contd.)

Database	*SP*(TWO)	*SP*(SBV)	*OI*	*SP*(ACP)	*MSO*	*AC*(SBV)	*AC*(ACP)
$\bigcup_{i=1}^{10} T_{1i}$	48448 bits	10879 bits	619 bits	7121 bits	7051 bits	1.79640	1.17586
$\bigcup_{i=1}^{10} T_{4i}$	75456 bits	16768 bits	549 bits	10681 bits	10661 bits	1.77778	1.13242

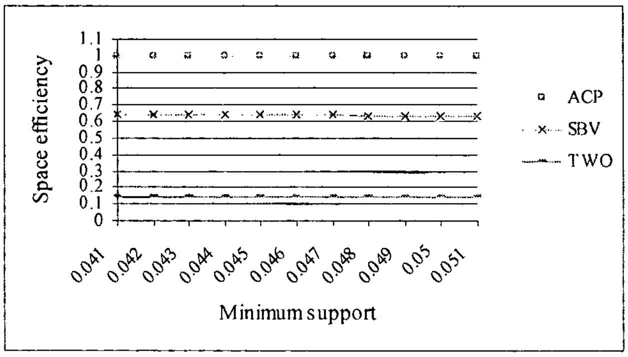
Table 3.3.7. Comparison among different rulebase representation techniques

Database	ε(TWO)	ε(SBV)	ε(ACP)
$\bigcup_{i=1}^{10} T_{1i}$	0.14554	0.64813	0.99017
$\bigcup_{i=1}^{10} T_{4i}$	0.14129	0.63579	0.99813

In the above table, we use the following abbreviations: *SP* stands for storage space (including overhead of indexing), *MSO* stands for minimum storage space for representing rulebases including the overhead of indexing, and *AC*(*T*) stands for amount of compression (bits/byte) using technique *T*. In Figure 3.3.1, we compare different rulebase representation techniques at different levels of minimum support.



(a) For association rules extracted fromT_{1i}, for i = 0, 1, ..., 9.



(b) For association rules extracted fromT_{4i}, for $i = 0, 1, \ldots, 9$.

Figure 3.3.1. Storage efficiency of different rulebase representation techniques

We have taken $\beta = 0.2$ for all the experiments. The results show that the ACP coding stores rulebases most efficiently among different rulebase representation techniques. Also, we find that the SBV coding reduces the size of a rulebase considerably, but stores less efficiently than ACP coding. ACP coding achieves maximum efficiency when the following two conditions are satisfied: (i) All the databases are similar type and extract identical set of association rules, and (ii) Each of the frequent itemsets of size greater than one generates all possible association rules.

Nelson [57] studied data compression with the Burrows-Wheeler Transformation (BWT) [23]. Experiments were carried out on 18 different files and average compression obtained by techniques using BWT and PKZIP are 2.41 bits/byte and 2.64 bits/byte, respectively. The commercial products like PKZIP, WINZIP would compress an association rule as a string. Though we have studied the technique BWT, it might be unfair to compare ACP coding with the compression technique using BWT.

The results of Figure 3.3.1(a) and Figure 3.3.1(b) are carried out at 11 different pairs of (α, β). Using ACP coding, we have obtained average compression 1.15014 bits/byte and 1.12190 bits/bytes for the experiments corresponding to Figure 3.3.1(a) and Figure 3.3.1(b), respectively.

3.3.6 Conclusion

An efficient storage representation of a set of pattern bases builds the foundation of a multi-database mining system. Many global applications could be developed effectively upon this foundation. Similar technique could be employed to store frequent itemsets in different databases efficiently. Experimental and theoretical results show that the proposed rulebase representation technique is very efficient.

Chapter 3.4

Efficient clustering of databases induced by local patterns

Many large organizations operate from multiple branches. Some of the branches collect data continuously and store data locally. Thus, the collection of all branch databases might be very large. Effective data analysis using a traditional data mining technique on multi-gigabyte repositories has proven difficult. A quick approximate knowledge from large databases would be adequate for many decision support applications.

Consider a company that deals with multiple large databases. The company might need to make an association analysis involving non-profit making items (products). The objective is to identify the items that neither make much profit nor help promoting other products. An association analysis involving non-profit making items might identify such items. The company could then stop dealing with such items. Such analysis might require identifying similar databases. Two databases are similar if they contain many similar transactions. Again, two transactions are similar if they have many common items. We shall observe latter that two databases containing many common items are not necessarily very similar. First we define a few terms used frequently in this chapter.

Let $I(D)$ be the set of items in database D. An *itemset* is a set of items in a database. An itemset X in D is associated with a statistical measure called support [11], denoted by $supp(X, D)$, for $X \subseteq I(D)$. *Support* of an itemset X in D is the fraction of transactions in D containing X. The importance of an itemset could be judged by its support. X is called a *frequent itemset* (*FIS*) in D if $supp(X, D) \geq \alpha$, where α is the user-defined *minimum sup-*

port. A frequent itemset possesses higher support. Thus, the collection of frequent itemsets determines major characteristics of a database. One could define similarity between a pair databases in terms of their frequent itemsets. Thus, two databases are similar if they have many common frequent itemsets.

Based on the similarity between two databases, one could cluster branch databases. After the clustering process, one could mine all the databases in a class together to make an approximate association analysis involving frequent items. An approximate association analysis could be performed using the frequent itemsets in the union of all the databases in a class. Clustering of databases thus helps reducing data for analyzing the items. In this chapter, we study the problem of clustering transactional databases using the local frequent itemsets.

For clustering transactional databases, Wu et at. [83] have proposed two similarity measures sim_1, and sim_2. Let $D = \{D_1, D_2, ..., D_n\}$, where D_i is the database corresponding to the i-th branch of a multi-branch company, for $i = 1, 2, ..., n$. sim_1 is based on the items in the databases, and defined as follows.

$sim_1(D_1, D_2) = |I(D_1) \cap I(D_2)| \,/\, |I(D_1) \cup I(D_2)|$

Let S_i be the set of association rules in D_i, for $i = 1, 2, ..., n$. sim_2 is based on the items generated from S_i, for $i = 1, 2, ..., n$. Let $I(S_i)$ be the set of items generated from S_i, for $i = 1, 2, ..., n$. Similarity measure sim_2 has been defined as follows:

$sim_2(D_1, D_2) = |I(S_1) \cap I(S_2)| \,/\, |I(S_1) \cup I(S_2)|$

$I(S_i) \subseteq I(D_i)$, for $i = 1, 2, ..., n$. sim_1 estimates similarity between two databases more correctly than sim_2, since the number of items participate in estimating the similarity between two databases under sim_1 is more than that of sim_2. A database may not extract any association rule at a given value of (α, β), where β is the user-defined *minimum confidence level*. In such situations, the accuracy of sim_2 is low. In the following example, we discuss a situation where the accuracy of sim_1 and sim_2 are low.

Example 3.4.1. A multi-branch company possesses following three databases. DB_1 = { {a, b, c, e}, {a, b, d, f}, {b, c, g}, {b, d, g} }, DB_2 = { {a, g}, {b, e}, {c, f}, {d, g} }, and DB_3 = { {a, b, c}, {a, b, d}, {b, c}, {b, d, g} }. Here, $I(DB_1)$ = {a, b, c, d, e, f, g}, $I(DB_2)$ = {a, b, c, d, e, f, g}, $I(DB_3)$ = {a, b, c, d, g}. Thus, $sim_1(DB_1, DB_2) = 1.0$ (maximum), and $sim_1(DB_1, DB_3) = 0.71429$. Ground realities are as follows: (i) The similarity between DB_1 and DB_2 is low, since they contain dissimilar transactions. (ii) The similarity between DB_1 and DB_3 is more than the similarity between DB_1 and DB_2, since DB_1 and DB_3 contain similar transactions. Thus, the similarity measures sim_1 produces low accuracy in finding the similarity between two databases. There is no frequent itemsets in DB_2, if $\alpha > 0.25$. Thus, $I(S_2) = \phi$, if $\alpha > 0.25$. Hence, the accuracy of sim_2 is low in finding the similarity between DB_1 and DB_2, if $\alpha > 0.25$. •

Thus, we have observed that the similarity measures based on items in databases might not be appropriate in finding similarity between two databases. A more appropriate similarity measure could be designed based on frequent itemsets in both the databases. The frequent itemsets in two databases could find similarity among transactions in two databases better. Thus, frequent itemsets in two databases could find similarity between two databases more correctly.

Wu et al. [80] have proposed a solution of inverse frequent itemset mining. They argued that one could efficiently generate a synthetic market basket database from the frequent itemsets and their supports. Thus, the similarity between two databases could be estimated more correctly by involving supports of the frequent itemsets. We propose two measures of similarity based on the frequent itemsets and their supports. A new algorithm for clustering databases is designed based on a proposed measure of similarity.

The existing industry practice is to refresh a data warehouse on a periodic basis. Let λ be the periodicity of data warehouse refreshing. In this situation, an incremental mining algorithm [52] could be used to obtain updated supports of the existing frequent itemsets in a database on a periodic basis. But, there could be addition or, deletion of frequent itemsets over time. Thus, we need to mine the databases individually again on a periodic basis. Let Λ be the periodicity of data warehouse mining. The values of λ and Λ could be chosen suitably such that $\Lambda > \lambda$. Based on the updated local frequent itemsets, one could cluster the databases afresh.

Another alternative for taming multi-gigabyte data could be sampling. A commonly used technique for approximate query answering is sampling [19]. If an itemset is frequent in a large database then it is likely that the itemset is frequent in a sample database. Thus, one could analyze approximately a database by analyzing the frequent itemsets in a sample database.

The rest of the chapter is organized as follows. We formulate the problem in Section 3.4.2. In Section 3.4.3, we discuss work related to this problem. In Section 3.4.4, we cluster all the branch databases. The experimental results are presented in Section 3.4.5.

3.4.2 Problem statement

Let there are n branch databases. Also, let $FIS(D_i, \alpha)$ be the set of frequent itemsets corresponding to database D_i at a given value of α, for $i = 1, 2, \ldots, n$. Thus, our problem could be stated as follows.

Find the best non-trivial partition (if it exists) of $\{D_1, D_2, \ldots, D_n\}$ *using* $FIS(D_i, \alpha)$, *for* $i = 1, 2, \ldots, n$.

A partition [53] is a specific type of clustering. Formal definition of a non-trivial partition is given in Section 3.4.4.

3.4.3 Related work

Jain et al. [44] have presented an overview of clustering methods from a statistical pattern recognition perspective, with a goal of providing useful advice and references to fundamental concepts accessible to the broad community of clustering practitioners. Chan and Chong [25] have devised a novel quantitative model of non-textual World

Wide Web classification based on image information. A traditional clustering technique [90] is based on *metric* attributes. A *metric* attribute is one whose values can be represented by explicit coordinates in a Euclidean space. Thus, a traditional clustering technique might not work in this case, since we are interested in clustering databases. Ali et al. [15] have proposed a partial classification technique using association rules. The clustering of databases using local association rules might not be a good idea. The number of frequent itemsets obtained from a set of association rules might be much less than the number of frequent itemsets extracted using apriori algorithm [13]. Thus, the efficiency of the clustering process would be low. Liu et al. [55] have proposed multi-database mining technique that searches only the relevant databases. Identifying relevant databases is based on selecting the relevant tables (relations) that contain specific, reliable and statistically significant information pertaining to the query. Our study involves in clustering transactional databases. Yin and Han [86] have proposed a new strategy for relational heterogeneous database classification. This strategy might not be suitable for clustering transactional databases. Chen et al. [27] have proposed a method of discovering customer purchasing patterns by extracting associations or co-occurrences from stores' transactional databases.

In the context of similarity measures, Tan et al. [75] have presented an overview of twenty one interestingness measures proposed in the statistics, machine learning and data mining literature. Support and confidence measures [11] are used to identify frequently occurring association rules between two sets of items in large databases. In this chapter, we have presented two similarity measures. Our first measure, $simi_1$, is similar to measure Jaccard [75]. Measures such as support, interest [75], cosine [75] do not serve as good measures of similarity, since their denominators are not appropriate. Other measures in [75] might not be relevant in finding similarity between two databases.

Zhang et al. [91] designed a local pattern analysis for mining multiple databases. Zhang et al. [93] studied various strategies for mining multiple databases. For utilizing the low-cost information and knowledge on the internet, Su et al. [74] have proposed a logical framework for identifying quality knowledge from different data sources. It helps working towards the development of an agreed ontology.

Data mining serves as a tool for decision making as well as managing activities. Accurate estimations of software size in the early stages of a software project are critical in software project management because they lead to a good planning and reduce project costs. García et al. [36] have studied the relation between early software size measures as the function points and measures of the final product as the lines of code.

3.4.4 Clustering databases

Our approach of finding the best partition of a set of databases has been explained in the following steps: (i) Find $FIS(D_i, \alpha)$, for $i = 1, 2, \ldots, n$. (ii) Determine the similarity between each pair of databases using the proposed measure of similarity $simi_2$. (iii) Check for the existence of partitions at the required similarity levels (as mentioned in Theorem 3.4.5). (iv) Calculate the goodness values for all the non-trivial partitions. (v) Report the non-trivial partition for which the goodness value is the maximum. All the steps (i) to (v) will be followed and explained with the help of a running example. We start with an example of a multi-branch company that has multiple databases.

Example 3.4.2. A multi-branch company has seven branches. The branch databases are given below.

$D_1 = \{(a, b, c), (a, c), (a, c, d)\}$; $D_2 = \{(a, c), (a, b), (a, c, e)\}$; $D_3 = \{(a, e), (a, c, e), (a, b, c)\}$; $D_4 = \{(f, d), (f, d, h), (e, f, d), (e, f, h)\}$; $D_5 = \{(g, h, i), (i, j), (h, i), (i, j, g)\}$; $D_6 = \{(g, h, i), (i, j, h), (i, j)\}$; $D_7 = \{ (a, b), (g, h), (h, i), (h, i, j) \}$. The sets of frequent itemsets are given below.

$FIS(D_1, 0.35) = \{ (a, 1.0), (c, 1.0), (ac, 1.0) \}$; $FIS(D_2, 0.35) = \{ (a, 1.0), (c, 0.67), (ac, 0.67) \}$; $FIS(D_3, 0.35) = \{ (a, 1.0), (c, 0.67), (e, 0.67), (ac, 0.67), (ae, 0.67) \}$; $FIS(D_4, 0.35) = \{ (d, 0.75), (e, 0.5), (f, 1.0), (h, 0.5), (df, 0.75), , (ef, 0.5), (fh, 0.5) \}$; $FIS(D_5, 0.35) = \{ (g, 0.5), (h, 0.5), (i, 1.0)\}, (j, 0.5), , (gi, 0.5), (hi, 0.5), (ij, 0.5) \}$; $FIS(D_6, 0.35)$

= { (i, 1.0), (j, 0.67), (h, 0.67), (hi, 0.67), (ij, 0.67) }; $FIS(D_7, 0.35)$ = { (h, 0.75), (i, 0.5), (hi, 0.5) }. •

Based on the sets of frequent itemsets in a pair of databases, one could define many measures of similarity between them. We propose two measures of similarity between a pair of databases. Our first measure $simi_1$ [6] is defined as follows.

Definition 3.4.1. The measure of similarity $simi_1$ between databases D_1 and D_2 is defined as follows.

$$simi_1(D_1, D_2, \alpha) = \frac{|FIS(D_1, \alpha) \cap FIS(D_2, \alpha)|}{|FIS(D_1, \alpha) \cup FIS(D_2, \alpha)|},$$

where, the symbols $\cap$ and $\cup$ stand for the intersection and union operations of set theory, respectively. •

The similarity measure $simi_1$ is the ratio of the number frequent itemsets common to D_1 and D_2, and the total number of distinct frequent itemsets in D_1 and D_2. Frequent itemsets are the dominant patterns that determine major characteristics of a database. There are many implementations [32] of mining frequent itemsets in a database. Let X and Y be two frequent itemsets in database DB. The itemset X is more dominant than the itemset Y in DB if $supp(X, DB) > supp(Y, DB)$. Therefore, the characteristics of DB are revealed more by the pair (X, $supp(X, DB)$) than that of (Y, $supp(Y, DB)$). Thus, a good measure of similarity between two databases is a function of the supports of the frequent itemsets in the databases. Our second measure of similarity $simi_2$ [6] is defined as follows.

Definition 3.4.2. The measure of similarity $simi_2$ between databases D_1 and D_2 is defined as follows.

$$simi_2(D_1, D_2, \alpha) = \frac{\sum_{X \in \{FIS(D_1, \alpha) \cap FIS(D_2, \alpha)\}} minimum\{supp(X, D_1), supp(X, D_2)\}}{\sum_{X \in \{FIS(D_1, \alpha) \cup FIS(D_2, \alpha)\}} maximum\{supp(X, D_1), supp(X, D_2)\}},$$

where, the symbols $\cap$ and $\cup$ stand for the intersection and union operations of set theory, respectively. Assume that, $supp(X, D_i) = 0$, if $X \notin FIS(D_i, \alpha)$, for $i = 1, 2$. •

With reference to Example 3.4.1, the frequent itemsets in different databases are given as follows: $FIS(DB_1, 0.3)$ = { $a(0.5)$, $b(1.0)$, $c(0.5)$, $d(0.5)$, $g(0.5)$, $ab(0.5)$, $bc(0.5)$, $bd(0.5)$ }, $FIS(DB_2, 0.3)$ = { $g(0.5)$ }, and $FIS(DB_3, 0.3)$ = { $a(0.5)$, $b(1.0)$, $c(0.5)$, $d(0.5)$, $ab(0.5)$, $bc(0.5)$, $bd(0.5)$ }. We obtain $simi_1(DB_1, DB_2, 0.3) = 0.125$, $simi_1(DB_1, DB_3, 0.3) = 0.875$, $simi_2(DB_1, DB_2, 0.3) = 0.111$, and $simi_2(DB_1, DB_3, 0.3) = 0.889$. Thus, our proposed measures $simi_1$ and $simi_2$ match ground reality better than the existing measures. Theorem 3.4.1 justifies the fact that $simi_2$ matches ground reality better than $simi_1$.

Theorem 3.4.1. *The similarity measure $simi_2$ has better discriminating power than that of the similarity measure $simi_1$.*

Proof. The support of a frequent itemset could be considered as its weight in the database. But, we attach an weight 1.0 to itemset X in database D_i, under the similarity measure $simi_1$, if $X \in FIS(D_i, \alpha)$, for $i = 1, 2$. We attach an weight $supp(X, D_i)$ to the itemset X in database D_i, under the similarity measure $simi_2$, if $X \in FIS(D_i, \alpha)$, for $i = 1, 2$. The similarity measures sim_1 and sim_2 are defined as a ratio of two quantities. If $X \in FIS(D_i, \alpha)$, and $X \in FIS(D_j, \alpha)$, then it is more justifiable to add *minimum* { $supp(X, D_i)$, $supp(X, D_j)$ } (instead of 1.0) in the numerator and *maximum* { $supp(X, D_i)$, $supp(X, D_j)$ } (instead of 1.0) in the denominator for the itemset X, for $i, j \in \{1, 2\}$. If $X \in FIS(D_i, \alpha)$, and $X \notin FIS(D_j, \alpha)$, then it is more justifiable to add 0 in the numerator and $supp(X, D_i)$ (instead of 1.0) in the denominator for the itemset X, for $i, j \in \{1, 2\}$. Hence, the theorem follows. •

Example 3.4.3 verifies that $simi_2$ matches ground reality better than $simi_1$.

Example 3.4.3. With reference to Example 3.4.2, $supp(\{a\}, D_1) = supp(\{c\}, D_1) = supp(\{a, c\}, D_1) = 1.0$, $supp(\{a\}, D_2) = 1.0$, and $supp(\{c\}, D_2) = supp(\{a, c\}, D_2) = 0.67$. $simi_2(D_1, D_2, 0.35) = 0.78$, and $simi_1(D_1, D_2, 0.35) = 1.0$. We observe that the databases D_1 and D_2 are highly similar, but not the same. Thus, the similarity obtained by $simi_2$ matches the ground reality better. •

We study some interesting properties of *simi*$_1$ and *simi*$_2$ using Theorems 3.4.2, 3.4.3 and 3.4.4.

