Mining Quantitative Association under Inequality Constraints

Submitted by

Charles, Cham Lo

Department of Computing
The Hong Kong Polytechnic University
Hung Hom, Hong Kong

September 2000

A Thesis Submitted
in Partial Fulfillment of
the Requirements for the Degree of
Master of Philosophy

Abstract

The problem of discovering association rules was first introduced in 1994 by R. Agrawal and R. Srikant. In the past several years, there has been much active work in developing algorithms for mining association rules. However, in discovering the patterns, it has been realized that not all associations are of interest. It is more desirable if a user can limit the target associations by specifying different constraints. For example, a marketing personnel may only want to know which items are often sold together with a total price more than 200. That is, he is interested in association rules which satisfy a given inequality constraint for a set of quantitative items. The aim of our work is to research for new methods and algorithms to extract subtle and embedded knowledge in the database satisfying inequality constraints.

Three types of constraints are considered in our work for different data item relationships. The first type of constraints are the *inequality constraints* which consider the quantitative relationships between items. The second type of constraints are the *temporal constraints* which consider the temporal and quantitative relationships between items. The last type of constraints are the *taxonomy constraints* which consider the multi-layer relationships

between items. In our work, we consider arithmetic inequality constraints which are composed of common operators such as (+,-,$*$,$\div$). We believe they are the most common constraints and can be easily extended to other queries such as the *max(), min()* and *avg()*.

Finding the interesting associations is not the only objective of our work, we also attempt to simplify and speed up the whole mining process by making use of the arithmetic properties of the input constraints. Finally, preliminary experimental results of the proposed algorithms are also reported and discussed.

Acknowledgement

I would like to express my gratitude to my supervisor, Dr. Vincent Ng, for his continual help and encouragement throughout my MPhil study, and in the preparation of this thesis with various aspects.

I would also like to take this opportunity to thank my parents and all my schoolmates at the University for their countless support within all stages of my research study.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Overview of Association Rules

Discovering association rules is to find the possible associations between items when a user specifies a minimum support (s) and a minimum confidence (c). For example, one possible association rule can be "there are 10% of customers who made long distance calls to England last month and 80% of these customers also called USA." In this example, the support is 10% and the confidence is 80%. The items in the above rule are binary attributes and does not posses any quantitative values. Different extensions on the binary association rules have been appeared in the past several years [1, 3, 8, 14, 12]. However, all of them merely reveal the existence of different items in transactions but not the possible quantitative relationships between them.

In mining the association rules, users often find that they are not only interested in the general associations. Rather, they always have a set of conditions for the possible rules to satisfy. Recently, Srikant et. al. [26] have proposed the *MultipleJoin* algorithm and the *Direct* algorithm to discover association rules which satisfy a given boolean expression over the items of a database. The expression is used to indicate the existence or non-existence of the items. Only association rules satisfying the expression are output. In our work, we are proposing to consider another kind of constraints as illustrated in the following examples.

Example 1.1 In a medical ward, doctors keep all clinical diagnosis treatment costs of their patients in a transaction. It is important if they can reveal the cost relationship between the different treatments and clinical diagonsis. Some typical transactions for treatments and their associated costs are listed below.

- nursing:100, clinical test:30, pharmacies:165, ...
- nursing:120, injection:130, pharmacies:100, ...
- operation:220, injection:542, clinical test:60, ...

One frequent question is to find out any two treatment costs with their total sum more than a thousand units.

Example 1.2 Consider a phone company with a large collection of items, and each item represents the country and the monthly long distance charge that a client spends in calling. Some typical transactions are listed below.

- England:300, China:200, USA:200, ...
- England:200, China:100, HK:200, ...

- Japan:200, England:100, France:200, ...

A frequent question that the company would like to ask is which countries with their total amount of long distance fees accumulated are less than a given amount. This would allow a business analyst to focus on those countries with low calling activities.