Theorem 3.4.2. *The similarity measure simi$_k$ satisfies the following properties,* (k = *1, 2*). (i) $0 \le simi_k(D_i, D_j, \alpha) \le 1$, (ii) $simi_k(D_i, D_j, \alpha) = simi_k(D_j, D_i, \alpha)$, (iii) $simi_k(D_i, D_i, \alpha) = 1$, *for* $i, j = 1, 2, ..., n$.

Proof. The properties follow from the definition of *simi*$_k$, (k = 1, 2). •

We express the distance between two databases in term of their similarity.

Definition 3.4.3. The distance measure *dist*$_k$ between two databases D_1 and D_2 based on the similarity measure *simi*$_k$ is defined as $dist_k(D_1, D_2, \alpha) = 1 - simi_k(D_1, D_2, \alpha)$, (k = 1, 2). •

A good distance measure satisfies the metric properties [21]. Higher the distance between two databases, lower is the similarity between them. For the purpose of elegant presentation, we use the notation I_i in place of $FIS(D_i, \alpha)$ in theorems Theorems 3.4.3 and 3.4.4, for i = 1, 2.

Theorem 3.4.3. *dist*$_1$ *is a metric over* [*0, 1*].

Proof. We show that *dist*$_1$ satisfies the triangular inequality. Other properties of a metric follow from Theorem 3.4.2.

$$dist_1(D_1, D_2, \alpha) = 1 - \frac{|I_1 \cap I_2|}{|I_1 \cup I_2|} \ge \frac{|I_1 - I_2| + |I_2 - I_1|}{|I_1 \cup I_2 \cup I_3|} \tag{3.4.1}$$

Thus, $dist_1(D_1, D_2, \alpha) + dist_1(D_2, D_3, \alpha) \ge \frac{|I_1 - I_2| + |I_2 - I_1| + |I_2 - I_3| + |I_3 - I_2|}{|I_1 \cup I_2 \cup I_3|}$ (3.4.2)

$$= \frac{|I_1 \cup I_2 \cup I_3| - |I_1 \cap I_2 \cap I_3| + |I_1 \cap I_3| + |I_2| - |I_1 \cap I_2| - |I_2 \cap I_3|}{|I_1 \cup I_2 \cup I_3|} \tag{3.4.3}$$

$$= 1 - \frac{|I_1 \cap I_2 \cap I_3| - |I_1 \cap I_3| - |I_2| + |I_1 \cap I_2| + |I_2 \cap I_3|}{|I_1 \cup I_2 \cup I_3|} \tag{3.4.4}$$

$$= 1 - \frac{\{|I_1 \cap I_2 \cap I_3| + |I_1 \cap I_2| + |I_2 \cap I_3|\} - \{|I_1 \cap I_3| + |I_2|\}}{|I_1 \cup I_2 \cup I_3|} \tag{3.4.5}$$

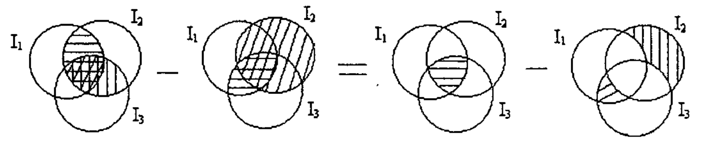


Figure 3.4.1. Simplification using Venn diagram

Let the number of elements in the shaded regions of Figures 3.4.1(c) and 3.4.1(d) be N_1 and N_2, respectively. Then the expression (3.4.5) becomes

$$1-\frac{N_1-N_2}{|I_1\cup I_2\cup I_3|}\geq\begin{cases}1-\dfrac{N_1-N_2}{|I_1\cup I_2\cup I_3|}, \text{if } N_1\geq N_2 & (\text{case }1)\\[2ex] 1-\dfrac{|I_1\cap I_3|}{|I_1\cup I_2\cup I_3|}, \text{if } N_1< N_2 & (\text{case }2)\end{cases} \tag{3.4.6}$$

In case 1, the expression remains the same. In case 2, a positive quantity $I_1\cap I_3$ has been put in place of a negative quantity N_1-N_2. Thus, the expression (3.4.6) is

$$\geq\begin{cases}1-\dfrac{N_1-N_2}{|I_1\cup I_3|}, \text{if } N_1\geq N_2\\[2ex] 1-\dfrac{|I_1\cap I_3|}{|I_1\cup I_3|}, \text{if } N_1< N_2\end{cases}\geq\begin{cases}1-\dfrac{N_1}{|I_1\cap I_3|}, \text{if } N_1\geq N_2\\[2ex] 1-\dfrac{|I_1\cap I_3|}{|I_1\cup I_3|}, \text{if } N_1< N_2\end{cases} \tag{3.4.7}$$

$$\geq\begin{cases}1-\dfrac{|I_1\cap I_3|}{|I_1\cup I_3|}, \text{if } N_1\geq N_2\\[2ex] 1-\dfrac{|I_1\cap I_3|}{|I_1\cup I_3|}, \text{if } N_1< N_2\end{cases}, \text{ where}, N_1=|I_1\cap I_2\cap I_3|\leq|I_1\cap I_3| \tag{3.4.8}$$

Therefore, irrespective of the relationship between N_1 and N_2, $dist_1(D_1, D_2, \alpha)$ + $dist_1(D_2, D_3, \alpha) \geq dist_1(D_1, D_3, \alpha)$. Thus, $dist_1$ satisfies the triangular inequality. •

We shall also show that $dist_2$ satisfies the metric properties.

Theorem 3.4.4. *$dist_2$ is a metric over* [0, 1].

Proof. We show that $dist_2$ satisfies the triangular inequality. Other properties of a metric follow from Theorem 3.4.2.

$$dist_2(D_1, D_2, \alpha) = 1 - \frac{\sum_{x \in I_1 \cap I_2} minimum\{supp(x, D_1), supp(x, D_2)\}}{\sum_{x \in I_1 \cup I_2} maximum\{supp(x, D_1), supp(x, D_2)\}} = 1 - \frac{\sum_{x \in I_1 \cap I_2} min_{12}(x)}{\sum_{x \in I_1 \cup I_2} max_{12}(x)} \tag{3.4.9}$$

where, $max_{ij}(x) = maximum\{\ supp(x, D_i), supp(x, D_j)\ \}$, and $min_{ij}(x) = minimum\{\ supp(x, D_i), supp(x, D_j)\ \}$, for $i \neq j$. Also, let $max_{123}(x) = maximum\{\ supp(x, D_1), supp(x, D_2), supp(x, D_3)\ \}$, and $min_{123}(x) = minimum\{\ supp(x, D_1), supp(x, D_2), supp(x, D_3)\ \}$.

Thus, $dist_2(D_1, D_2, \alpha) + dist_2(D_2, D_3, \alpha)$

$$= \frac{\sum_{x \in I_1 \cup I_2} max_{12}(x) - \sum_{x \in I_1 \cap I_2} min_{12}(x)}{\sum_{x \in I_1 \cup I_2} max_{12}(x)} + \frac{\sum_{x \in I_2 \cup I_3} max_{23}(x) - \sum_{x \in I_2 \cap I_3} min_{23}(x)}{\sum_{x \in I_2 \cup I_3} max_{23}(x)} \tag{3.4.10}$$

$$\geq \frac{\sum_{x \in I_1 - I_2} max_{12}(x) + \sum_{x \in I_2 - I_1} max_{12}(x)}{\sum_{x \in I_1 \cup I_2} max_{12}(x)} + \frac{\sum_{x \in I_2 - I_3} max_{23}(x) + \sum_{x \in I_3 - I_2} max_{23}(x)}{\sum_{x \in I_2 \cup I_3} max_{23}(x)} \tag{3.4.11}$$

$$\geq \frac{\sum_{x \in I_1 - I_2} max_{12}(x) + \sum_{x \in I_2 - I_1} max_{12}(x) + \sum_{x \in I_2 - I_3} max_{23}(x) + \sum_{x \in I_3 - I_2} max_{23}(x)}{\sum_{x \in I_1 \cup I_2 \cup I_3} max_{123}(x)} \tag{3.4.12}$$

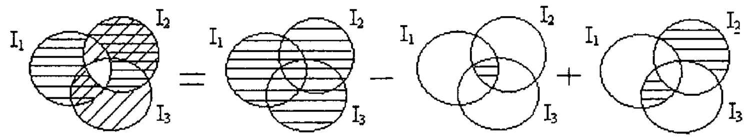


Figure 3.4.2. Simplification using Venn diagram

Using the simplification performed in Figure 3.4.2, the expression (3.4.12) becomes

$$\frac{\sum_{x \in I_1 \cup I_2 \cup I_3} max_{123}(x) - N_1 + N_2}{\sum_{x \in I_1 \cup I_2 \cup I_3} max_{123}(x)} \tag{3.4.13}$$

where, N_1 and N_2 are the value of $\sum_x max_{123}(x)$ over the shaded regions of Figures 3.4.2(c) and 3.4.2(d), respectively. The expression (3.4.13) is equal to

$$1 - \frac{N_1 - N_2}{\sum\limits_{x \in I_1 \cup I_2 \cup I_3} max_{123}(x)} \geq \begin{cases} 1 - \dfrac{N_1}{\sum\limits_{x \in I_1 \cup I_2 \cup I_3} max_{123}(x)}, \text{if } N_1 \geq N_2 \\ 1 - \dfrac{N_1 - N_2}{\sum\limits_{x \in I_1 \cup I_2 \cup I_3} max_{123}(x)}, \text{if } N_1 < N_2 \end{cases} \geq \begin{cases} 1 - \dfrac{\sum\limits_{x \in I_1 \cap I_3} max_{13}(x)}{\sum\limits_{x \in I_1 \cup I_2 \cup I_3} max_{123}(x)}, \text{if } N_1 \geq N_2 \\ 1 - \dfrac{\sum\limits_{x \in I_1 \cap I_3} max_{13}(x)}{\sum\limits_{x \in I_1 \cup I_2 \cup I_3} max_{123}(x)}, \text{if } N_1 < N_2 \end{cases} \quad (3.4.14)$$

Therefore, irrespective of the relationship between N_1 and N_2, $dist_2(D_1, D_2, \alpha) + dist_2(D_2, D_3, \alpha) \geq dist_2(D_1, D_3, \alpha)$. Thus, $dist_2$ satisfies the triangular inequality. •

Given a set of databases, the similarity between pairs of databases could be expressed by a square matrix, called *database similarity matrix* (DSM). We define DSM of a set of databases as follows.

Definition 3.4.4. Let $D = \{D_1, D_2, \ldots, D_n\}$ be the set of all databases. The database similarity matrix DSM_k of D using the measure of similarity $simi_k$, is a symmetric square matrix of size n by n, whose (i, j)-th element $DSM_k^{i,j}(D, \alpha) = simi_k(D_i, D_j, \alpha)$; for $D_i, D_j \in D$, and $i, j = 1, 2, \ldots, n$, $(k = 1, 2)$. •

For n databases, there are ${}^nC_2$ pairs of databases. For each pair of databases, we compute similarity between them. If the similarity is high then the databases may be put in the same class. We define a class as follows.

Definition 3.4.5. Let $D = \{D_1, D_2, \ldots, D_n\}$. A class $class_k^{\delta}$ formed at the level of similarity δ under the measure of similarity $simi_k$, is defined as $class_k^{\delta}(D, \alpha) = \begin{cases} P : P \subseteq D, |P| \geq 2, \text{and } simi_k(A, B, \alpha) \geq \delta, \text{for } A, B \in P \\ P : P \subseteq D, |P| = 1 \end{cases}, (k = 1, 2)$. •

A DSM could be viewed as a complete weighted graph. Each database forms a vertex. An weight of an edge is the similarity between the pair of concerned databases. Let $D = \{D_1, D_2, \ldots, D_n\}$ be the set of all databases. During the process of clustering, we assume that the databases $D_1, D_2, \ldots, D_r$ have been included in some classes, and the remaining databases are yet to be clustered. Then the clustering process forms the next class by finding a maximal complete sub-graph of the complete weighted graph containing vertices D_{r+1}, D_{r+2}, ..., D_n. A maximal complete sub-graph is defined as follows.

Definition 3.4.6. An weighted complete sub-graph g of a complete weighted graph G is maximal at the similarity level δ if the following conditions are true: (i) The weight of every edge of g is greater than or equal to δ. (ii) The addition of one more vertex (i.e., a database) to g leads to the addition of at least one edge to g having weight less than δ. •

We need to find out a maximal weighted complete sub-graph of the complete weighted graph of the remaining vertices to form the next class. This process continues till all the vertices are clustered. A clustering of databases could be defined as follows.

Definition 3.4.7. Let D be a set of databases. Let $\pi_k^\delta(D, \alpha)$ be a clustering of databases in D at the similarity level δ under the similarity measure $simi_k$. Then, $\pi_k^\delta(D, \alpha) = \{X: X \in \rho(D)$, and X is a $class_k^\delta(D, \alpha)\}$, where $\rho(D)$ is the power set of D, $(k = 1, 2)$. •

During the clustering process one may like to impose the restriction that each database belongs to at least one class. This restriction makes a clustering complete. We define a complete clustering as follows.

Definition 3.4.8. Let D be a set of databases. Let $\pi_k^\delta(D, \alpha) = \{C_{k,1}^\delta(D, \alpha), C_{k,2}^\delta(D, \alpha), \ldots, C_{k,m}^\delta(D, \alpha)\}$, where $C_{k,i}^\delta(D, \alpha)$ is the i-th class of π_k^δ, for $i = 1, 2, \ldots, m$. π_k^δ is complete, if $\bigcup_{i=1}^{m} C_{k,i}^\delta(D, \alpha) = D$, $(k = 1, 2)$. •

In a complete clustering, two classes may have a common database. One may be interested in finding out a clustering of mutually exclusive classes. A mutually exclusive clustering could be defined as follows.

Definition 3.4.9. Let D be a set of databases. Let $\pi_k^\delta(D, \alpha) = \{C_{k,1}^\delta(D, \alpha), C_{k,2}^\delta(D, \alpha), \ldots, C_{k,m}^\delta(D, \alpha)\}$, where $C_{k,i}^\delta(D, \alpha)$ is the i-th class of π_k^δ, for $i = 1, 2, \ldots, m$. π_k^δ is mutually exclusive if $C_{k,i}^\delta(D, \alpha) \cap C_{k,j}^\delta(D, \alpha) = \phi$, for $i \neq j$, $1 \leq i, j \leq m$, $(k = 1, 2)$. •

One may be interested in finding out such a mutually exclusive and complete clustering. A partition of a set of databases is defined as follows.

Definition 3.4.10. Let D be a set of databases. Also, let $\pi_k^{\delta}(D, \alpha)$ be a clustering of databases in D at the similarity level δ under the similarity measure $simi_k$. If $\pi_k^{\delta}(D, \alpha)$ is a mutually exclusive and complete clustering then it is called a partition ($k = 1, 2$). •

Definition 3.4.11. Let D be a set of databases. Also, let $\pi_k^{\delta}(D, \alpha)$ be a partition of D at the similarity level δ under the similarity measure $simi_k$. $\pi_k^{\delta}(D, \alpha)$ is called a non-trivial partition if $1 < |\pi_k^{\delta}| < n$ ($k = 1, 2$). •

A clustering is not necessarily be a partition. In the following example, we wish to find partitions (if they exist) of a set of databases.

Example 3.4.4. With reference to Example 3.4.2, consider the set of databases $D = \{D_1, D_2, \ldots, D_7\}$. The corresponding DSM_2 is given as follows.

$$DSM_2(D, 0.35) = \begin{bmatrix} 1.0 & 0.780 & 0.539 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.780 & 1.00 & 0.636 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.539 & 0.636 & 1.0 & 0.061 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.061 & 1.0 & 0.063 & 0.065 & 0.087 \\ 0.0 & 0.0 & 0.0 & 0.063 & 1.0 & 0.641 & 0.353 \\ 0.0 & 0.0 & 0.0 & 0.065 & 0.641 & 1.0 & 0.444 \\ 0.0 & 0.0 & 0.0 & 0.087 & 0.353 & 0.444 & 1.0 \end{bmatrix},$$

We arrange all non-zero and distinct $DSM_2^{i,j}(D, 0.35)$ values in non-increasing order, for $1 \leq i < j \leq 7$. The arranged similarity values are given as follows: 0.780, 0.641, 0.636, 0.539, 0.444, 0.353, 0.087, 0.065, 0.063, 0.061. We get many non-trivial partitions at different similarity levels. At similarity levels 0.780, 0.641, 0.539, and 0.353, we get non-trivial partitions as $\pi_2^{0.780} = \{ \{D_1, D_2\}, \{D_3\}, \{D_4\}, \{D_5\}, \{D_6\}, \{D_7\} \}$, $\pi_2^{0.641} = \{ \{D_1, D_2\}, \{D_3\}, \{D_4\}, \{D_5, D_6\}, \{D_7\} \}$, $\pi_2^{0.539} = \{ \{D_1, D_2, D_3\}, \{D_4\}, \{D_5, D_6\}, \{D_7\} \}$, and $\pi_2^{0.353} = \{ \{D_1, D_2, D_3\}, \{D_4\}, \{D_5, D_6, D_7\} \}$, respectively. •

Our *BestDatabasePartition* algorithm (as presented in Section 3.4.4.1) is based on binary similarity matrix (*BSM*). We derive binary similarity matrix BSM_k from the corresponding DSM_k ($k = 1, 2$). BSM_k is defined as follows.

Definition 3.4.12. The (i, j)-th element of the binary similarity matrix BSM_k at the similarity level δ using the similarity measure $simi_k$ is defined as follows.

$$BSM_k^{i,j}(D,\alpha,\delta) = \begin{cases} 1, \text{if } simi_k(D_i, D_j, \alpha) \geq \delta \\ 0, \text{otherwise} \end{cases}, \text{ for } i, j = 1, 2, \ldots, n\ (k = 1, 2). \bullet$$

We take an example of BSM_2 and observe the distribution of 0s and 1s.

Example 3.4.5. With reference to Example 3.4.4, the BSM_2 at similarity level 0.353 is given below.

$$BSM_2(D, 0.35, 0.353) = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix}, \text{ where } D = \{D_1, D_2, \ldots, D_7\}. \bullet$$

There may exist two same partitions at two distinct similarity levels. Two partitions are distinct if they are not the same. In the following, we define two same partitions at two distinct similarity levels.

Definition 3.4.13. Let D be a set of databases. Let $C \subseteq D$, and $C \neq \phi$. Two partitions $\pi_k^{\delta_1}$ (D, α) and $\pi_k^{\delta_2}$ (D, α) are the same, if the following statement is true: $C \in \pi_k^{\delta_1}$ if and only if $C \in \pi_k^{\delta_2}$, for $\delta_1 \neq \delta_2$. •

We would like to enumerate the maximum number of possible distinct partitions. In Theorem 3.4.5, we find the maximum number of possible distinct partitions of a set of databases [6].

Theorem 3.4.5. *Let D be a set of databases. Let m be the number of distinct non-zero similarity values in the upper triangle of DSM_2. Then the number of distinct partitions is less than or equal to m.*

Proof. We arrange the non-zero similarity values of the upper triangle of DSM_2 in non-increasing order. Let $\delta_1, \delta_2, \ldots, \delta_m$ be m non-zero ordered similarity values. Let δ_i, δ_{i+1} be two consecutive similarity values in the sequence of non-increasing similarity values. Let $x, y \in [\delta_i, \delta_{i+1})$, for some $i = 1, 2, \ldots, m$, where $\delta_{m+1} = 0$. Then $BSM_2(D, \alpha, x) = BSM_2(D, \alpha, y)$. Thus, there exists at the most one distinct non-trivial partition in the interval $[\delta_i, \delta_{i+1})$, for $i = 1, 2, \ldots, m$. We have m such semi-closed intervals $[\delta_i, \delta_{i+1})$, for $i = 1, 2, \ldots, m$. The theorem follows. •

For the purpose of finding partitions of the input databases, we shall first design a simple algorithm that uses apriori property [13]. The similarity values considered here are based on $simi_2$. Initially, we have n database classes, where n is the number of databases. At this time, each class contains a single database object. These classes are assumed at level 1. Based on the classes at level 1, we construct database classes at level 2. At level 1, we assume that the i-th class contains database D_i, for $i = 1, 2, \ldots, n$. i-th class and j-th class of level 1 could be merged if $simi_2(D_i, D_j) \geq \delta$, where δ is the user defined level of similarity. We proceed further until no more classes could be generated and no more levels could be generated. The algorithm [6] is presented below.

Algorithm 3.4.1. Partitions (if they exist) of a set of databases using apriori property.

```
procedure AprioriDatabaseClustering ($n$, $DSM_2$)
Input: $n$, $DSM_2$
$n$: number of databases, $DSM_2$: database similarity matrix
Output: Partitions (if they exist) of input databases
01: sort all the non-zero values that exist in the upper triangle of $DSM_2$ in non-increasing
02: order into an array called simValues; let the number of non-zero values be $m$;
03: let $k = 1$; let $simValues(m+1) = 0$; let $delta = simValues(k)$;
```

```
04: while ($delta > 0$) do
05:   construct $n$ classes, where each class contains a single database; // level: 1
06:   repeat line 7 until no more level could be generated;
07:     construct all possible classes at level ($i$ +1) using lines 8-10; // level: $i$ +1
08:       let $A$ and $B$ be two classes at the $i$-th level such that $|A \cap B| = i - 1$;
09:       let $a \in (A-B)$, and $b \in (B-A)$;
10:       if $DSM_2^{a,b} \geq \delta$ then construct a new class $A \cup B$; end if
11:   repeat line 12 from top level to level 1;
12:     for each class at the current level do
13:      if all databases of the current class are not included a class generated earlier then
14:         generate the current class;
15:       end if
16:     end for
17:     if the current clustering is a partition then store it; end if
18:     increase $k$ by 1; let $delta = simValues(k)$;
19: end while
20: display all the partitions;
end procedure
```

Lines 1-2 take O($m \times \log(m)$) time to sort m data. While-loop at line 4 executes m times. Line 5 takes O(n) time. Initially (at line 5), n classes are constructed. At the first iteration of line 6, the maximum number of classes generated is ${}^{n}C_{2}$. At the second iteration, the maximum number of classes generated is ${}^{n}C_{3}$. Lastly, at the (n-1)-th iteration, the maximum number of classes generated is ${}^{n}C_{n}$. Thus, the maximum number of possible classes is $O\left(\sum_{i=1}^{n} {}^{n}C_{i}\right)$, i.e., O($2^n$). Let p be the average size of a class. Line 8 takes O(p) time. Also, line 11 takes O(2^n) time, since the maximum number of possible classes is O(2^n). Thus, the time complexity of lines 4-19 is O($m \times p \times 2^n$). The line 20 takes time O($m \times n$), since the maximum number of partitions is m. Thus, the time complexity of the procedure *AprioriDatabaseClustering* is *maximum* {O($m \times \log(m)$), O($m \times p \times 2^n$), O($m \times n$)}, i.e., O($m \times p \times 2^n$), since $p \times 2^n > 2^n > n^2 > m > \log_2(m)$, for $p > 1$ and $n > 4$. The *AprioriDatabaseClustering* algorithm generates all possible classes level-wise. It is a simple but not an efficient clustering technique, since the time-complexity of the algorithm is an exponential function of n.

In Theorems 3.4.6, 3.4.7, 3.4.8, and 3.4.9, we discuss some properties of BSM_2.

Theorem 3.4.6. *Let* $D = \{D_1, D_2, ..., D_n\}$. *Let* $\pi_2^\delta(D, \alpha)$ *be a clustering of databases in D at the similarity level* δ. π_2^δ *is a partition if and only if the corresponding* BSM_2 *gets transformed into the following form by inter-changing jointly a row and the corresponding column with another row and the corresponding column.*

$$\begin{bmatrix} U_1 & 0 & \dots & 0 \\ 0 & U_2 & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & U_m \end{bmatrix},$$ *U_i is a matrix of size $n_i \times n_i$, containing all elements as 1,*

$\sum_{i=1}^{m} n_i = n$, $|\pi_2^\delta| = m$.

Proof. Let { D_1^i, D_2^i, ..., $D_{n_i}^i$ } be the i-th database class of the partition at the similarity level δ, for $i = 1, 2, \dots, m$. The row corresponding to D_j^i of BSM_2 corresponds to a unique combination of 0s and 1s, for $j = 1, 2, \dots, n_i$. Similarly, the column corresponding to D_j^i of BSM_2 corresponds to a unique combination of 0s and 1s, for $j = 1, 2, \dots, n_i$. All such n_i rows and columns may not be consecutive initially, for $i = 1, 2, \dots, m$. We shall keep these n_i rows and columns consecutive, for $i = 1, 2, \dots, m$. Initially, we keep n_1 rows and the corresponding columns of the first database class consecutively. Then, we keep n_2 rows and the corresponding columns of the second database class consecutively, and so on. In general, to fix the matrix U_i at the proper position, we interchange jointly $\left(\sum_1^{i-1} n_j + k\right)$-th row and $\left(\sum_1^{i-1} n_j + k\right)$-th column with D_k^i-th row and D_k^i-th column of BSM_2, for $1 \le k \le n_i$, for $i = 1, 2, \dots, m$. •

With reference to BSM_2 in Example 3.4.5, we apply Theorem 3.4.6, and conclude that a partition exists at similarity level 0.353.

Theorem 3.4.7. *Let D be a set of databases. Let $\pi_2^{\delta}(D, \alpha)$ be a clustering of databases in D at the similarity level δ. Let $\{ D_1^i, D_2^i, ..., D_{n_i}^i \}$ be the i-th database class of $\pi_2^{\delta}(D, \alpha)$. Then D_k^i-th row (or, D_k^i-th column) of BSM_2 contains n_i 1s, for $k = 1, 2, ..., n_i$, $i = 1, 2, ..., | \pi_2^{\delta} |$.*

Proof. If possible, let D_k^i-th row or, D_k^i-th column has $(n_i + 1)$ 1s. Then D_k^i-th database would belong to two database classes. It contradicts mutually exclusiveness of classes of a partition. If possible, let D_k^i-th row or, D_k^i-th column contains $(n_i - 1)$ 1s. It contradicts the fact that $BSM_2^{D_j^i, D_k^i} = 1$, for $j = 1, 2, ..., n_i$, and $j \neq k$. •

Theorem 3.4.8. *Let D be a set of databases. Let $\pi_2^{\delta}(D, \alpha)$ be a clustering of databases in D at the similarity level δ. Then, the rank of the corresponding BSM_2 is $| \pi_2^{\delta} |$.*

Proof. Let $\{ D_1^i, D_2^i, ..., D_{n_i}^i \}$ be the i-th database class of π_2^{δ}. Then,

$$BSM_2^{D_j^i, D_k^i}(D, \alpha) = \begin{cases} 1, & \text{for } D_j^i, D_k^i \in \{ D_1^i, D_2^i, ..., D_{n_i}^i \} \\ 0, & \text{for } D_j^i \in \{ D_1^i, D_2^i, ..., D_{n_i}^i \} \text{ and } D_k^i \notin \{ D_1^i, D_2^i, ..., D_{n_i}^i \} \\ 0, & \text{for } D_j^i \notin \{ D_1^i, D_2^i, ..., D_{n_i}^i \} \text{ and } D_k^i \in \{ D_1^i, D_2^i, ..., D_{n_i}^i \} \end{cases}$$

The row corresponding to D_j^i of BSM_2 corresponds to a unique combination of 0s and 1s, for $j = 1, 2, ..., n_i$. So, all the rows of BSM_2 are divided into $|\pi_2^{\delta}|$ groups such that all the rows in a group correspond to a unique combination of 0s and 1s. Thus, BSM_2 has $|\pi_2^{\delta}|$ independent rows. •

Theorem 3.4.9. *Let $D = \{D_1, D_2, ..., D_n\}$. At a given value of the triplet (D, α, δ), there exists at the most one partition of D.*

Proof. At a given value of the pair (D, α), the element $DSM_2^{i,j}$ is unique, for $i, j = 1, 2, ..., n$. Thus, at a given value of the tuple (D, α, δ) the element $BSM_2^{i,j}$ is unique, for $i, j =$

1, 2, ..., *n*. There exists a partition if the BSM_2 gets transformed into a specific form (as mentioned in Theorem 3.4.6), by inter-changing jointly a row and the corresponding column with another row and the corresponding column. Hence, the theorem follows. •

3.4.4.1 Finding the best non-trivial partition

We return back to Example 3.4.4. We observe that at different similarity levels there may exist different partitions. We have observed the existence of four non-trivial partitions. We would like to find the best partition among these partitions. The best partition is based on the principle of maximizing the intra-class similarity and maximizing the inter-class distance. The intra-class similarity and inter-class distance are defined as follows.

Definition 3.4.14. The intra-class similarity *intra-sim* of a partition π at the similarity level δ using the similarity measure $simi_2$ is defined as follows:

$$intra-sim(\pi_2^\delta) = \sum_{C \in \pi_2^\delta} \sum_{D_i, D_j \in C; i<j} simi_2(D_i, D_j, \alpha). \bullet$$

Definition 3.4.15. The inter-class distance *inter-dist* of a partition π at the similarity level δ using the similarity measure $simi_2$ is defined as follows:

$$inter-dist(\pi_2^\delta) = \sum_{C_p, C_q \in \pi_2^\delta;\, p<q} \sum_{D_i \in C_p;\, D_j \in C_q;\, i<j} dist_2(D_i, D_j, \alpha). \bullet$$

The best partition among a set of partitions is selected on the basis of goodness value of a partition. The goodness measure *goodness* of a partition is defined as follows.

Definition 3.4.16. The goodness of a partition π at similarity level δ using the similarity measure $simi_2$ is defined as follows: $goodness(\pi_2^\delta)$ = $intra\text{-}sim(\pi_2^\delta)$ + $inter\text{-}dist\ (\pi_2^\delta)$ - $|\pi_2^\delta|$, where $|\pi_2^\delta|$ is the number classes in π. •

We have subtracted $|\pi_2^\delta|$ from the sum of intra-class similarity and inter-class distance to remove the bias of goodness value of a partition. Higher the value of *goodness*, better is the partition. Now, we would like to partition the set of databases D using the proposed goodness measure.

Example 3.4.6. With reference to Example 3.4.4, we would like to calculate the goodness value of each of the non-trivial partitions using $simi_2$.

$intra\text{-}sim(\pi_2^{0.353}) = 3.185$, $inter\text{-}dist(\pi_2^{0.353}) = 15.276$, $|\pi_2^{0.353}| = 3$, and $goodness(\pi_2^{0.353}) = 15.461$. $intra\text{-}sim(\pi_2^{0.539}) = 2.596$, $inter\text{-}dist(\pi_2^{0.539}) = 16.666$, $|\pi_2^{0.539}| = 4$, and $goodness(\pi_2^{0.539}) = 15.262$. $intra\text{-}sim(\pi_2^{0.641}) = 1.421$, $inter\text{-}dist(\pi_2^{0.641}) = 17.491$, $|\pi_2^{0.641}| = 5$, and $goodness(\pi_2^{0.641}) = 13.912$. $intra\text{-}sim(\pi_2^{0.780}) = 0.780$, $inter\text{-}dist(\pi_2^{0.780}) = 17.118$, $|\pi_2^{0.780}| = 6$, and $goodness(\pi_2^{0.780}) = 11.898$.

The goodness value corresponding to the partition $\pi_2^{0.353}$ is the maximum. Thus, the partition $\pi_2^{0.353} = \{ \{D_1, D_2, D_3\}, \{D_4\}, \{D_5, D_6, D_7\} \}$ is the best among all the non-trivial partitions. Let us look back into the databases of Example 3.4.2. We find that the partition $\pi_2^{0.353}$ matches the ground reality best among the partitions reported. •

We shall now present an algorithm [6] for finding the best non-trivial partition of a set of databases.

Algorithm 3.4.2. Best non-trivial partition (if it exists) of a set of databases.

```
procedure BestDatabasePartition (n, DSM_2)
Input: n, DSM_2
n: number of databases, DSM_2: database similarity matrix
Output: The best partition (if it exists) of input databases
01: sort all the non-zero values that exist in the upper triangle of DSM_2 in non-increas-
02: ing order into an array called simValues; let the number of non-zero values be m;
03: let k = 1; let simValues(m + 1) = 0; let delta = simValues(k);
04: while (delta > 0) do
05:     for i = 1 to n do class(i) = 0; end for
```

```
06:     construct the $BSM_2$ at current level of the similarity delta;
07:     let currentClass = 1; let currentRow = 1; let class(1) = currentClass;
08:     for col = (currentRow + 1) to n do
09:        if ($BSM_2^{currentRow, col}$ = 1) then
10:           if (class(col) = 0) then class(col) = currentClass;
11:           else if (class(col) ≠ currentClass) then go to line 24; end if
12:           end if
13:        end if
14:     end for
15:     let i = 1; let class(n +1) = 0;
16:     while (class(i) ≠ 0) do increase i by 1; end while
17:     if (i = n +1) then
18:        store the content of array class and current similarity level delta;
19:     else
20:       increase currentRow by 1;
21:        if (class(currentRow) = 0) then increase currentClass by 1; end if
22:        go to line 8;
23:     end if
24:     increase k by 1; let delta = simValues(k);
25:   end while
26:   for each non-trivial partition do
27:     calculate the goodness value of the current partition;
28:   end for
29:   return the partition whose goodness value is the maximum;
end procedure
```

We have sorted all non-zero values in the upper triangle of DSM_2 in non-increasing order at step 1. Thus, the algorithm checks the existence of a partition starting with the maximum of all the similarity values. At line 5, we initialize the class label of each database to 0. The algorithm starts forming a class with D_1 (the first database) as the variable *currentRow* is initialized with 1. Also, class label starts with 1 as the variable *currentClass* is initialized with 1. Lines 8-14 are used to check the similarity of $D_{currentRow}$ with other databases. If the condition at line 9 is true then databases $D_{currentRow}$ and D_{col} are similar. At line 10, D_{col} is put in the *currentClass* if it is still unlabelled. If D_{col} is already labeled with a class label not equal to current class label then D_{col} get another label. Thus, partition does not exist at the current similarity level. Some useful explanations of the algorithm are given in Theorem 3.4.10.

Line 1 takes O($m \times \log(m)$) time. Line 3 repeats for m times. Line 6 constructs BSM_2 in O(n^2) time as the order of BSM_2 is $n \times n$. Each of lines 5 and 16 takes O(n) time. For-loop at line 8, repeats maximum n times. Line 18 takes O(n) time, since the time required to store a partition is O(n). Thus, the time complexity of lines 4-25 is O($m \times n^2$). Therefore, the time complexity of the procedure *best-database-partition* is *maximum* { O($m \times \log(m)$), O($m \times n^2$) }, i.e., O($m \times n^2$), since $n^2 > m > \log_2(m)$.

Our *BestDatabasePartition* algorithm performs better than the *BestClassification* algorithm [83]. *BestClassification* algorithm has the following drawbacks: (i) Step value for assigning the next similarity level is user-input. Thus, it fails to find the exact similarity level at which a partition exists. (ii) The algorithm *BestClassification* calls procedure *GreedyClass* [83] at different places. There is a mistake in the procedure *GreedyClass*. The following example shows that the procedure *GreedyClass* fails to construct the correct classes at a given level of similarity.

Example 3.4.7. A multi-branch company has four branch databases D_1, D_2, D_3, and D_4. Let D = {D_1, D_2, D_3, D_4}, and $\alpha = 0.05$. Assume that the corresponding DSM_2 has the following form.

$$DSM_2(D, 0.05) = \begin{bmatrix} 1.0 & 0.1 & 0.1 & 0.4 \\ 0.1 & 1.0 & 0.1 & 0.1 \\ 0.1 & 0.1 & 1.0 & 0.5 \\ 0.4 & 0.1 & 0.5 & 1.0 \end{bmatrix}$$

At the similarity level 0.3, we should get clustering as $\{\{D_1, D_4\}, \{D_2\}, \{D_3, D_4\}\}$. But, the *GreedyClass* procedure generates clustering as $\{\{D_1, D_4\}, \{D_2, D_4\}, \{D_3, D_4\}\}$. The class $\{D_2, D_4\}$ should not be formed, since $simi_2(D_2, D_4, 0.05) < 0.3$. •

The proposed *BestDatabasePartition* algorithm reports the exact similarity level at which a partition exists. Also, the algorithm works faster, since it is required to check for the existence of partitions only at m similarity levels. In Theorem 3.4.10, we prove that the proposed algorithm works correctly.

Theorem 3.4.10. *Algorithm BestDatabasePartition works correctly.*

Proof. Let $D = \{D_1, D_2, ..., D_n\}$. Let there are m distinct non-zero similarity values in the upper triangle of DSM_2. Using Theorem 3.4.5, we could conclude that the maximum number of partitions of D is m at a given value of pair (D, α). While-loop at line 4 checks for the existence of partitions at m similarity levels. At each similarity level, we get a new BSM_2. The existence of a partition is determined from the BSM_2. We have an array *class* that stores the class label given to each database under the current level of similarity. In a partition, each database has a unique class label. The existence of a partition is checked based on the principle that every database receives a unique class label. As soon as we find that a labeled database receives another class label, we conclude that a partition does not exist at the current level of similarity *delta* (at line 11). Initially, we put the class label 0 to all databases using line 5. Then, we start from the row 1 of BSM_2 that corresponds to database D_1. Thus, D_1 is kept in the first database class. If there is a 1 in the j-th column of BSM_2, then we put class label of D_j as 1 using line 10. We find a database D_i that has not been clustered yet using lines 15-16. Then, we start at row i of BSM_2. If there is a 1 in the j-the column of row i, then we put database D_j in the current class. Thus, the algorithm *BestDatabasePartition* works correctly. •

3.4.4.2 Efficiency of clustering technique

The proposed clustering algorithm is based on the similarity measure $simi_2$. Also, the similarity measure $simi_2$ is based on supports of the frequent itemsets in databases. If we vary the value of α then the number of frequent itemsets in a database varies. The accuracy of similarity between two databases increases as the number of frequent itemsets increases. Therefore, a clustering process would be more accurate at a smaller value of α. The frequent itemsets participate in the clustering process is limited by main memory. If we can store more frequent itemsets in main memory then $simi_2$ could determine similarity between two databases more accurately. Thus, the clustering process would be more accurate. This limitation begs a space efficient representation of the frequent itemsets in main memory. For this purpose, we propose a coding for representing frequent itemsets space efficiently. The coding allows more frequent itemsets to participate in determining the similarity between two databases.

3.4.4.2.1 *Space efficient representation of frequent itemsets in different branch databases*

In this technique, we represent each frequent itemset using a bit vector. Each frequent itemset has three components: database identification, frequent itemset, and support. Let the number of databases be n. There exists an integer p such that $2^{p-1} < n \leq 2^p$. Then p bits are enough to represent a database. Let k be the number of digits after the decimal point to represent support. Support value 1.0 could be represented as 0.99999, for $k = 5$. If we represent the support s as an integer d containing of k digits then $s = d \times 10^{-k}$. The number digits required to represent a decimal number could be obtained by Theorem 3.4.11.

Theorem 3.4.11. *A p-digit decimal number can be represented by a* $\lceil p \times log_2 10 \rceil$ *-digit binary number.*

Proof. Let t be the minimum number of binary digits required to represent a p-digit decimal number x. Then $x < 10^p < 2^t$. So, $t > p \times log_2 10$, since $log_k(y)$ is a monotonic increasing function of y, for $k > 1$. Thus, we find the minimum integer t for which $x < 2^t$ is true as $\lceil p \times log_2 10 \rceil$. •

The proposed coding is described with the help of Example 3.4.8.