Example 1.3 Consider a mobile operator with a large collection of WAP access logs. Each transaction in the log represents the browsing pattern of a mobile subscriber with the air-time the person consumed. The browsing patterns are formed by sequence of URL. A typical URL is shown below.

www.mportal.com/entertainment/singer/female/Madonna/music.wml

Some WAP access log transactions are listed below.

- url1:40, url2:90, url3:12, ...
- url2:23, url3:19, url4:8, ...
- url5:12, url3:22, url4:13, ...

In order to provide the best wireless portal service, the operator has to know the favourite URLs of their subscribers. Those information can be found from their access patterns and the total amount of air-time they spent on browsing.

In our work, three types of constraints regarding to item relationship are considered. The first type of constraints are the *Inequality Constraints* which consider the quantitative relationships between items. The second type of constraints are the *Temporal Constraints* which consider the temporal and quantitative relationships between items. The last type of constraints are the *Taxonomy Constraints* which consider the multi-layer and quantitative relationships between items.

1.2 Inequality Constraints for quantitative relationships

In practice, most databases contain quantitative attributes instead of binary attributes. A user may want to find association rules which satisfy a given inequality for a set of quantitative items. The CAP algorithm, which is proposed by Raymond Ng in 1997 [20], is developed to deal with constrained association queries, such as $min(S) \geq C$ where S is a set of items and C is a constant. It shows how to optimize the mining process when the input constraints are either satisfying the *anti-monotone* or the *succinct* properties. The CAP algorithm is designed specially for queries with these properties and does not deal with inequality constraints that involved a mix of basic arithmetic operators $(+, -, *, \div)$. In most of the applications, it is often desirable to find patterns that satisfy a given inequality. For example, a doctor would like to find patterns of patients that the difference of the accumulated survival scores between treatments and clinical diagnosis are greater than a threshold. In addition, the work has been done mainly on those constraints which are either *anti-monotone* or *succinct*. For constraints which are not, such as $min(S) \leq C$, the effect may not be significant. We will describe more details about Raymond's work and other related research work in next chapter.

In our work, the input constraints are not necessary belong to specified category. They can be any inequality constraints that involved a mix of basic arithmetic operators $(+, -, *, \div)$.

The comparison can be any one in $\{<, >, =, \geq, \leq\}$, such as

- $Q_1 * Q_2 + Q_3 + Q_4 < 500$

For this type of constraints, we have developed the *Quantitative Mining on Inequality Constraint* algorithm (QMIC) to find associations (patterns) that satisfy a given inequality from the quantitative database. It will be less contributing if we only involve a constraint but not improve the mining process. In our work, regarding to QMIC, we propose two techniques to improve mining efficiency. The first technique, *Generating Sequence*, is to skip the unnecessary database scanning by exploiting the size information of an inequality constraint. For some aggregate operators and arithmetic expressions, the size of the inequality can be estimated. The second technique, *Max_Min pruning*, is to reduce the number of candidate itemsets in each iteration by exploiting the maximum and minimum values of the sub-expressions in the inequality. Details of them will be discussed in chapter 3.

1.3 Temporal Constraints for sequential relationships

The previous constraint type does not include the temporal aspect of the data. That is, instead of a quantitative value, we can have the value plus the timestamp for each item in a given transaction. Similar work has been done by Heikki in 1995 [18]. It is about a framework for discovering frequent episodes in sequential data. The framework consists of defining episodes as partial orders, and finds all episodes from a given class of episodes that are frequent enough. However, it only deals with binary attributes and does not involve any user constraints and improvement on mining efficiency.

Suppose t_{ij} is the timestamp of item I_{ij} in transaction T_i, and t_s and t_e are the starting and ending times of an interval, respectively. With the new information, we extend the work in the previous problem to handle the temporal aspect. There are two sub-problems need to be solved:

- For a given time interval IN, say $[t_s, t_e]$, find all associations such that the timestamps of all items in each association are within IN. Also, the itemsets in each association are satisfying an inequality.

- For a given time duration d, find all associations such that the difference between the earliest and the latest timestamps of the items in each association are less than, greater than or equal to d. Also, the itemsets in each associations are satisfying an inequality constraint.