Example 3.4.8. We refer to Example 3.4.2. Sorted the frequent itemsets on number of extractions in non-increasing order are given as follows: (h, 4), (a, 3), (ac, 3), (c, 3), (hi, 3), (*i*, 3), (*ae*, 2), (*e*, 2), (*ij*, 2), (*ab*, 1), (*b*, 1), (*d*, 1), (*df*, 1), (*ef*, 1), (*f*, 1), (*fh*, 1), (*g*, 1), (*gi*, 1), (*j*, 1). (*X*, μ) denotes itemset *X* having number of extractions μ. We code these frequent itemsets left to right. These frequent itemsets are coded using a technique similar to Huffman coding [43]. We attach code 0 to itemset *h*, 1 to itemset *a*, 00 to itemset *ac*, 01 to itemset *c*, etc. Itemset *h* gets a code of minimal length, since it has been extracted maximum number of times. We call this coding as itemset (IS) coding. It is a lossless coding [67]. IS coding and Huffman coding are not the same, in the sense that an IS code may be a prefix of another IS code. Coded itemsets are given as follows: (*h*, 0), (*a*, 1), (*ac*, 00), (*c*, 01), (*hi*, 10), (*i*, 11), (*ae*, 000), (*e*, 001), (*ij*, 010), (*ab*, 011), (*b*, 100), (*d*, 101), (*df*, 110), (*ef*, 1111), (*f*, 0000), (*fh*, 0001), (*g*, 0010), (*gi*, 0011), (*j*, 0100). (*X*, ν) denotes itemset *X* having IS code ν. •

3.4.4.2.2 *Efficiency of IS coding*

Using the above representation of the frequent itemsets, one could store more frequent itemsets in the main memory during the clustering process. Thus, it enhances the efficiency of the clustering process.

Definition 17. Let there are *n* databases D_1, D_2, ..., D_n. Let $S^T\left(\bigcup_{i=1}^{n} FIS(D_i)\right)$ be the amount of storage space (in bits) required to represent $\bigcup_{i=1}^{n} FIS(D_i)$ by a technique *T*. Let $S_{min}\left(\bigcup_{i=1}^{n} FIS(D_i)\right)$ be the minimum amount of storage space (in bits) required to represent $\bigcup_{i=1}^{n} FIS(D_i)$. Let τ, κ, and λ denote a clustering algorithm, similarity measure, and computing resource under consideration, respectively. Let Γ be the set of all frequent itemset representation techniques. We define efficiency of a frequent itemset representation technique *T* at a given value of triplet (τ, κ, λ) as follows.

$\varepsilon(T \mid \tau, \kappa, \lambda) = S_{min}\left(\bigcup_{i=1}^{n} FIS(D_i)\right) / S^T\left(\bigcup_{i=1}^{n} FIS(D_i)\right)$, for $T \in \Gamma$. •

One could store an itemset conveniently using the following components: database identification, items in the itemset, and support. Database identification, an item and a support could be stored as a short integer, an integer and a real type data, respectively. A typical compiler represents a short integer, an integer and a real number using 2 bytes, 4 bytes and 8 bytes, respectively. Thus, a frequent itemset of size 2 could consume $(2 + 2 \times 4 + 8) \times 8$ bits, i.e. 144 bits. An itemset representation may have an overhead of indexing frequent itemsets. Let $OI(T)$ be the overhead of indexing coded frequent itemsets using technique T.

Theorem 3.4.12. *IS coding stores a set of frequent itemsets using minimum storage space, if OI(IS coding) $\le OI(T)$, for $T \in \Gamma$.*

Proof. A frequent itemset has three components, viz., database identification, itemset, and support. Let the number of databases be n. Then, $2^{p-1} < n \le 2^p$, for an integer p. We need minimum p bits to represent a database. The representation of database identification is independent of the corresponding frequent itemsets. If we keep k digits to store a support then $\lceil k \times \log_2 10 \rceil$ binary digits are needed to represent a support (as mentioned in Theorem 3.4.11). Thus, the representation of support becomes independent of the other components of the frequent itemset. Also, the sum of all IS codes is the minimum because of the way they are constructed. Thus, the space consumed by IS coding for representing a set of frequent itemsets is the minimum. •

Thus, the efficiency of a frequent itemset representation technique T could be expressed as follows:

$\varepsilon(T | \tau, \kappa, \lambda) = S^{\text{IS coding}}\left(\bigcup_{i=1}^{n} FIS(D_i)\right) / S^{T}\left(\bigcup_{i=1}^{n} FIS(D_i)\right)$, provided OI(IS coding) $\le OI(T)$, for $T \in \Gamma$. (3.4.15)

If the condition in (3.4.15) is satisfied, then IS coding performs better than any other techniques. If the condition in (3.4.15) is not satisfied, then IS coding performs better than any other techniques in almost all cases. The following corollary is derived from Theorem 3.4.12.

Corollary 3.4.1. *Efficiency of IS coding is maximum, if OI(IS coding) $\le$ OI(T), for $T \in \Gamma$.*

Proof. ε(IS coding | τ, κ, λ) = 1.0. •

IS coding maintains an index table to decode/search a frequent itemset. In the following example, we compute the amount of space required to represent the frequent itemsets using an ordinary method and IS coding.

Example 3.4.9. With reference to Example 3.4.8, there are 35 frequent itemsets. Among them, there are 20 itemsets of size 1 and 14 itemsets of size 2. Thus, an ordinary method could consume ($112 \times 20 + 144 \times 15$) bits, i.e., 4400 bits. The amount of space required to represent frequent itemsets in seven databases using IS coding is equal to $P + Q$ bits, where P is the amount of space required to store frequent itemsets, and Q is the amount of space required to maintain the index table. Since there are seven databases, we need 3 bits to identify a database. The amount of memory required to represent the database identification for 35 frequent itemsets is equal to 35×3 bits = 105 bits. Suppose we keep 5 digits after the decimal point for a support. Thus, $\lceil 5 \times \log_2(10) \rceil$ bits, i.e., 17 bits are required to represent a support. The amount of memory required to represent the supports of 35 frequent itemsets is equal to 35×17 bits = 595 bits. Let the number of items be 10000. Therefore, 14 bits are required to identify an item. The amount of storage space would require for itemsets *h* and *ac* are 14 bits and 28 bits, respectively. To represent 35 frequent itemsets, we need ($20 \times 14 + 15 \times 28$) bits = 700 bits. Thus, P = (105 + 595 + 700) bits = 1400 bits. There are 19 frequent itemsets. Using IS coding, 19 frequent itemsets consume 54 bits. To represent 19 frequent itemsets, we need $14 \times 10 + 28 \times 9$ bits, i.e., 392 bits. Thus, Q = 392 + 54 = 446 bits. The total amount of memory space required (including the overhead of indexing) to represent frequent itemsets in 7 databases using IS coding is equal to $P + Q$ bits, i.e., 1846 bits. The amount of space saving in compared to an ordinary method is equal to 2554 bits, i.e., 58% approximately.

A technique without optimization (TWO) may not maintain index table separately. In this case, *OI*(TWO) = 0. In spite of that, IS coding performs better than a TWO in most of the cases. •

Finally, we claim that our clustering technique is more accurate. There are two reasons for this claim: (i) We propose more appropriate measures of similarity than the existing measures. Thus, the similarity between two databases is estimated more accurately. (ii) Also, the proposed IS coding enables us to mine local databases further at a lower level of α to accommodate more frequent itemsets in main memory. As a result, more frequent itemsets could participate in the clustering process.

3.4.5 Experiments

We have carried out several experiments to study the effectiveness of our approach. All the experiments have been implemented on a 1.6 GHz Pentium processor with 256 MB of memory, using visual C++ (version 6.0) software. We present experimental results using two synthetic databases, and one real database. The synthetic databases *T10I4D100K* [34] and *T40I10D100K* [34] have been generated using synthetic database generator from the IBM Almaden Quest research group. The real database *BMS-Web-Wiew-1* [47] could be found from KDD CUP 2000. We present some characteristics of these databases in Table 3.4.1. Let *NT*, *ALT*, *AFI*, and *NI* denote the number of transactions, the average length of a transaction, the average frequency of an item, and the number of items in the database (*DB*), respectively.

Table 3.4.1. Database characteristics

DB	*NT*	*ALT*	*AFI*	*NI*
T10I4D100K (T_1)	1,00,000	11.102280	1276.124138	870
T40I10D100K (T_4)	1,00,000	40.605070	4310.516985	942
BMS-Web-Wiew-1 (B_1)	1,49,639	2.000000	155.711759	1922

Each of the above databases is divided into 10 databases for the purpose of carrying out experiments. The databases obtained from *T10I4D100K*, and *T40I10D100K* are named as T_{1j}, and T_{4j}, respectively, for j = 0, 1, ..., 9. The databases obtained from *BMS-Web-Wiew-1* are named as B_{1j}, for j = 0, 1, ..., 9. The databases T_{ij} and B_{1j} are called input databases, for i = 1, 4, and j = 0, 1, ..., 9. Some characteristics of these input databases are presented in the Table 3.4.2.

Table 3.4.2. Input database characteristics

DB	*NT*	*ALT*	*AFI*	*NI*	*DB*	*NT*	*ALT*	*AFI*	*NI*
T_{10}	10000	11.05500	127.65589	866	T_{15}	10000	11.13910	128.62702	866
T_{11}	10000	11.13330	128.41177	867	T_{16}	10000	11.10780	128.56250	864
T_{12}	10000	11.06700	127.64706	867	T_{17}	10000	11.09840	128.45370	864
T_{13}	10000	11.12260	128.43649	866	T_{18}	10000	11.08150	128.55568	862
T_{14}	10000	11.13670	128.74798	865	T_{19}	10000	11.08140	128.10867	865
T_{40}	10000	40.56710	431.56489	940	T_{45}	10000	40.50630	430.46015	941
T_{41}	10000	40.58240	432.18743	939	T_{46}	10000	40.74350	433.44149	940
T_{42}	10000	40.63190	431.79490	941	T_{47}	10000	40.62380	431.70882	941
T_{43}	10000	40.62690	431.74176	941	T_{48}	10000	40.52810	431.15000	940
T_{44}	10000	40.66110	432.56489	940	T_{49}	10000	40.57960	432.15761	939
B_{10}	14000	2.0000	14.94130	1874	B_{15}	14000	2.0000	280.00000	100
B_{11}	14000	2.0000	280.00000	100	B_{16}	14000	2.0000	280.00000	100
B_{12}	14000	2.0000	280.00000	100	B_{17}	14000	2.0000	280.00000	100
B_{13}	14000	2.0000	280.00000	100	B_{18}	14000	2.0000	280.00000	100
B_{14}	14000	2.0000	280.00000	100	B_{19}	23639	2.0000	472.78000	100

At a given value of α, there may exist many partitions. Partitions of the set of input databases are presented in Table 3.4.3. If we vary the value of α, the set of frequent itemsets in a database varies. Thus, the similarity between a pair of databases changes over the change of α.

Table 3.4.3. Partitions of the input databases at a given value of α

Databases	α	Non-trivial distinct partition (π)	δ	*goodness* (π)
$\{T_{10}, \ldots, T_{19}\}$	0.03	$\{\{T_{10}\},\{T_{11}\},\{T_{12}\},\{T_{13}\},\{T_{14},T_{18}\}, \{T_{15}\},\{T_{16}\},\{T_{17}\},\{T_{19}\}\}$	0.880798	0.011031
$\{T_{40}, \ldots, T_{49}\}$	0.1	$\{\{T_{40}\},\{T_{41},T_{45}\},\{T_{42}\},\{T_{43}\}, \{T_{44}\},\{T_{46}\},\{T_{47}\},\{T_{48}\},\{T_{49}\}\}$	0.949743	-3.977703
		$\{\{T_{40}\},\{T_{41}, T_{45}\},\{T_{42}\},\{T_{43}\}, \{T_{44}\},\{T_{46}\},\{T_{47}\},\{T_{48},T_{49}\}\}$	0.943098	11.716271
		$\{\{T_{40}\}, \{T_{41}, T_{43}, T_{45}\}, \{T_{42}\}, \{T_{44}\}, \{T_{46}\}, \{T_{47}\}, \{T_{48}, T_{49}\}\}$	0.942427	24.206474
$\{B_{10},\ldots, B_{19}\}$	0.009	$\{\{B_{10}\},\{B_{11}\},\{B_{12},B_{14}\},\{B_{13}\},\{B_{15}\}, \{B_{16}\},\{B_{17}\},\{B_{18}\},\{B_{19}\}\}$	0.726502	11.702650
		$\{\{B_{10}\},\{B_{11}\},\{B_{12},B_{14}\},\{B_{13}\},\{B_{15}\}, \{B_{16},B_{19}\},\{B_{17}\},\{B_{18}\}\}$	0.698836	27.694834
		$\{\{B_{10}\},\{B_{11}\},\{B_{12},B_{13}, B_{14}\}, \{B_{15}\}, \{B_{16}, B_{19}\},\{B_{17}\}, \{B_{18}\}\}$	0.684409	36.970604
		$\{\{B_{10}\}, \{B_{11}\}, \{B_{12}, B_{13}, B_{14}, B_{15}, B_{16}, B_{19}, B_{17}, B_{18}\}\}$	0.582443	55.984833
		$\{\{B_{10}, B_{11}\}, \{B_{12}, B_{13}, B_{14}, B_{15}, B_{16}, B_{17}, B_{18}, B_{19}\}\}$	0.535796	81.028792

At a smaller value of α, more frequent itemsets are reported from a database. So, we get a more accurate value of similarity between a pair of databases. Thus, the partition generated at a smaller value of α would be more correct. In Tables 3.4.4 and 3.4.5, we have presented best partitions of a set of databases at different αs. So, the best partition of a set of databases may change over the change of α.

Table 3.4.4. Best partitions of $\{T_{10}, T_{11}, \ldots, T_{19}\}$

α	Best partition (π)	δ	*goodness* (π)
0.07	$\{\{T_{10},T_{13},T_{14},T_{16},T_{17}\},\{T_{11}\},\{T_{12},T_{15}\},\{T_{18},T_{19}\}\}$	0.724778	85.585823
0.06	$\{\{T_{10},T_{11},T_{15},T_{16},T_{17},T_{18}\},\{T_{12}\},\{T_{13},T_{14},T_{19}\}\}$	0.732767	81.077277
0.05	$\{\{T_{10}\},\{T_{11}\},\{T_{12}\},\{T_{13}\},\{T_{14},T_{16}\},\{T_{15}\},\{T_{17},T_{19}\},\{T_{18}\}\}$	0.889875	13.345596
0.04	$\{\{T_{10}\},\{T_{11},T_{13}\},\{T_{12}\},\{T_{14}\},\{T_{15}\},\{T_{16}\},\{T_{17}\},\{T_{18}\},\{T_{19}\}\}$	0.949766	-2.067990
0.03	$\{\{T_{10}\},\{T_{11}\},\{T_{12}\},\{T_{13}\},\{T_{14},T_{18}\},\{T_{15}\},\{T_{16}\},\{T_{17}\},\{T_{19}\}\}$	0.880798	0.011031

Table 3.4.5. Best partitions of $\{B_{10}, B_{11}, \ldots, B_{19}\}$

α	Best partition (π)	δ	*goodness* (π)
0.020	$\{\{B_{10}\},\{B_{11},B_{12},B_{13},B_{14},B_{15},B_{16},B_{17},B_{18},B_{19}\}\}$	0.667728	51.897608
0.017	$\{\{B_{10}\},\{B_{11},B_{12},B_{13},B_{14},B_{15},B_{16},B_{17},B_{18},B_{19}\}\}$	0.664585	66.100075
0.014	$\{\{B_{10}\},\{B_{11},B_{12},B_{13},B_{14},B_{15},B_{16},B_{17},B_{18},B_{19}\}\}$	0.581388	72.153178
0.010	$\{\{B_{10},B_{11}\},\{B_{12},B_{13},B_{14},B_{15},B_{16},B_{17},B_{18},B_{19}\}\}$	0.559567	63.671384
0.009	$\{\{B_{10},B_{11}\},\{B_{12},B_{13},B_{14}\},\{B_{15}\},\{B_{16},B_{19}\},\{B_{17}\},\{B_{18}\}\}$	0.535796	81.028792

Thus, a partition may not remain the same over the change of α. But, we have observed a general characteristic that the databases show more similarity over a larger value of α. As the value of α becomes smaller, more frequent itemsets are reported from a database, and databases become more dissimilar.

In Figure 3.4.3, we have shown how the execution time of an experiment increases as the number databases increases. The execution time increases faster as we increase input databases from database T_1. The reason is that the size of each local database obtained from T_1 is larger than that of T_4 and B_1.

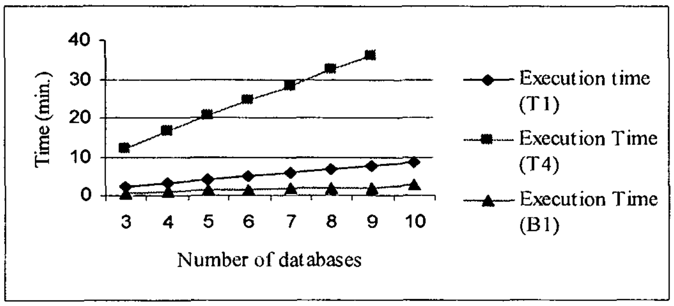


Figure 3.4.3. Execution time versus the number of databases

The number of frequent itemsets decreases as the value of α increases. Thus, the execution time of an experiment decreases as α increases. We observe such phenomenon in Figures 3.4.4 and 3.4.5.

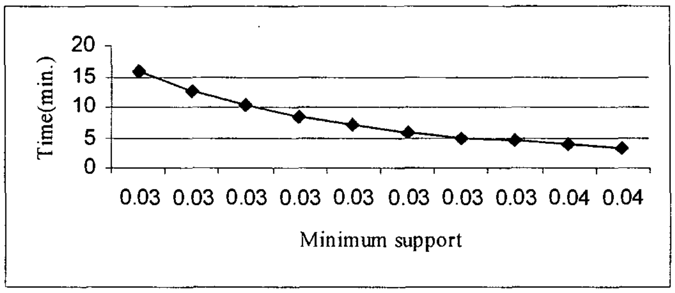


Figure 3.4.4. Execution time versus α for experiment with $\{T_{10}, T_{11}, ..., T_{19}\}$

Chapter 3.5

Study of select items in multiple databases by grouping

The number of multi-branch companies is increasing over the time. Many of them transact from different branches. Therefore, they possess multiple databases. Each branch maintains a database for the transactions made at that branch. Such multi-branch companies collect data continuously through their different branches. The challenges involve in making a quality decision based on large volume of data that are distributed over the branches. It creates not only risks but also opportunities. One of the risks might be a significant amount investment on hardware and software to deal with the large volume of data. Our objective is to provide good quality of knowledge by minimizing the risks.

Many important decisions are based on a set of specific items called the *select items*. In the following, we mention a few decision support applications where the decisions are based on the performances of select items.

- Consider a set of items (products) that are profit making. We could consider them as the select items in this context. Naturally, the company would like to promote them. There are various ways one could promote an item. An indirect way of promoting a select item is to promote items that are positively associated with it. The implication of positive association between a select item P and another item Q is that if Q is purchased by a customer then P is likely to be purchased by the same customer at the same time. Thus, P is indirectly promoted. It is important to identify the items that are positively associated with a select item.

- Each of the select items could be of high standard. Thus, they bring goodwill for the company. They help in promoting other items. Thus, it is important to know how the sales of select items affect the other items. Before making such analyses, we need to identify the items that are positively associated with the select items.
- Again, each of the select items could be a low-profit making product. Thus, it is important to know how they promote the sales of other items. Otherwise, the company could stop dealing with those products.

In general, the performances of select items would affect many decision making problems. Thus, a better analysis of select items leads to a better decision. We study the select items based on the frequent itemsets extracted from multiple databases. The first question comes to our mind whether a traditional data mining technique could provide a good solution in dealing with multiple large databases. The traditional way of mining multiple databases might not provide a good solution due to the following reasons.

- The company might have to employ parallel machines to deal with a large volume of data. It might involve high initial investment on setting up a new infrastructure and recurring investment on hiring technical people.
- A single computer might take unreasonable amount of time to mine a large volume of data. Sometimes, it might not be feasible to carry out the mining task.
- A traditional data mining algorithm might extract a large number of patterns of many irrelevant items. Thus, the processing of patterns could be complex and time consuming.

Therefore, the traditional way of mining multiple databases could not provide a good solution to this problem. In this situation local pattern analysis [91] could be a solution. Under this model of mining multiple databases, each branch requires to mine the local database using a traditional data mining technique. Afterwards, each branch is required to forward the pattern base to the central office. Then, the central office could process the pattern bases collected from different branches for synthesizing the global patterns, or making decisions related to some problems. Due to the following reasons, the local pattern analysis would not be a judicious choice to solve the proposed problem.

Each local pattern base might contain a large number of patterns of many irrelevant items. Thus, the data analysis becomes complicated and time consuming. In this situation, a pattern of a select item might be absent in some local pattern bases. Thus, it might be required to estimate, or ignore some patterns in some databases. Therefore, we may fail to report the true patterns of select items in the union of all local databases.

Thus, the local pattern analysis alone might not provide a good solution for this situation. The above difficulties motivate us to propose a model of mining global patterns of select items from multiple databases. The model has been presented in Section 3.5.3. There are two benefits of the proposed model of synthesizing global patterns of select items. Firstly, the synthesized global patterns are exact in nature. In other words, there is no necessity of estimating a pattern in different databases. Secondly, the company could save funds for dealing with large volume of data. Thus, it minimizes risks involved in decision making process.

Let *I*(*DB*) be the set of items in database *DB*. A common measure of similarity [83], [84] between two objects could be used as a measure of positive association between two items in a database. Thus, we define positive association between two items in a database as follows.

$$PA(x, y, DB) = \frac{\#\,\text{transaction containing both } x \text{ and } y,\, DB}{\#\,\text{transaction containing at least one of } x \text{ and } y,\, DB}, \text{ for } x, y \in I(DB). \tag{3.5.1}$$

where, # *P*, *DB* is the number of transactions in *DB* that satisfy the predicate *P*. *PA* measures only positive association between two items in a database. It does not measure negative association between two items in a database. In the following example, we show that *PA* fails to compute overall association between two items.

Example 3.5.1. Let there ~~are~~ be four branches of a multi-branch company. Let D_i be the database corresponding to the i-th branch of the company, for $i = 1, 2, 3, 4$. The company is interested in analyzing a set of select items (SI) globally. Let SI = $\{a, b\}$. The contents of different databases are given as follows: $D_1 = \{ \{a, e\}, \{b, c, g\}, \{b, e, f\}, \{g, i\} \}$; $D_2 = \{ \{b, c\}, \{f, h\} \}$; $D_3 = \{ \{a, b, c\}, \{a, e\}, \{c, d\}, \{g\} \}$; $D_4 = \{ \{a, e\}, \{b, c, g\} \}$. Initially, we wish to measure the association between two items in a single database, say D_1. Now, $PA(a, b, D_1) = 0$, since there is no transaction in D_1 containing both the items a and b. In these transactions, if one of the items of $\{a, b\}$ is present then the other item of $\{a, b\}$ is not present. Thus, the transactions $\{a, e\}$, $\{b, c, g\}$ and $\{b, e, f\}$ in D_1 imply that the items a and b are negatively associated. Thus, we need to define a measure of negative association between two items in a database. In similar to the measure of positive association, one could define a measure of negative association between two items in a database as follows.

$$NA(x, y, DB) = \frac{\#\,\text{transaction containing exactly one of } x \text{ and } y,\ DB}{\#\,\text{transaction containing at least one of } x \text{ and } y,\ DB}, \text{ for } x, y \in I(DB). \tag{3.5.2}$$

Now, $NA(a, b, D_1) = 1$. We note that $PA(a, b, D_1) < NA(a, b, D_1)$. Overall, we have to say that the items a and b are negatively associated, and the amount of overall association between the items a and b in D_1 is $PA(a, b, D_1) - NA(a, b, D_1) = -1.0$. Thus, the accuracy of association analysis might be low if we consider only the positive association between two items. •

The analysis of relationships among variables is a fundamental task being at the heart of many data mining problems. For example, metrics such as support, confidence, lift, correlation, and collective strength have been used extensively to evaluate the interestingness of association patterns [48], [70], [11], [71], [54]. These metrics are defined in terms of the frequency counts tabulated in a 2×2 contingency table as shown in Table 3.5.1. Tan et al. [75] presents an overview of twenty one interestingness measures proposed in statistics, machine learning and data mining literature. We continue our discussion with the examples cited in [75], and show that none of the proposed measures is effective in finding the overall association by considering both positive and negative associations between two items in a database.

Table 3.5.1. A 2 × 2 contingency table for variables x and y

	y	$\neg y$	Total
x	f_{11}	f_{10}	$f_{1.}$
$\neg x$	f_{01}	f_{00}	$f_{0.}$
Total	$f_{.1}$	$f_{.0}$	$f_{..}$

Table 3.5.2. Examples of contingency tables

Example	f_{11}	f_{10}	f_{01}	f_{00}
E1	8123	83	424	1370
E2	8330	2	622	1046
E3	9481	94	127	298
E4	3954	3080	5	2961
E5	2886	1363	1320	4431
E6	1500	2000	500	6000
E7	4000	2000	1000	3000
E8	4000	2000	2000	2000
E9	1720	7121	5	1154
E10	61	2483	4	7452

From the examples in Table 3.5.2, we notice that the overall association between two items could be negative as well as positive. In fact, a measure of overall association between two items in a database lies in [-1, 1]. We consider the following five out of twenty one interestingness measures, since the association between two items calculated using one of these five measures lies in [-1, 1]. Thus, we shall study their usefulness for the specific requirement of the proposed problem. These five measures are presented in Table 3.5.3.

Table 3.5.3. Relevant interestingness measures for association patterns

Symbol	Measure	Formula
ϕ	ϕ-coefficient	$\dfrac{\mathrm{P}(\{x\}\cup\{y\}) - \mathrm{P}(\{x\})\times\mathrm{P}(\{y\})}{\sqrt{\mathrm{P}(\{x\})\times\mathrm{P}(\{y\})\times(1-\mathrm{P}(\{x\})\times(1-\mathrm{P}(\{y\}))}}$
Q	Yule's Q	$\dfrac{\mathrm{P}(\{x\}\cup\{y\})\times\mathrm{P}(\neg(\{x\}\cap\{y\})) - \mathrm{P}(\{x\}\cup\neg\{y\})\times\mathrm{P}(\neg\{x\}\cup\{y\})}{\mathrm{P}(\{x\}\cup\{y\})\times\mathrm{P}(\neg(\{x\}\cap\{y\})) - \mathrm{P}(\{x\}\cup\neg\{y\})\times\mathrm{P}(\neg\{x\}\cup\{y\})}$
Y	Yule's Y	$\dfrac{\sqrt{\mathrm{P}(\{x\}\cup\{y\})\times\mathrm{P}(\neg(\{x\}\cap\{y\}))} - \sqrt{\mathrm{P}(\{x\}\cup\neg\{y\})\times\mathrm{P}(\neg\{x\}\cup\{y\})}}{\sqrt{\mathrm{P}(\{x\}\cup\{y\})\times\mathrm{P}(\neg(\{x\}\cap\{y\}))} - \sqrt{\mathrm{P}(\{x\}\cup\neg\{y\})\times\mathrm{P}(\neg\{x\}\cup\{y\})}}$
κ	Cohen's κ	$\dfrac{\mathrm{P}(\{x\}\cup\{y\}) + \mathrm{P}(\neg\{x\}\cup\neg\{y\}) - \mathrm{P}(\{x\})\times\mathrm{P}(\{y\}) - \mathrm{P}(\neg\{x\})\times\mathrm{P}(\neg\{y\})}{1 - \mathrm{P}(\{x\})\times\mathrm{P}(\{y\}) - \mathrm{P}(\neg\{x\})\times\mathrm{P}(\neg\{y\})}$
F	Certainty factor	$\max\left(\dfrac{\mathrm{P}(\{y\}\mid\{x\}) - \mathrm{P}(\{y\})}{1-\mathrm{P}(\{y\})}, \dfrac{\mathrm{P}(\{x\}\mid\{y\}) - \mathrm{P}(\{x\})}{1-\mathrm{P}(\{x\})}\right)$

In Table 3.5.4, we rank the contingency tables using each of the above measures.

Table 3.5.4. Ranking of contingency tables using above interestingness measures

Example	*E1*	*E2*	*E3*	*E4*	*E5*	*E6*	*E7*	*E8*	*E9*	*E10*
ϕ	1	2	3	4	5	6	7	8	9	10
Q	3	1	4	2	8	7	9	10	5	6
Y	3	1	4	2	8	7	9	10	5	6
κ	1	2	3	5	4	7	6	8	9	10
F	4	1	6	2	9	7	8	10	3	5

Also, we rank the contingency tables based on the concept of overall association explained in Example 3.5.1. In Table 3.5.5, we present the ranking of contingency tables using overall association.

Table 3.5.5. Ranking of contingency tables using overall association

Example	Overall association	Rank
E1	0.7616	3
E2	0.7706	2
E3	0.9260	1
E4	0.0869	5
E5	0.0203	6
E6	-0.1000	8
E7	0.1000	4
E8	0	7
E9	-0.5406	10
E10	-0.2426	9

None of the five measures ranks contingency tables like the rank given in Table 3.5.5. Thus, none of the above five measures serves the special requirement of the proposed problem. Based on the above discussion, we present the following measure *OA* [1] as an overall association between two items in a database.

Definition 3.5.1. $OA(x, y, DB) = PA(x, y, DB) - NA(x, y, DB)$, for $x, y \in I(DB)$. •

If $OA(x, y, DB) > 0$ then the items x and y are positively associated in *DB*. If $OA(x, y, DB) < 0$ then the items x and y are negatively associated in *DB*. The problem discussed here is not concerned about the association between two items in a group, where none of them is a nucleus item. Thus, it could be considered as a problem of grouping rather than a problem of classification, or clustering.

The rest of the chapter is organized as follows. We state our problem formally in Section 3.5.2. In Section 3.5.3, a model of mining global patterns of select items from multiple databases is proposed. The properties of different measures are discussed in Section 3.5.4. We discuss a method of grouping frequent items in multiple databases in Section 3.5.5. The experimental results are presented in Section 3.5.6. In Section 3.5.7, we discuss work related to this problem.

3.5.2 Problem statement

Consider a multi-branch company that has n branches. Each branch maintains a separate database for the transactions made in that branch. Let D_i be the database corresponding to the i-th branch of the multi-branch company, for $i = 1, 2, \ldots, n$. Also, let D be the union of all branch databases. A large section of a local database might be irrelevant to the current problem. Thus, we divide database D_i into FD_i and RD_i, where FD_i and RD_i are called the *forwarded database* and *remaining database* corresponding to the i-th branch, respectively, for $i = 1, 2, \ldots, n$. We are interested in the forwarded databases, since every transaction in a forwarded database contains at least one select item. The database FD_i is forwarded to the central office for mining global patterns of select items, for $i = 1, 2, \ldots, n$. All the local forwarded databases are amassed into a single database (FD) for the purpose of mining task. We note that the database FD is not large as it contains transactions related to select items. Before stating our problem, we first give some formal definitions.

A set of items is referred to as an *itemset*. An itemset containing k items is called a *k-itemset*. The *support* (supp) [11] of an itemset refers to the fraction of transactions containing the itemset. If an itemset satisfies the user-specified minimum support (α) criterion, then it is called a *frequent itemset* (*FIS*). Similarly, if an item satisfies the user-specified minimum support criterion, then it is called a *frequent item* (*FI*). If a k-itemset is frequent then every item in the k-itemset is also frequent. In this chapter, we study the items in SI. Let $SI = \{s_1, s_2, \ldots, s_m\}$. We wish to construct m groups of frequent items in such a way that the i-th group grows centering round the nucleus item s_i, for $i = 1, 2, ..., m$. Let FD be the union of FD_i, for $i = 1, 2, ..., n$. Also, let $FIS_k(DB \mid \alpha)$ be the set of frequent k-itemsets in database DB at the minimum support level α, for $k = 1, 2$. We state our problem as follows:

Let G_i be the i-the group of frequent items containing the nucleus item $s_i \in SI$, for $i = 1, 2, ..., m$. Construct G_i using $FIS_2(FD \mid \alpha)$ and local patterns in D_i such that $x \in G_i$ implies $OA(s_i, x, D) > 0$, for $i = 1, 2, ..., m$.

Two groups may not be mutually exclusive, as our study involves identifying pairs of items such that the following conditions are true: (i) the items in each pair are positively associated between each other in D, and (ii) one of the items in a pair is a select item. Thus, our study is not concerned with the associativity between a pair of items in a group such that none of them is a select item. The above problem actually results in $m + 1$ groups, where $(m + 1)$-th group G_{m+1} contains the items that are not positively associated with any one of the select items. The proposed study is not concerned with the items in G_{m+1}.

The crux of the proposed problem is to determine the supports of the relevant frequent itemsets in multiple large databases. A technique of estimating support of a frequent itemset in multiple real databases has been proposed by Adhikari and Rao [5]. To estimate the support of an itemset in a database, this technique makes use of the trend of supports of the same itemset in other databases. The trend approach for estimating support of an itemset in a database could be stated as follows: Let the itemset X gets reported from databases $D_1, D_2, ..., D_m$. Also, let $supp(X, \cup_{i=1}^{m} D_i)$ be the support of X in the union of $D_1, D_2, ..., D_m$. Let D_k be a database that does not report X, for $k = m + 1, m + 2, ..., n$. Then the support of X in D_k could be estimated by $\alpha \times supp(X, \cup_{i=1}^{m} D_i)$. Given an itemset X, some local supports of X are estimated and the remaining local supports of X are obtained using a traditional data mining technique. The global support of X is obtained by combining these local supports with the numbers of transactions (i.e., sizes) of the concerned databases. The proposed technique synthesizes relevant frequent itemsets exactly in multiple databases.

We have discussed the limitations of traditional way of mining multiple large databases at the beginning of this chapter. Also, we have observed that local pattern analysis alone could not provide an effective solution to this problem. Thus, we propose a model of mining global patterns of select items from multiple databases. A pattern based on all the local databases is called a *global pattern.* A global pattern containing at least one select item is called a *global pattern of select item.*

3.5.3 A model of mining global patterns of select items from multiple databases

The model of mining global patterns of select items is presented in Figure 3.5.1 [1]. The model could be explained using the following steps:

1. Each branch office constructs the forwarded database and sends it to the central office.
2. Also, each branch extracts patterns from its local database.
3. The central office clubs these forwarded databases into a single database *FD*.
4. A traditional data mining technique is applied to extract patterns from *FD*.
5. The global patterns of select items could be extracted effectively from local patterns and the patterns extracted from *FD*.

The local databases are kept at the bottom level of the proposed model. We need to process these databases as they may not be at the right state for the mining task. Various data preparation techniques [65] like data cleaning, data transformation, data integration, data reduction etc. are applied to data in local databases. We get local processed database PD_i for the i-th branch, for $i = 1, 2, \ldots, n$. The proposed model has a set of interfaces and a set of layers. Each interface is a set of operations that produces database(s) (or, knowledge) based on the lower level database(s). There are five distinct interfaces in the proposed model. The functions of the interfaces are described below.

Interface 2/1 cleans / transforms / integrates / reduces data at the lowest level. By applying these procedures we get processed database from the original database. These operations are performed at the respective branch. At interface 3/2, we apply an algorithm to partition a local database into two parts: forwarded database and remaining database. It is easy to find the forwarded database corresponding to a given database. In the following paragraph, we discuss how to construct FD_i, from D_i, for $i = 1, 2, \ldots, n$.

Initially, FD_i is kept empty. Let T_{ij} be the j-the transaction of D_i, for $j = 1, 2, \ldots, |D_i|$. For D_i, a for-loop on j would run for $|D_i|$ times. At the j-th iteration, the transaction T_{ij} is tested. If T_{ij} contains at least one of the select items then FD_i is updated by $FD_i \cup \{T_{ij}\}$. At the end of the for-loop on j, FD_i gets constructed.

A transaction related to select items might contain items other than the select items. A traditional data mining algorithm could be applied at the interface 5/4 to extract patterns from *FD*. Let *PB* be the pattern base returned by a traditional data mining algorithm. Since, the database *FD* is not large, we could lower further the values of user-defined inputs, like minimum support, minimum confidence, so that *PB* contains more patterns of select items. Therefore, we could get a better analysis of select items. If we wish to study the association between a select item and other frequent items then the exact support values of other items might not be available in *PB*. Then the central office sends a request to each branch office to forward the details (like support values) of some items that would be required to study the select items. Each branch then applies a traditional mining algorithm (at interface 3/2) on its local database and forwards the details of local patterns requested by the central office. Let LPB_i be the details of i-th local pattern base requested by the central office, for $i = 1, 2, \ldots, n$. A global pattern mining application of select items might be required to access the local patterns and the patterns in *PB*. Thus, a global pattern mining application (interface 6/5) could be developed based on the patterns in *PB* and LPB_i, for $i = 1, 2, \ldots, n$. The proposed model of mining global patterns of select items is efficient due to the following reasons:

- One could extract more patterns of select items by lowering further the input parameters like minimum support, minimum confidence, based on the level of data analysis of select items, since *FD* is reasonably small.
- One gets true global patterns of select items as there is no need of estimating them.

Thus, the quality of global patterns is high.

3.5.3.1 An application

We discuss here an application of above model in a real world situation. A multi-branch company wishes to promote two household products *H1* and *H2* with a view to become market leader in the respective category of products. Suppose products *H1* and *H2* are launched before four and six years back, respectively. The company might take a number

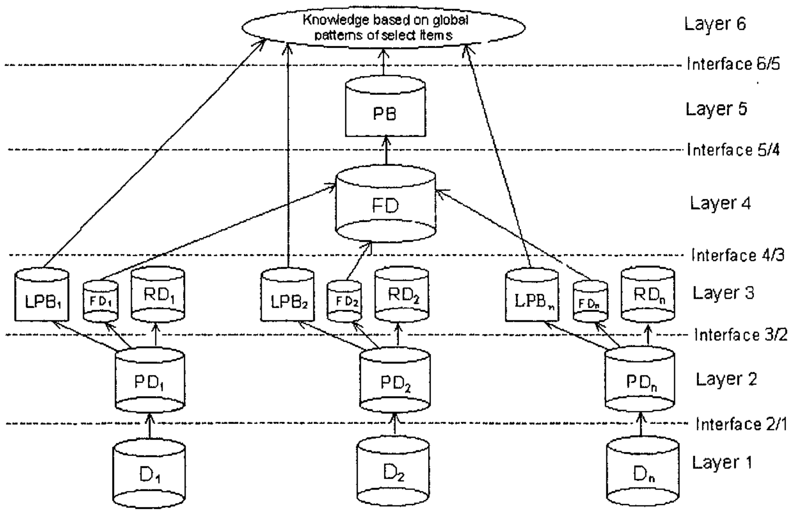


Figure 3.5.1. A model of mining global patterns of select items from multiple databases

of initiatives to increase sales of these two products. Some initiatives are direct in nature. For example, offering some free gifts with these products, advertising in popular news papers, etc. Other initiatives could be based on knowledge available in data across the databases. Using the proposed model, the company could analyze the transactions during last six years in different branches and find the items that are positively associated with *H1* and *H2*. As a part of new initiatives, the company could also promote items that are positively associated with *H1* and *H2*.