We have developed the Order Pattern Mining (OPM) algorithm to work on this type of constraints that consider the temporal and quantitative relationships between items. The OPM algorithm extends the QMIC algorithm by embedding the temporal calculations in it. In OPM, itemsets with different temporal orders are treated as two distinct itemsets even though they contain the same items. We have been successfully applied the algorithm in mining the browsing patterns among WWW logs. The details will be provided in chapter 4.

1.4 Taxonomy Constraints for multi-layer relationships

It is natural to have a taxonomy in the database. One good example is the clinical database. There are surgical data, therapeutic data, diagnostic data, ward resource data, etc. For each data group, there are different levels of sub-categories and sub-groups. It will be interesting to find the multi-layer associations amongst the items. Similar work has been done by Han and Fu in 1995 [10]. In the work of Jonghyun in 1997 [13], the proposed Count Propagation algorithm (CPA) finds the co-occurrences of the items among a taxonomy database. With the taxonomy, the CPA algorithm speeds up the mining process by propagating the counts of itemsets to their ancestors in the taxonomy, instead of counting individual itemsets separately. In our work, we concern both the quantitative and multi-layer relationships amongst the items. For example, in the QMIC algorithm, we only deal with the quantitative relationships. However, the taxonomy of the data has not been considered. In the QMIC algorithm, there is no clear type nor category for each data item. This may not be accurate in reality. Therefore, we have developed the *Multi-level Mining on Inequality Constraint* algorithm (MMIC) for this new type of constraints. In our proposed algorithm, we use an acyclic graph to represent item hierarchies. Each intermediate node in a graph represent a particular data category. We first find if there are associations of the intermediate nodes satisfying both the frequency and inequality constraints. If one of the constraints fails, there is no need to probe further. It is because when the whole category fails a constraint, none of its sub-category do. Therefore, if any intermediate node fails the constraints, we can prune away the whole subtree roots at that intermediate node. We have also defined the *Inequality Expression Tree* (IET) structure which is a new technique to improve the efficiency of the pruning process. The IET will not only simplify the pruning process but also generate the *Enchanced Generating Mining Sequence* (EGMS). The details of those techniques will be in chapter 5.

1.5 Dissertation Organization

The rest of this dissertation will be divided into the following chapters. Chapter 2 reviews the literature of related work. Chapter 3 describes our basic QMIC algorithm that mines the quantitative relationships between data items. Chapter 4 and 5 describe the mining of temporal and muli-level relationships respectively. Chapter 6 discusses the experiments and their parameters. It also presents the corresponding results. Chapter 7 concludes the dissertation and suggests further work.

Chapter 2

Related Work

Database mining has recently attracted tremendous amount of attention in database research because of its applicability in many areas. Many interesting and efficient data mining algorithms have been proposed. They improve the data mining process by two ways: finding the more useful association rules or speeding up the mining process of association rules. In this chapter, we will discuss different previous works since they have aspired our proposed algorithms.

2.1 Mining Association Rules between Sets of Items in Large Database

Although this paper was written in 1993, it provides the fundamental idea for mining association rules [3]. The paper proposed the famous Apriori algorithm which can be divided into two phases.

- Find all itemsets whose supports are greater than the minimum support s and these itemsets are called *large itemsets.*

- Generate the association rules. If {ABC} and {AB} are both in the large itemsets, we compute its confidence which is (the support of {ABC})/(the support of {AB}). If the ratio is higher than a preset threshold r, the rule is established.

Obviously, the first phase is the core of the algorithm. To generate the large itemsets L_k, we need to have the candidate set C_k first. The idea is as followed: given L_k we can generate a superset of L_{k+1} which we call the candidate set C_{k+1}. C_{k+1} is formed by merging the itemsets in L_k as illustrated below.

Suppose

$$X = X_1X_2 \dots X_k \text{ and } Y = Y_1Y_2 \dots Y_k$$

where X is one of the itemsets in L_k and Y is another one. One of the itemsets C in C_{k+1} will be

$$C = X_1X_2 \dots X_kY_k$$

iff $X_1 = Y_1, X_2 = Y_2, \dots, X_{k-1} = Y_{k-1}$. The proof of it is given in [2].