3.5.4 Properties of different measures

If the itemset $\{x, y\}$ is frequent in *DB* then $OA(x, y, DB)$ is not necessarily be positive, since the number of transactions containing only one of the items of $\{x, y\}$ could be more than the number of transactions containing both the items x and y. $OA(x, y, DB)$ could attain maximum value for an infrequent itemset $\{x, y\}$ also. Let $\{x, y\}$ be infrequent. The distributions of x and y in *DB* are such that items x and y occur together in some transactions in *DB*. Thus, $OA(x, y, DB) = 1.0$. In the following, we discuss a few properties of different measures.

Lemma 3.5.1. (i) $0 \le PA(x, y, DB) \le 1$; (ii) $0 \le NA(x, y, DB) \le 1$; (iii) $-1 \le OA(x, y, DB) \le 1$; (iv) $PA(x, y, DB) + NA(x, y, DB) = 1$; *for* $x, y \in I(DB)$. •

A characteristic of a good distance measure is that it satisfies metric properties [21] over the concerned domain.

Lemma 3.5.2. *$NA(x, y, DB) = 1 - PA(x, y, DB)$ is a metric over* $[0, 1]$, *for* $x, y \in I(DB)$.

Proof. In the following, we prove only the property of triangular inequality, since the remaining two properties of a metric are trivially true. Let $I(DB) = \{a_1, a_2, \ldots, a_N\}$. Also, let ST_i be the set of transactions containing item $a_i \in I(DB)$, for $i = 1, 2, \ldots, N$.

$$1 - PA(a_p, a_q, DB) = 1 - \frac{|ST_p \cap ST_q|}{|ST_p \cup ST_q|} \ge \frac{|ST_p - ST_q| + |ST_q - ST_p|}{|ST_p \cup ST_q \cup ST_r|} \qquad (3.5.3)$$

Thus, $1 - PA(a_p, a_q, DB) + 1 - PA(a_q, a_r, DB)$

$$\ge \frac{|ST_p - ST_q| + |ST_q - ST_p| + |ST_q - ST_r| + |ST_r - ST_q|}{|ST_p \cup ST_q \cup ST_r|} \qquad (3.5.4)$$

$$= \frac{|ST_p \cup ST_q \cup ST_r| - |ST_p \cap ST_q \cap ST_r| + |ST_p \cap ST_r| + |ST_q| - |ST_p \cap ST_q| - |ST_q \cap ST_r|}{|ST_p \cup ST_q \cup ST_r|}$$

(3.5.5)

$$= 1 - \frac{|ST_p \cap ST_q \cap ST_s| - |ST_p \cap ST_s| - |ST_q| + |ST_p \cap ST_q| + |ST_q \cap ST_s|}{|ST_p \cup ST_q \cup ST_s|} \tag{3.5.6}$$

$$= 1 - \frac{\{|ST_p \cap ST_q \cap ST_s| + |ST_p \cap ST_q| + |ST_q \cap ST_s|\} - \{|ST_p \cap ST_s| + |ST_q|\}}{|ST_p \cup ST_q \cup ST_s|} \tag{3.5.7}$$

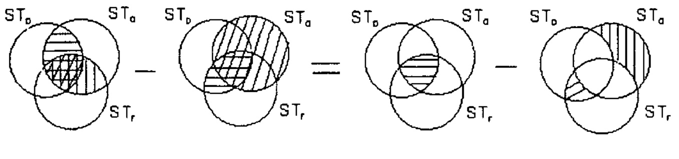


Figure 3.5.2. Simplification using Venn diagram

Let the number of elements in the shaded region of Figures 3.5.2(c) and 3.5.2(d) be N_1 and N_2 respectively. Then, the expression (3.5.7) becomes

$$1 - \frac{N_1 - N_2}{|ST_p \cup ST_q \cup ST_r|} \geq \begin{cases} 1 - \dfrac{N_1 - N_2}{|ST_p \cup ST_q \cup ST_r|}, \text{if } N_1 \geq N_2 & (\text{Case } 1) \\ 1 - \dfrac{|ST_p \cap ST_r|}{|ST_p \cup ST_q \cup ST_r|}, \text{if } N_1 < N_2 & (\text{Case } 2) \end{cases} \tag{3.5.8}$$

In Case 1, the expression remains the same. In Case 2, a positive quantity $ST_p \cap ST_r$ has been put in place of a negative quantity N_1-N_2. The expression (3.5.8) is

$$\geq \begin{cases} 1 - \dfrac{N_1 - N_2}{|ST_p \cup ST_r|}, \text{if } N_1 \geq N_2 \\ 1 - \dfrac{|ST_p \cap ST_r|}{|ST_p \cup ST_r|}, \text{if } N_1 < N_2 \end{cases} \geq \begin{cases} 1 - \dfrac{N_1}{|ST_p \cup ST_r|}, \text{if } N_1 \geq N_2 \\ 1 - \dfrac{|ST_p \cap ST_r|}{|ST_p \cup ST_r|}, \text{if } N_1 < N_2 \end{cases} \geq \begin{cases} 1 - \dfrac{|ST_p \cap ST_r|}{|ST_p \cup ST_r|}, \text{if } N_1 \geq N_2 \\ 1 - \dfrac{|ST_p \cap ST_r|}{|ST_p \cup ST_r|}, \text{if } N_1 < N_2 \end{cases} \tag{3.5.9}$$

where, $N_1 = |ST_p \cap ST_q \cap ST_r| \leq |ST_p \cap ST_r|$. Therefore, irrespective of the relationship between N_1 and N_2, 1- $PA(a_p, a_q, DB)$ + 1- $PA(a_q, a_r, DB) \geq$ 1- $PA(a_p, a_r, DB)$. Thus, 1-$PA(x, y, DB)$ satisfies triangular inequality. •

To compute overall association between two items, we need to express OA in terms of supports of frequent itemsets.

Lemma 3.5.3. *For any two items $x, y \in I(DB)$, $OA(x, y, DB)$ can be expressed as follows.*

$$OA(x, y, DB) = \frac{3 \times supp(\{x, y\}, DB) - supp(\{x\}, DB) - supp(\{y\}, DB)}{supp(\{x\}, DB) + supp(\{y\}, DB) - supp(\{x, y\}, DB)} \qquad (3.5.10)$$

Proof. $OA(x, y, DB) = PA(x, y, DB) - NA(x, y, DB)$

Now, $$PA(x, y, DB) = \frac{supp(\{x, y\}, DB)}{supp(\{x\}, DB) + supp(\{y\}, DB) - supp(\{x, y\}, DB)} \qquad (3.5.11)$$

Also, $$NA(x, y, DB) = \frac{supp(\{x\}, DB) + supp(\{y\}, DB) - 2 \times supp(\{x, y\}, DB)}{supp(\{x\}, DB) + supp(\{y\}, DB) - supp(\{x, y\}, DB)} \qquad (3.5.12)$$

Thus, the lemma follows. •

3.5.5 Grouping of frequent items

For the purpose of explaining the grouping process we continue our discussion of Example 3.5.1.

Example 3.5.2. With reference to Example 3.5.1, the forwarded databases are given as follows: $FD_1 = \{ \{a, e\}, \{b, c, g\}, \{b, e, f\} \}$; $FD_2 = \{ \{ b, c\} \}$; $FD_3 = \{ \{a, b, c\}, \{a, e\} \}$; $FD_4 = \{ \{a, e\}, \{b, c, g\} \}$. Let *size*(*DB*) be the number of transactions in *DB*. Then, $size(D_1) = 4$, $size(D_2) = 2$, $size(D_3) = 4$ and $size(D_4) = 2$. Let *FD* be the union of all forwarded databases. Then, $FD = \{ \{a, e\}, \{b, c, g\}, \{b, e, f\}, \{ b, c\}, \{a, b, c\}, \{a, e\}, \{a, e\}, \{b, c, g\} \}$. The transaction $\{a, e\}$ has been shown thrice, since it originated from three data sources. We minc the database *FD* and get the following set of frequent itemsets: $FIS_1(FD \mid 1/14) = \{ \{a\} (4/12), \{b\} (5/12) \}$, and $FIS_2(FD \mid 1/14) = \{ \{a, b\} (1/12), \{a, c\} (1/12), \{a, e\} (3/12), \{b, c\} (4/12), \{b, e\} (1/12), \{b, f\} (1/12), \{b, g\} (2/12) \}$, where $X(\eta)$ denotes the fact that the frequent itemset X has support η. All the transactions containing item $x \notin$ SI might not be available in *FD*. Thus, other frequent itemsets of size one could not be mined correctly from *FD*. Thus, they are not shown in $FIS_1(FD)$. Each frequent itemset extracted from *FD* contains an item from *SI*. The collection of patterns in $FIS_1(FD \mid 1/14)$ and $FIS_2(FD \mid 1/14)$ could be considered as *PB* with reference to Figure 3.5.1. Using the frequent itemsets in $FIS_1(FD \mid \alpha)$ and $FIS_2(FD \mid \alpha)$ we might not be able to compute the value of *OA* between two items. The central office requests each branch for the supports of the relevant items (*RI*s) to calculate the overall association between two items. Such information would help the central office to compute the value of overall association in the union of all databases exactly. Relevant items are the items in $FIS_2(FD \mid \alpha)$ that do not belong to *SI*. In this example, *RI*s are *c*, *e*, *f* and *g*. The supports of relevant items in different databases are given below.

$RI(D_1)$ = { {*c*} (1/4), {*e*} (2/4), {*f*} (1/4), {*g*} (2/4) }, $RI(D_2)$ = { {*c*} (1/2), {*e*} (0), {*f*} (1/2), {*g*} (0) }, $RI(D_3)$ = { {*c*}(2/4), {*e*}(1/4), {*f*}(0), {*g*}(1/4) }, $RI(D_4)$ = { {*c*} (1/2), {*e*} (1/2), {*f*} (0), {*g*} (1/2) }. •

$RI(D_i)$ could be considered as LPB_i with reference to Figure 3.5.1, for $i = 1, 2, \ldots, n$. We follow here a grouping technique based on the proposed measure of overall association *OA*. If $OA(x, y, D) > 0$ then y could be put in the group of *x*, for $x \in SI = \{a, b\}, y \in I(D)$. We explain the procedure of grouping frequent items with the help of Example 3.5.3.

Example 3.5.3. We continue here the discussion of Example 3.5.2. Based on the available supports of local 1-itemsets, we synthesize 1-itemsets in *D* as mentioned in Table 3.5.6.

Table 3.5.6. Supports of relevant 1-itemsets in *D*

Itemset ({*x*})	{*a*}	{*b*}	{*c*}	{*e*}	{*f*}	{*g*}
supp({*x*}, *D*)	4/12	5/12	5/12	4/12	2/12	4/12

We note that the supports of {*a*} and {*b*} do not required to be synthesized, since they could be determined exactly from mining *FD*. *OA* values corresponding to itemsets of FIS_2 are presented in Table 3.5.7.

Table 3.5.7. Overall association between two items in a frequent 2-itemset in *FD*

Itemset ({*x*, *y*})	{*a*, *b*}	{*a*, *c*}	{*a*, *e*}	{*b*, *c*}	{*b*, *e*}	{*b*, *f*}	{*b*, *g*}
$OA(x, y, D)$	-3/4	-3/4	1/5	1/3	-3/4	-2/3	-3/7

In Table 3.5.7, we find that the items *a* and *e* are positively associated. Thus, item *e* could be put in the group containing nucleus item *a*. Also, items *b* and *c* are positively associated. Thus, item *c* could be put in the group containing nucleus item *b*. Thus, the output grouping π using this technique is given as follows: $\pi(FIS_1(D) \mid \{a, b\}, 1/12) = \{$ Group 1, Group 2 $\}$, where, Group 1 = { (*a*, 1.0), (*e*, 0.2) }, Group 2 = { (*b*, 0.1), (*c*, 0.33333) }. Each item in a group is associated with a real number. This real number represents the amount of overall association between the item and the nucleus item of the group. The proposed grouping technique also constructs the third group of items, i.e., {*f*, *g*}. The proposed study is not concerned with the items in {*f*, *g*}. •

Each group grows centering round a select item. The *i*-th group (G_i) grows centering round the *i*-th select item s_i, for $i = 1, 2, \ldots, m$. With respect to group G_i, the item s_i is called the nucleus item of G_i, for $i = 1, 2, \ldots, m$. We define a group as follows.

Definition 3.5.2. The *i*-th group is a collection of frequent items a_j and the nucleus item $s_i \in SI$ such that $OA(s_i, a_j, D) > 0$, for, $j = 1, 2, \ldots, |G_i|$, and $i = 1, 2, \ldots, m$. •

We describe here the data structures used in the algorithm for finding groups. The set of frequent *k*-itemsets is maintained in an array *FISk*, for $k = 1, 2$. After finding *OA* value between two items in a 2-itemset, it is kept in array *IS2*. Thus, the number of itemsets in *IS2* is equal to the number of frequent 2-itemsets extracted from *FD*. A two-dimensional array Groups is maintained to store *m* groups. The *i*-the row of Groups stores the *i*-th group, for $i = 1, 2, \ldots, m$. The first element of *i*-th row contains the *i*-th select item, for $i = 1, 2, \ldots, m$. In general, the *j*-th element of the *i*-th row contains a pair (*item*, *value*), where *item* refers to the *j*-th item of the *i*-th group and *value* refers to the amount of *OA* value between the *i*-th select item and *item*, for $j = 1, 2, \ldots, |G_i|$. The algorithm [1] for grouping technique is presented as follows.

Algorithm 3.5.1. Construct *m* groups of frequent items in *D* such that *i*-th group grows centering round the *i*-th select item, for $i = 1, 2, \ldots, m$.

```
procudure m-grouping ($m$, $SI$, $N_1$, $FIS1$, $N_2$, $FIS2$, $GSize$, $Groups$)
Input: $m$, $SI$, $N_1$, $FIS1$,$N_2$, $FIS2$
$m$: the number of select items
$SI$: set of select items
$N_k$: number of frequent $k$-itemsets
$FISk$: set of frequent $k$-itemsets
Output: $GSize$, $Groups$
$GSize$: array of number of elements in each group
$Groups$: array of $m$ groups;
01: for $i = 1$ to $N_2$ do
02:   $IS2(i).value = OA(FIS2(i).item1, FIS2(i).item2, D)$;
03:   $IS2(i).item1 = FIS2(i).item1$; $IS2(i).item2 = FIS2(i).item2$;
04: end for
05: for $i = 1$ to $m$ do
06:   $Groups(i)(1).item = SI(i)$; $Groups(i)(1).value = 1.0$; $GSize(i) = 1$;
07: end for
08: for $i = 1$ to $N_2$ do
09:   for $j = 1$ to $m$ do
10:     if (($IS2(i).item1 = SI(j)$) and ($IS2(i).value > 0$)) then
11:       $GSize(j) = GSize(j) + 1$; $Groups(j)(GSize(j)).item = IS2(i).item2$;
12:       $Groups(j)(GSize(j)).value = IS2(i).value$;
13:     end if
14:     if (($IS2(i).item2 = SI(j)$) and ($IS2(i).value > 0$)) then
15:       $GSize(j) = GSize(j) + 1$; $Groups(j)(GSize(j)).item = IS2(i).item1$;
16:       $Groups(j)(GSize(j)).value = IS2(i).value$;
17:     end if
18:   end for
19: end for
```

```
20: for i = 1 to m do
21:     sort items of group i in non-increasing order on OA value;
22: end for
end procedure
```

The algorithm works as follows. Using formula (3.5.10), we compute *OA* value for each itemset in *FIS2*. After computing *OA* value for each itemset, we store the details of the itemset in *IS2*. The algorithm performs these tasks using the for-loop at line 1. We initialize each group with the corresponding nucleus item using lines 05-07. A relevant item or an item in *SI* could belong to one or more groups. Thus, we check for the possibility of including each of relevant items and items in *SI* to each group using the for-loop at line 9. All the relevant items and items in SI are covered using for-loop at line 8. For the purpose of better presentation, we finally sort items of *i*-group in non-increasing order on *OA* value, for $i = 1, 2, \ldots, m$.

Assume that the frequent itemsets in *FIS1* and *FIS2* are sorted on items. Thus, the time complexities for searching an itemset in *FIS1* and *FIS2* are $O(\log(N_1))$ and $O(\log(N_2))$, respectively. The time complexity of line 02 is $O(\log(N_1))$, since $N_1 > N_2$. Thus, the time complexity of lines 01-04 is $O(N_2 \times \log(N_1))$. Lines 05-07 do necessary initialization. The time complexity of this program segment is $O(m)$. Lines 08-19 process frequent 2-itemsets and construct m groups. If one of the two items in a frequent 2-itemset is a select item, then other item could be placed in the group of the select item, provided the overall association between them is positive. The time complexity of this program segment is $O(m \times N_2)$. Lines 20-22 sort m groups. Each group is sorted in non-increasing order on *OA* value. The association of nucleus item with itself is 1.0. Thus, the nucleus item is kept at the beginning of the group (line 06). Let the average size of a group be k. Then, the time complexity of this program segment is $O(m \times k \times \log(k))$. The time complexity of the procedure *m-grouping* is *maximum* $\{ O(N_2 \times \log(N_1)), O(m), O(m \times N_2), O(m \times k \times \log(k)) \}$, i.e., *maximum* $\{ O(N_2 \times \log(N_1)), O(m \times N_2), O(m \times k \times \log(k)) \}$.

3.5.5.1 Error of grouping experiment

Let $\{X_1, X_2, \ldots, X_k\}$ be set of relevant frequent itemsets used to study select items in *D*. One could define the error of experiment in many ways. We define following two types of errors of an experiment:

1. Average Error (AE)

$$\text{AE}(SI, \alpha) = \frac{1}{m} \Sigma_{i=1}^{m} \left| supp_a(X_i, D) - supp_s(X_i, D) \right| \tag{3.5.13}$$

2. Maximum Error (ME)

$$\text{AE}(SI, \alpha) = maximum \left\{ \left| supp_a(X_i, D) - supp_s(X_i, D) \right|, i = 1, 2, \ldots, m \right\} \tag{3.5.14}$$

where, $supp_a(X_i, D)$ and $supp_s(X_i, D)$ are the actual support and synthesized support of itemset X_i in *D*, respectively.

The value of synthesized support of an itemset might differ from one synthesizing method to another synthesizing method. Again, the synthesized support and actual support of itemset X_i in *D* might differ when some databases fail to extract the itemset X_i at a given support. The process of synthesis would affect the synthesized support of an itemset. The model in Figure 3.5.1 enables us to find the actual supports of the relevant itemsets in *D*. Thus, both the AE and ME are 0s, and become independent of the concerned database. Therefore, the technique discussed above is efficient.

3.5.6 Experiments

We have carried out several experiments to study the effectiveness of our approach. All the experiments have been implemented on a 1.6 GHz Pentium IV with 256 MB of memory using visual C++ (version 6.0) software. We present the experimental results using four databases, viz., *retail* [34], *mushroom* [34], *T10I4D100K* [34], and *check*. The database *retail* is real and obtained from an anonymous Belgian retail supermarket store. The database *mushroom* is real and obtained from UCI databases. The database *T10I4D100K* is synthetic and obtained using generator from IBM Almaden Quest research group. The database *check* is artificial whose grouping is already known. We have experimented on database *check* to verify that our grouping technique works correctly. We present some characteristics of these databases in Table 3.5.8. Let *NT*, *AFI*, *ALT*, and *NI* denote the number of transactions, average frequency of an item, average length of a transaction, and number of items in the data source, respectively.

Table 3.5.8. Characteristics of databases

Database	*N T*	*ALT*	*AFI*	*NI*
retail (*R*)	88,162	11.305755	99.673800	10000
mushroom (*M*)	8,124	24.000000	1624.800000	120
T10I4D100K (*T*)	1,00,000	11.10228	1276.12413	870
check(*C*)	40	3.025000	3.102564	39

For the purpose of conducting the experiments, we divide each of these databases into ten databases called *input databases*. The input databases obtained from *R*, *M*, *T* and *C* are names as R_i, M_i, T_i, and C_i, respectively, for i = 1, 2, ..., 10. We present some characteristics of the input databases in Table 3.5.9.

Table 3.5.9. Characteristics of input databases obtained from *retail* and *mushroom*

DB	*NT*	*ALT*	*AFI*	*NI*	*DB*	*NT*	*ALT*	*AFI*	*NI*
R_1	9000	11.24389	12.07001	8384	M_1	*812*	*24.00000*	*295.27272*	*66*
R_2	9000	11.20922	12.26541	8225	M_2	*812*	*24.00000*	*286.58823*	*68*
R_3	9000	11.33667	14.59657	6990	M_3	*812*	*24.00000*	*249.84615*	*78*
R_4	*9000*	*11.48978*	*16.66259*	*6206*	M_4	812	24.00000	282.43478	69
R_5	*9000*	*10.95678*	*16.03953*	*6148*	M_5	812	24.00000	259.84000	75
R_6	9000	10.85578	16.70977	5847	M_6	812	24.00000	221.45454	88
R_7	9000	11.20011	17.41552	5788	M_7	812	24.00000	216.53333	90
R_8	9000	11.15511	17.34554	5788	M_8	812	24.00000	191.05882	102
R_9	9000	11.99711	18.69032	5777	M_9	812	24.00000	229.27058	85
R_{10}	9000	11.69199	*15.34787*	5456	M_{10}	816	24.00000	227.72093	86

Table 3.5.9(continued). Characteristics of input databases obtained from *T10I4D100K*

DB	*ALT*	*AFI*	*NI*	*DB*	*ALT*	*AFI*	*NI*
T_1	11.05500	127.65588	866	T_6	11.13910	128.62702	866
T_2	11.13330	128.41176	867	T_7	11.10780	128.56250	864
T_3	11.06700	127.64705	867	T_8	11.09840	128.45376	864
T_4	11.12260	128.43649	866	T_9	11.08150	128.55568	862
T_5	11.13670	128.74797	865	T_{10}	11.08140	128.10867	865

The input databases obtained from database *check* are given as follows: C_1 = { {1, 4, 9, 31}, {2, 3, 44, 50}, {6, 15, 19}, {30, 32, 42} }; C_2 = { {1, 4, 7, 10, 50}, {3, 44}, {11, 21, 49}, {41, 45, 59} }; C_3 = { {1, 4, 10, 20, 24}, {5 ,7, 21}, {21, 24, 39}, {26, 41, 46} }; C_4 = { {1, 4, 10, 23}, {5, 8}, {5, 11, 21}, {42, 47} }; C_5 = { {1, 4, 10, 34}, { 5, 49}, {25, 39, 49}, {49} }; C_6 = { {1, 3, 44}, {6, 41}, {22, 26, 38}, {45, 49} }; C_7 = { {1, 2, 3, 10, 20, 44}, {11, 12, 13}, {24, 35}, {47, 48, 49} }; C_8 = { {2, 3, 20, 39}, {2, 3, 20, 44, 50}, {32, 49}, {42, 45} }; C_9 = { {2, 3, 20, 44}, {3, 19, 50}, {5, 41, 45}, {21} }; C_{10} = { {2, 20, 45}, {5, 7, 21}, {11, 19}, {22, 30, 31} }.

In Table 3.5.10, we present some relevant information regarding different experiments. We have chosen the first 10 frequent items as the select items, except for the last experiment. One could choose select items as the items whose data analyses are of urgent need.

Table 3.5.10. Some relevant information regarding experiments

Database	α	*SI*
R	0.03	{0,1,2,3,4,5,6,7,8, 9}
M	0.05	{1, 3, 9, 13, 23, 34, 36, 38, 40, 52}
T	0.01	{2, 25, 52, 240, 274, 368, 448, 538, 561, 630}
C	0.07	{1, 2, 3}

The first experiment is based on database *retail*. The grouping of frequent items in *retail* is given below:

π (*FI*(*retail*) | *SI*, α) = { $\underline{0}$ (1.000000); $\underline{1}$ (1.000000); $\underline{2}$ (1.000000); $\underline{3}$ (1.000000); $\underline{4}$ (1.000000); $\underline{5}$ (1.000000); $\underline{6}$ (1.000000); $\underline{7}$ (1.000000); $\underline{8}$ (1.000000); $\underline{9}$ (1.000000) }.

Two groups are separated by semicolon (;). The nucleus item in each group is underlined. Each item in a group is associated with a real number and shown in bracket. This value represents the amount of overall association between the item and the nucleus item. The groups are shaded alternately for the purpose of clarity of visualization. We observe that no item in database *retail* is positively associated with the select items using the measure *OA*.

The second experiment is based on database *mushroom*. The grouping of frequent items in *mushroom* is given below:

π (*FI*(*mushroom*) | *SI*, α) = { 1 (1.000000), 24 (0.225614), 110 (0.115114), 29 (0.103166), 36 (0.098787), 61 (0.097537), 38 (0.058040), 66 (0.064039), 90 (0.002633); 3 (1.000000); 9 (1.000000); 13 (1.000000); 23 (1.000000), 93 (0.530769), 59 (0.215038), 2 (0.139073), 39 (0.010047), 63 (0.153263); 34 (1.000000), 86 (0.993444), 85 (0.948301), 90 (0.800148), 36 (0.625308), 39 (0.329887), 59 (0.229706), 63 (0.170491), 53 (0.164519), 67 (0.128127), 24 (0.117184), 76 (0.107910); 36 (1.000000), 85 (0.677006), 90 (0.649821), 86 (0.631345), 34 (0.625308), 59 (0.169751), 39 (0.158109), 63 (0.106293), 110 (0.098801), 1 (0.098787); 38 (1.000000), 48 (0.375796), 102 (0.185956), 58 (0.135678), 1 (0.058040), 94 (0.046083), 110 (0.011029); 40 (1.000000); 52 (1.000000) }.

We observe that some frequent items are not included in any of these groups, since their overall associations with each of the select items are non-positive.

The third experiment is based on database *T10I4D100K*. The grouping of frequent items in *T10I4D100K* is given below:

π (*FI*(*T10I4D100K*) | *SI*, α) = { 2 (1.000000); 25 (1.000000); 52 (1.000000); 240 (1.000000); 274 (1.000000); 368 (1.000000); 448 (1.000000); 538 (1.000000); 561 (1.000000); 630 (1.000000) }.

We observe that databases *retail* and *T10I4D100K* are sparse. Thus, the grouping contains groups of singleton item for these two databases. The overall association between a nucleus item and itself is 1.0. Otherwise, the overall association between a frequent item and a nucleus item is non-positive for two databases.

The fourth experiment is based on database *check*. The database *check* is constructed artificially to verify the following existing grouping.

π (*FI*(*check*) | *SI*, α) = { (1, 1.000000), (4, 0.428571), (10, 0.428571); (2, 1.000000), (20, 0.428571), (3, 0.111111); (3, 1.000000), (44, 0.500000), (2, 0.111111) }.

We have calculated average errors using both trend and proposed approaches. Figures 3.5.3, 3.5.4 and 3.5.5 show the graphs of AE versus the number databases for the first three databases. The proposed model enables us to find actual supports of all the relevant itemsets in a database. Thus, the AE of an experiment for the proposed approach remains 0. As the number of databases increases, the relative presence of a frequent itemset normally decreases. Thus, the error of synthesizing an itemset also increases. So, the AE of an experiment using trend approach is likely to increase as the number of databases increases. We observe such phenomenon in Figures 3.5.3, 3.5.4 and 3.5.5.

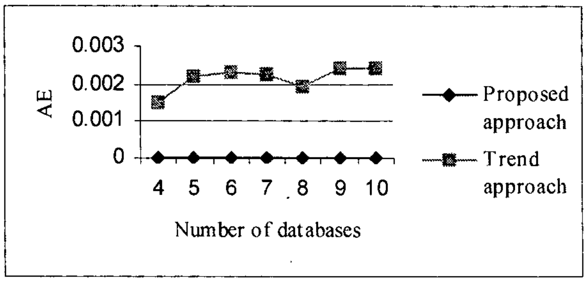


Figure 3.5.3. Average error versus the number of the databases from *retail*

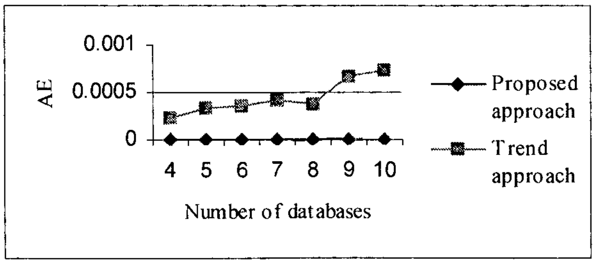


Figure 3.5.4. Average error versus the number of databases from *mushroom*

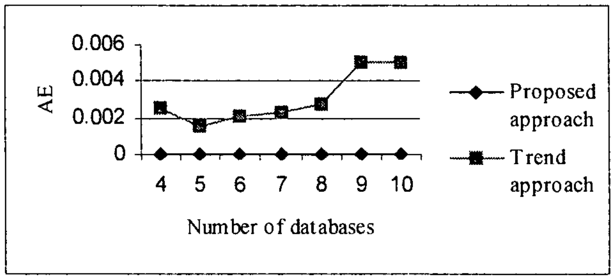


Figure 3.5.5. Average error versus the number of databases from *T10I4D100K*

3.5.7 Related work

Multi-database mining has been recognized recently as a research area in KDD community. The work reported so far could be classified broadly into following two categories: mining / synthesizing patterns in multiple databases and postprocessing of local patterns. We mention some work related to first category. Wu and Zhang [81] have proposed a weighting method for synthesizing high-frequency rules in multiple databases. Zhang et al. [89] have proposed an algorithm to identify global exceptional patterns in multiple databases. Now, we mention some work related to second category. Wu et al. [83] have proposed a technique for clustering multiple databases for multi-database mining. Adhikari and Rao [6] have proposed an efficient technique for clustering multiple databases using local patterns.

In the context of estimating support of itemsets in a database, Jaroszewicz and Simovici [45] have proposed a method using Bonferroni-type inequalities [35]. Also, the maximum-entropy approach to support estimation of a general Boolean expression is proposed by Pavlov et al. [62]. But, these support estimation techniques are suitable for a single database.

Zhang et al. [93], Zhang [88] have studied various strategies for mining multiple databases. Proefschrift [64] has studied data mining on multiple relational databases.

Existing parallel mining techniques [12], [26], [28] could also be used to deal with multi-databases. These techniques provide expensive solutions for studying select items in multiple databases.

3.5.8 Conclusion

The measure of overall association *OA* is effective as it considers both positive and negative association between two items. Association analysis of select items in multiple market basket databases is important as well as promising issue, since many data analyses of a multi-branch company are based on select items. The model of mining global patterns of select items in multiple databases is efficient, since it is not required to estimate the patterns in multiple databases.

Chapter 3.6

A framework for developing better multi-database mining applications

In the recent years, a number of multi-database mining applications [1], [6], [50], [81], [83], [85] have been reported. There are many strategies by which one could develop a multi-database mining application. All the solutions for a particular application might not produce good result. The goal of this chapter is to provide a framework to develop better multi-database mining applications systematically. A multi-database mining application could be developed using a number of stages. It might be possible to provide a frame work for each stage of the development process. Before we discuss them, first we discuss some existing approaches for mining multiple large databases.

3.6.2 Existing approaches of multi-database mining

In a distributed data mining environment, one may encounter different types of data. For example, stream data, geographical data, image data, transactional data are quite common. In this thesis, we deal with multiple transactional databases. One of the main hurdles for developing better multi-database mining applications is to mine multiple databases with high accuracy. In the following sections, we discuss three approaches to mining multiple large databases.

3.6.2.1 Local pattern analysis

A convenient way to mine global patterns is to mine each local database, and then analyze all the local patterns to synthesize global patterns. This technique is called local pattern analysis. Zhang et al. [91] designed local pattern analysis for the purpose of addressing various problems related to multiple large databases. Let there are n branches of a multi-branch company. Also, let D_i be the database corresponding to the i-th branch, for $i = 1, 2, ..., n$. Mining multiple databases using local pattern analysis could be explained using Figure 3.6.1.

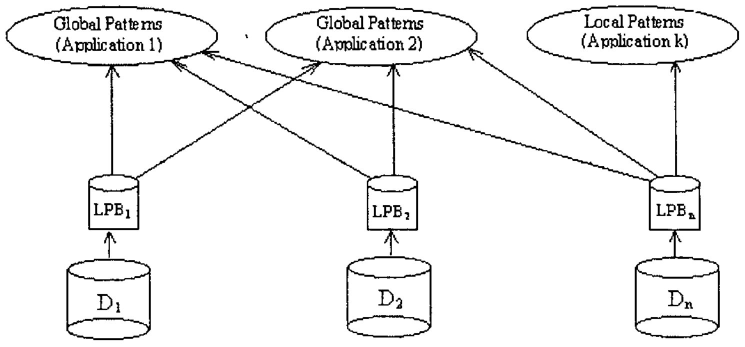

Global Patterns (Application 1)
Global Patterns (Application 2)
Local Patterns (Application k)
LPB1
LPB2
LPBn
D1
D2
Dn

Figure 3.6.1. Mining patterns in multiple databases using local pattern analysis

Let LPB_i be the local pattern base corresponding to D_i, for $i = 1, 2, ..., n$. In mult-database environment, local patterns could be used in three ways: (i) Analyzing local data, (ii) Synthesizing non-local patterns, and (iii) Measuring relevant statistics for decision making problems. Multi-database mining using local pattern analysis could be considered as an approximate method of mining multiple large databases. Thus, it might be required to enhance the quality of knowledge synthesized from multiple databases.

3.6.2.2 Sampling

In multi-database environment, the collection of all branch databases might be very large. Effective data analysis using a traditional data mining technique on multi-gigabyte repositories has proven difficult. A quick approximate knowledge from a large database would be adequate for many decision support applications. In these cases, one could tame multiple large databases by sampling. A commonly used technique for approximate pattern answering is sampling [76]. If an itemset is frequent in a large database then it is likely that the itemset is frequent in a sample database. Thus, one could analyze approximately the database by analyzing the frequent itemsets in a representative sample data. A combination of sampling and local pattern analysis could be a useful approach for mining multiple databases for addressing many decision support applications. Multi-database mining using sampling might not be satisfactory, since we may need good precision of output patterns in many occasions.

3.6.2.3 Re-mining

For the purpose of mining multiple databases, one could apply partition algorithm proposed by Savasere et al. [66]. The algorithm is designed for mining a very large database by partitioning. The algorithm works as follows. It scans the database twice. The database is divided into disjoint partitions, where each partition is small enough to fit in memory. In a first scan, the algorithm reads each partition and computes locally frequent itemsets in each partition using apriori algorithm [13]. In the second scan, the algorithm counts the supports of all locally frequent itemsets toward the complete database. In this case, each local database could be considered as a partition. Though partition algorithm mines frequent itemsets in a database exactly, it might be an expensive solution to mining multiple large databases, since each database is required to scan twice. During the time of second scanning, all the local patterns obtained at the first scan are analyzed. Thus, the partition algorithm used for mining multiple databases could be considered as another type of local pattern analysis. Multi-database mining using re-mining strategy could be expensive in many cases, since thc sizcs of the available databases could be large.

3.6.3 Improving multi-database mining applications

One could mine multiple databases using one the following approaches: (i) a traditional data mining approach, (ii) a non-traditional data mining approach. Some examples of traditional data mining approaches are apriori algorithm [13], FP-growth algorithm [39], and P-tree algorithm [29]. For applying a traditional data mining approach, one needs to amass all the databases together. Thus, the collection of branch databases could be considered as a single source of data. In this case, the extracted patterns are exact in nature. Thus, no improvement of patterns (output) is required. But, it might be possible to improve different traditional data mining algorithms with respect to time complexity, space complexity, and other parameters of different mining algorithms. Though these are interesting topics, but we do not discuss these issues here. Some examples of non-traditional data mining approaches that could be used for mining multiple databases are partition algorithm [66], local pattern analysis [91], and sampling technique [76]. In Section 3.6.2, we have observed drawbacks of each of these non-traditional data mining approaches. We propose various strategies for improving multi-database mining applications. Some improvements are general in nature, while others are specific. The efficiency of a multi-database application could be enhanced by choosing an appropriate multi-database mining model, an appropriate pattern synthesizing technique, a better pattern representation technique, and a better algorithm for solving the problem. In this thesis, we have illustrated each of these issues either in the context of a specific problem, or in general. We do not discuss an efficient implementation of different algorithms, since the topic has been well studied.

3.6.3.1 Preparation of data warehouses

It might be possible that all the data sources are not of the same format. One needs to process them before the mining task resumes. Relevant data are required to be retained. Also, the definitions of data are required to be the same at every data source. Thus, the preparation of data warehouse could be a significant task for handling multiple large databases. Adhikari and Rao [5] have proposed an extended model of synthesizing global patterns from local patterns in different databases. In Chapter 2.3, we have discussed how this model could be used for mining heavy association rules in multiple databases. Also, it shows how the task of data preparation could be broken into sub-tasks so that the data preparation task becomes easy and systematic. Though the above model introduces many layers and interfaces for synthesizing global patterns, but in a real life application, many of these layers and interfaces might not be useful.

3.6.3.2 Selection of databases

Selection of databases could be based on the inherent knowledge in the databases. Thus, one needs to mine each of the local databases. One could then process the local patterns in different databases for the purpose of selecting relevant databases. Local patterns help selecting relevant databases. In other words, the clustering of databases is based on a measure of similarity between two databases. Thus, the measure of similarity between two databases is an important issue. It could be based on local patterns in the databases.

Wu et al. [83] have proposed a similarity measure sim_1 to identify similar databases based on item similarity. The authors have designed a clustering algorithm based on measure sim_1 to cluster databases for the purpose of selecting relevant databases. Such clustering is useful when the similarity is based on items in different databases. Such similarity measure might not be useful for many multi-database mining applications where clustering of databases might be based on other criteria. For example, if we are interested in the databases based on transaction similarity then the above measures might not be appropriate. Adhikari and Rao [6] have proposed database clustering based on transaction similarity of different databases. The authors have proposed a similarity measure $simi_1$ to cluster databases. Also, the authors have designed a clustering algorithm based on $simi_1$ for the purpose of selecting relevant databases. In Chapter 3.4, we have discussed all these issues.

A quick approximate knowledge from large databases would be adequate for many decision support applications. Thus, the selection of databases might be important in many decision support applications. It reduces the cost of searching necessary information.

3.6.3.3 Choosing appropriate technique of multi-database mining

A particular technique of mining multiple databases might not be appropriate in all the situations. Adhikari and Rao [8] have proposed a multi-database mining technique, MDMT: *PipelinedFeedbackModel + simple pattern synthesizing*, for mining multiple large databases. It improves multi-database mining significantly as compared to an existing technique. We have shown in Chapter 3.2, the effectiveness of MDMT: *PipelinedFeedbackModel + simple pattern synthesizing*, by conducting relevant experiments.

But, MDMT: *PipelinedFeedbackModel + simple pattern synthesizing* might not be the best at all the situations. For example, Adhikari and Rao [1] have proposed a technique for mining multiple large databases to study problems involving a set of specific items in multiple databases. This technique performs better than MDMT: *PipelinedFeedbackModel + simple pattern synthesizing*. It extracts patterns related to a set of specific items from multiple databases exactly. In Chapter 3.5, we have discussed a multi-database mining technique for studying a set of specific items in multiple databases.

Thus the choice of a multi-database mining technique is an important issue. One could obtain better solution by choosing an appropriate multi-database mining technique.

3.6.3.4 Representing patterns space efficiently

Multi-database mining using local pattern analysis could be considered as an approximate method of mining multiple large databases. Thus, it might be required to enhance the quality of knowledge synthesized from multiple databases. Also, many decision-making applications are directly based on the available local patterns in different databases. The quality of synthesized knowledge / decision based on local patterns in different databases could be enhanced by incorporating more local patterns in the knowledge synthesizing / processing activities. Thus, the available local patterns play a crucial role in building efficient multi-database mining applications. One could represent patterns in condensed form by employing a suitable coding. It allows us to consider more local patterns by lowering further the user inputs, like minimum support and minimum confidence. The coding enables more local patterns participate in the knowledge synthesizing / processing activities and thus, the quality of synthesized knowledge based on local patterns in different databases gets enhanced significantly at a given pattern synthesizing algorithm and computing resource.

3.6.3.4.1 *Representing association rules*

Adhikari and Rao [4] have proposed ACP coding to represent association rules in multiple databases. In Chapter 3.3, we have discussed ACP coding and presented experimental results to show the effectiveness of ACP coding for representing association rules in multiple databases.

3.6.3.4.2 *Representing frequent itemsets*

Adhikari and Rao [6] have proposed IS coding to represent frequent itemsets in multiple databases. In Chapter 3.4, we have discussed theoretically the effectiveness of IS coding.

3.6.3.5 Designing a better algorithm for solving the problem

Using suitable data structures and an algorithm [14] one could develop a better multi-database mining application. In the context of developing better multi-database mining applications, Adhikari and Rao [5] have proposed a better algorithm for extracting high-frequent association rules in multiple databases. In Chapter 2.3, we have presented experimental results to show effectiveness of the algorithm. Also, Adhikari and Rao [6] have designed a better algorithm for clustering the databases. In Chapter 3.5, we have discussed different issues including designing an efficient algorithm for clustering databases. The execution time of the algorithm has been improved using Theorem 3.4.5.

These are a few instances of how a multi-database mining application could be developed in a better way.

3.6.4 Conclusions

Multi-database mining applications might come with different types and complexities. It might be difficult to give a generalized framework for developing better multi-database mining applications. Nevertheless, it might be possible to identify some important stages of the development process. Thus, our framework has attempted to provide a better solution to each stage of the development process. We believe that such framework might help practitioners of data mining to develop systematically better multi-database mining technologies.

Chapter 3.7

Conclusion

In the context of mining multiple large databases, we have presented MDMT: PFM+SPS for mining multiple databases. It improves significantly the accuracy of mining multiple databases as compared to existing techniques that scan each database only once.

An efficient storage representation of a set of pattern bases builds the foundation of a multi-database mining system. In this regard, we have proposed a space efficient represent of association rules, called ACP coding. Many global applications could be developed effectively upon this foundation.

Clustering a set of databases is an important activity. It reduces cost of searching relevant information for many problems. We provide an efficient solution to this problem in three ways. Firstly, we propose more appropriate measures of similarity between two databases. Secondly, we need to find the existence of the best clustering only at few similarity levels. Thus, the proposed clustering algorithm executes faster. Lastly, we introduce IS coding for storing frequent itemsets in main memory space efficiently. It allows more frequent itemsets to participate in the clustering process. IS coding enhances the accuracy of the clustering process.

We have proposed a measure, called *OA*, for measuring overall association between two items in a database. The proposed measure of association *OA* is effective as it considers both positive and negative association between two items. Association analysis of select items in multiple market basket databases is important as well as promising issue, since many data analyses of a multi-branch company are based on select items.

We have worked on theory and application of multi-database mining. Also, we have made significant contribution towards designing an efficient multi-database mining system. We believe that our efforts would help building efficient multi-database mining technologies as explained in Chapter 3.6.

References

[1] A. Adhikari, P. R. Rao, "Study of select Items in multiple databases by grouping", *Proceedings of 3rd Indian International Conference on Artificial Intelligence*, 2007, pp. 1699 - 1718.

[2] A. Adhikari, P. R. Rao, "Synthesizing global exceptional patterns in multiple databases", *Proceedings of 3rd Indian International Conference on Artificial Intelligence*, 2007, pp. 512 - 531.

[3] A. Adhikari, P. R. Rao, "A framework for synthesizing arbitrary Boolean expressions induced by frequent itemsets", *Proceedings of 3rd Indian International Conference on Artificial Intelligence*, 2007, pp. 5 - 23.

[4] A. Adhikari, P. R. Rao, "Enhancing quality of knowledge synthesized from multi-database mining", *Pattern Recognition Letters* 28(16), 2007, pp. 2312 - 2324.

[5] A. Adhikari, P. R. Rao, "Synthesizing heavy association rules from different real data sources", *Pattern Recognition Letters* 29(1), 2008, pp. 59-71.

[6] A. Adhikari, P. R. Rao, "Efficient clustering of databases induced by local patterns", *Decision Support Systems* 44(4), 2008, pp. 925 - 943.

[7] A. Adhikari, P. R. Rao, "Association Rules Induced by Item and Quantity Purchased", J. R. Haritsa, R. Kotagiri, and V. Pudi (Eds.): *Proceedings of International Conference on Database Systems for Advanced Applications*, LNCS 4947, pp. 478 - 485, 2008.

[8] A. Adhikari, P. R. Rao, J. Adhikari, "Mining Multiple Large Databases", *Proceedings of 10th International Conference on Information Technology*, 2007, pp. 80 - 84.

[9] A. Adhikari, P. R. Rao, "Mining Conditional Patterns in a Database", *Pattern Recognition Letters* 29(10), 2008, pp. 1515-1523.

[10] C. Aggarwal, P. Yu, "A new framework for itemset generation", *Proceedings of the 17th Symposium on Principles of Database Systems*, 1998, pp. 18-24.

[11] R. Agrawal, T. Imielinski, A. Swami, "Mining association rules between sets of items in large databases", *Proceedings of ACM SIGMOD Conference*, 1993, pp. 207 - 216.

[12] R. Agrawal, J. Shafer, "Parallel mining of association rules", *IEEE Transactions on Knowledge and Data Engineering* 8(6), 1999, pp. 962 - 969.

[13] R. Agrawal, R. Srikant, "Fast algorithms for mining association rules", *Proceedings of International Conference on Very Large Data Bases*, 1994, pp. 487 - 499.

[14] A. V. Aho, J. E. Hopcroft, J. D. Ullman, *Data structures and algorithm*, Addison-Wesley, 1987.

[15] K. Ali, S. Manganaris, R. Srikant, "Partial classification using association rules", *Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining*, 1997, pp. 115-118.

[16] V.S. Ananthanarayana, M. N. Murty, D. K. Subramanian, "Tree structure for efficient data mining using rough sets", *Pattern Recognition Letters* 24(6), 2003, pp. 851 - 862.

[17] M.-L. Antonie, O.R. Zaïane, "Mining positive and negative association rules: An approach for confined rules", *Proceedings of PKDD*, 2004, pp. 27 - 38.

[18] J. Aronis, V. Kolluri, F. Provost, B. Buchanan, "The WoRLD: Knowledge discovery from multiple distributed databases", *Proceedings of the Tenth International Florida AI Research Symposium*, 1997, pp. 337 - 341.

[19] B. Babcock, S. Chaudhury, G. Das, "Dynamic sample selection for approximate query processing", *Proceedings of ACM SIGMOD Conference Management of Data*, 2003, pp. 539 - 550.

[20] V. Barnett, T. Lewis, *Outliers in statistical data*, 3rd edition, Vol. 2, Wiley, 1995.

[21] R. G. Barte, *The elements of real analysis*, 2nd edition, John Wiley & Sons, 1976.

[22] S. Brin, R. Motwani, J. D. Ullman, S. Tsur, "Dynamic itemset counting and implication rules for market basket data", *Proceedings of ACM SIGMOD Conference*, 1997, pp. 255 - 264.

[23] M. Burrows, D. J. Wheeler, "A block-sorting lossless data compression algorithm", *DEC, Digital Systems Research Center*, Research Report 124, 1994.

[24] A. Bykowski, C. Rigotti, "A condensed representation to find frequent patterns for efficient mining", *Information Systems* 28(8), 2003, pp. 949 - 977.

[25] S. W. K. Chan, M. W. C. Chong, "Unsupervised clustering for nontextual web document classification", *Decision Support Systems* 37(3), 2004, pp. 377 - 396.

[26] J. Chattratichat, J. Darlington, M. Ghanem, Y. Guo, H. Hüning, M. Köhler, J. Sutiwaraphun, H. W. To, D. Yang, "Large scale data mining: Challenges, and responses", *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining*, 1997, pp. 143 - 146.

[27] Y. -L. Chen, K. Tang, R. -J. Shen, Y. -H. Hu, "Market basket analysis in a multiple store environment", *Decision Support Systems* 40(2), 2005, pp. 339 - 354.

[28] D. Cheung, V. Ng, A. Fu, Y. Fu, "Efficient mining of association rules in distributed databases", *IEEE Transactions on Knowledge and Data Engineering* 8(6), 1996, pp. 911 - 922.

[29] F. Coenen, P. Leng, S. Ahmed, "Data structure for association rule mining: T-trees and P-trees", *IEEE Transactions on Knowledge and Data Engineering* 16(6), 2004, pp. 774 - 778.

[30] V. Estivill-Castro, J. Yang, "Fast and robust general purpose clustering algorithms", *Data Mining and Knowledge Discovery* 8(2), 2004, 127 - 150.

[31] W. Feller, *An introduction to probability theory and its applications*, 3rd edition, Vol 1, Wiley, 1968.

[32] FIMI 2004, http://fimi.cs.helsinki.fi/src/

[33] J. B. Fraleigh, *A first course in abstract algebra*, third edition, Addision-Wesley, 1982.

[34] Frequent itemset mining dataset repository, http://fimi.cs.helsinki.fi/data.

[35] J. Galambos, I. Simonelli, *Bonferroni-type inequalities with applications*, Springer, 1996.

[36] M. N. M. García, L. A. M. Quintales, F. J. G. Peñalvo, M. J. P. Martín, "Building knowledge discovery-driven models for decision support in project management", *Decision Support Systems* 38(2), 2004, pp. 305 - 317.

[37] J. R. Gregg, *Ones and zeros: Understanding Boolean algebra, digital circuits, and the logic of sets*, Wiley-IEEE Press, 1998.

[38] J. Han, M. Kamber, *Data Mining: Concepts and techniques*, Morgan Kauffmann Publishers, 2001.

[39] J. Han, J. Pei, Y.Yiwen, "Mining frequent patterns without candidate generation", *Proceedings of ACM SIGMOD Conference on Management of Data*, 2000, pp.1 - 12.

[40] S. L. Hershberger, D. G. Fisher, *Measures of association, encyclopedia of statistics in behavioral science*, John Wiley & Sons, 2005.

[41] R. J. Hilderman, H. J. Hamilton, "Knowledge discovery and interestingness measures: A survey", Technical Report CS-99-04, *Department of Computer Science, University of Regina*, 1999.

[42] S. J. Hong, S. M. Weiss, "Advances in predictive models for data mining", *Pattern Recognition Letters* 22(1), 2001, pp. 55 - 61.

[43] D. A. Huffman, "A method for the construction of minimum redundancy codes", *Proceedings of the IRE* 40(9), 1952, pp. 1098 - 1101.

[44] A. K. Jain, M. N. Murty, P. J. Flynn, "Data clustering: A review", *ACM Computing Surveys*, 31(3), 1999, pp. 264 - 323.

[45] S. Jaroszewicz, D. A. Simovici, "Support approximations using Bonferroni-type inequalities", *Proceedings of Sixth European Conference on Principles of Data Mining and Knowledge Discovery*, 2002, pp. 212 - 223.

[46] B. Jeudy, J. F. Boulicaut, "Using condensed representations for interactive association rule mining", *Proceedings of PKDD*, LNAI 2431, 2002, pp. 225 - 236.

[47] KDD CUP 2000, http://www.ecn.purdue.edu/KDDCUP.

[48] M. Klemettinen, H. Mannila, P. Ronkainen, T. Toivonen, A. Verkamo, "Finding interesting rules from large sets of discovered association rules", *Proceedings of the 3rd International Conference on Information and Knowledge Management*, 1994, pp. 401 - 407.

[49] D. E. Knuth, *The art of computer programming*, Vol. 3, Addision-Wesley, 1973.

[50] H. -C. Kum, H. J. Chang, W. Wang, "Sequential pattern mining in multi-databases via multiple alignment", *Data Mining and Knowledge Discovery* 12(2-3), pp. 151 - 180, 2006.

[51] M. Last, A. Kandel, "Automated detection of outliers in real-world data", *Proceedings of the Second International Conference on Intelligent Technologies*, 2001, pp. 292 - 301.

[52] C. -H. Lee, C. -R. Lin, M. -S. Chen, "Sliding-window filtering: An efficient algorithm for incremental mining", *Proceedings of 10th International Conference on Information and Knowledge Management*, 2001, pp. 263 - 270.

[53] C. L. Liu, *Elements of discrete mathematics*, 2nd edition, McGraw-Hill, 1985.

[54] B. Liu, W. Hsu, Y. Ma, "Pruning and summarizing the discovered associations", *Proceedings of the 5th International Conference on Knowledge Discovery and Data Mining*, 1999, pp. 125 - 134.

[55] H. Liu, H. Lu, J. Yao, "Toward multi-database mining: Identifying relevant databases", *IEEE Transactions on Knowledge and Data Engineering* 13(4), 2001, pp. 541 - 553.

[56] J. Muhonen, H. Toivonen, "Closed non-derivable itemsets", *Proceedings of PKDD*, pp. 601 - 608, 2006.

[57] M. R. Nelson, "Data compression with the Burrows-Wheeler transformation", *Dr. Dobb's Journal*, September, 1996, pp. 46 - 50.

[58] E. R. Omiecinski, "Alternative interest measures for mining associations in databases", *IEEE Transactions on Knowledge and Data Engineering* 15(1), pp. 57-69, 2003.

[59] G. K. Palshikar, M. S. Kale, M. M. Apte, "Association rule mining using heavy itemsets", *Proceedings of Eleventh International Conf. on Management of Data*, 2005, pp. 148 - 155.

[60] A. Papoulis, *Probability, random variables, and stochastic processes*, 2nd edition, McGraw-Hill, 1984.

[61] N. Pasquier, R. Taouil, Y. Bastide, G. Stumme, L. Lakhal, "Generating a condensed representation for association rules", *Journal of Intelligent Information Systems* 24(1), pp. 29 - 60, 2005.

[62] D. Pavlov, H. Mannila, P. Smyth, "Probabilistics models for query approximation with large sparse binary data sets", *Proceedings of Sixteenth Conference on Uncertainty in Artificial Intelligence*, 2000, pp. 465-472.

[63] G. Piatetsky-Shapiro, "Discovery, analysis, and presentation of strong rules", *Proceedings of Knowledge Discovery in Databases*, 1991, pp. 229 - 248.

[64] Proefschrift, *Multi-relational data mining*, Ph D thesis, Dutch Graduate School for Information and Knowledge Systems, Aan de Universiteit Utrecht, 2004.

[65] D. Pyle, *Data preparation for data mining*, Morgan Kufmann, San Francisco, 1999.

[66] A. Savasere, E. Omiecinski, S. Navathe, "An efficient algorithm for mining association rules in large databases", *Proceedings of the 21st International Conference on Very Large Data Bases*, 1995, pp. 432 - 443.

[67] K. Sayood, *Introduction to data compression*, Morgan Kaufmann, 2000.

[68] P. Shenoy, J.R. Haritsa, S. Sudarshan, G. Bhalotia, M. Bawa, D. Shah, "Turbo-charging vertical mining of large databases", *Proceedings of ACM SIGMOD Conference on Management of Data*, 2000, pp. 22 - 33.

[69] Y. Shima, S. Mitsuishi, K. Hirata, M. Harao, E. Suzuki, S. Arikawa, "Extracting minimal and closed monotone dnf formulas", *Proceedings of International Conference on Discovery Science*, Vol. 3245, 2004, pp. 298 - 305.

[70] A. Silberschatz, A. Tuzhilin, "What makes patterns interesting in knowledge discovery systems", *IEEE Transactions on Knowledge and Data Engineering* 8(6), 1996, pp. 970 - 974.

[71] C. Silverstein, S. Brin, R. Motwani, "Beyond market baskets: Generalizing association rules to dependence rules", *Data Mining and Knowledge Discovery* 2(1), 1998, pp. 39 - 68.

[72] M. Steinbach, V. Kumar, "Generalizing the notion of confidence", *Proceedings of ICDM*, 2004, pp. 402 - 409.

[73] M. Steinbach, P.-N. Tan, H. Xiong, V. Kumar, "Generalizing the notion of support", *Proceedings of KDD*, 2004, pp. 689 - 694.

[74] K. Su, H. Huang, X. Wu, S. Zhang, "A logical framework for identifying quality knowledge from different data sources", *Decision Support Systems* 42(3), 2006, pp. 1673 - 1683.

[75] P. -N. Tan, V. Kumar, J. Srivastava, "Selecting the right interestingness measure for association patterns", *Proceedings of SIGKDD Conference*, 2002, pp. 32 - 41.

[76] H. Toivonen, "Sampling large databases for association rules", *Proceedings of the 22-th International Conference on Very Large Data Bases*, 1996, pp. 134 - 145.

[77] UCI ML repository content summary, http://www.ics.uci.edu/~mlearn/MLSummary.html.

[78] P. Viswanath, M. N. Murty, S. Bhatnagar, "Partition based pattern synthesis technique with efficient algorithms for nearest neighbor classification", *Pattern Recognition Letters* 27(14), 2006, pp. 1714 – 1724.

[79] K. Wang, S. Zhou, Y. He, "Hierarchical classification of real life documents", *Proceedings of the 1st (SIAM) International Conference on Data Mining*, 2001, pp. 1 - 16.

[80] X. Wu, Y. Wu, Y. Wang, Y. Li, "Privacy-aware market basket data set generation: A feasible approach for inverse frequent set mining", *Proceedings of SIAM International Conference on Data Mining*, 2005, pp. 103 - 114.

[81] X. Wu, S. Zhang, "Synthesizing high-frequency rules from different data sources", *IEEE Transactions on Knowledge and Data Engineering* 14(2), 2003, pp. 353 - 367.

[82] X. Wu, C. Zhang, S. Zhang, "Efficient mining of both positive and negative association rules", *ACM Transactions on Information Systems* 22(3), 2004, pp. 381 - 405.

[83] X. Wu, C. Zhang, S. Zhang, "Database classification for multi-database mining", *Information Systems* 30(1), 2005, pp. 71-88.

[84] D. Xin, J. Han, X. Yan, H. Cheng, "Mining compressed frequent-pattern sets", *Proceedings of the 31st VLDB Conference*, 2005, pp. 709-720.

[85] J. Yan, N. Liu, Q. Yang, B. Zhang, Q. Cheng, Z. Chen, 2006. Mining adaptive ratio rules from distributed data sources. *Data Mining and Knowledge Discovery* 12 (2-3), pp. 249 - 273.

[86] X. Yin, J. Han, "Efficient classification from multiple heterogeneous databases", *Proceedings of 9-th European Conf. on Principles and Practice of Knowledge Discovery in Databases*, 2005, pp. 404 - 416.

[87] M. J. Zaki, M. Ogihara, "Theoretical foundations of association rules", *Proceedings of the DMKD Workshop on Research Issues in Data Mining and Knowledge Discovery*, 1998, pp. 7:1 - 7:8.

[88] S. Zhang, *Knowledge discovery in multi-databases by analyzing local instances*, Ph D thesis, Deakin University, 2002.

[89] C. Zhang, M. Liu, W. Nie, S. Zhang, "Identifying global exceptional patterns in multi-database mining", *IEEE Computational Intelligence Bulletin* 3(1), 2004, pp.19 - 24.

[90] T. Zhang, R. Ramakrishnan, M. Livny, "BIRCH: A new data clustering algorithm and its applications", *Data Mining and Knowledge Discovery* 1(2), 1997, pp. 141 - 182.

[91] S. Zhang, X. Wu, C. Zhang, "Multi-database Mining", *IEEE Computational Intelligence Bulletin* 2(1), 2003, pp. 5 - 13.