After C_{k+1} is generated, we have to prune those itemsets whose supports are less than the minimum requirement. So, we have to scan the database to count the number of supports of each itemset in C_{k+1}. After counting, we get the desired L_{k+1}.

Apriori gives the fundamental idea to find all the large itemsets effectively among thousands of transactions. However, the algorithm is only dealing with binary attributes. In practice, we deal with transactions containing items with positive values instead of 0 or 1. Also, Apriori only focuses on the existence of items without performing any manipulation on the input.

2.2 Discovering frequent episodes in sequences

The paper considers the problem of recognizing frequent episodes in sequences of events [18]. An episode is defined to be a collection of events that occur within time intervals of a given size in a given partial order. Once such episodes are known, one can produce rules for describing or predicting the behavior of the sequence. The proposed algorithm has two alternating phases: 1) building new candidate episodes; 2) evaluating how often these occur in the sequence. The efficiency of the algorithm is based on three observations.

1. If a serial episode ABC is frequent, with the same window width the serial subepisodes AB, BC, AC are also frequent. This holds in general: all subepisodes are at least as frequent as the superepisode. The algorithm utilizes this observation in the opposite direction. It is only necessary to test the occurrences of episodes whose subepisodes are all frequent.
2. Episodes can be recognized efficiently by "sliding" a window on the input sequence. Typically, when two adjacent windows have a lot of overlap, they are very similar to each other. The algorithm takes advantage of this similarity: after recognizing episodes in a window, it makes incremental updates in the data structures to recognize the episodes that occur in the next window.
3. Recognition of a complex episode can be reduced to the recognition of simple ones: every episode can be seen as a recursive combination of parallel and serial episodes.

The methods have been applied to a telecommunication network fault management database and the preliminary results are quite encouraging. However, observation (1) uses the frequency constraint of Apriori for the pruning process only. It can be further improved by introducing new pruning techniques. In our proposed alogorithms, the new pruning techniques will be introduced to speed up the process.

2.3 Mining Optimized Association Rules with Categorical and Numeric Attributes

The idea in [23] is to generalize the optimized association rules problem in three ways: 1) association rules are allowed to contain disjunctions over uninstantiated attributes, 2) association rules are permitted to contain an arbitrary number of uninstantiated attributes, and 3) uninstantiated attributes can be either categorical or numeric. The generalized association rules enable users to extract more useful information about seasonal and local patterns involving multiple attributes.

Since the problem of computing optimized rules is intractable, effective mechanisms are developed for both, exploring as well as pruning the search space. A *weight* is assigned to each instantiation, and the Native algorithm considered instantiations in the decreasing order of their weights. Thus, based on input parameters such as minimum support and number of disjunctions, by appropriately assigning weights to instantiations, the exploration of the search space can be guided efficiently. In addition, the paper also proposed a general depth first algorithm that keeps track of the current optimized rule and uses it to prune the search space. For categorical attributes, it proposed a graph search algorithm that uses intermediate results to eliminate paths that cannot result in the optimized rule. For numeric attributes, it developed an algorithm for pruning instantiated rules prior to performing the search for the optimized confidence rule.

2.4 Mining Association Rules with Item Constraints

In [26], it considers the problem of integrating constraints that are boolean expressions over the presence or absence of items into the association discovery algorithm. It also revealed that *taxonomies* (is-a hierarchies) over items are often available. Examples are like says "Jacket is-a Outerwear", "Ski Pants is-a Outwear", "Outerwear is-a Clothes", etc.When taxonomies are present, users are usually interested in generating rules that span different levels of the taxonomy. It allows the elements of a boolean expression to include ancestor(item) or descendant(item) rather than just a single item. For example,

$$(Jacket \wedge Shoes) \vee (descendants(Clothes) \wedge \neg ancestors(HikingBoots))$$

expresses a constraint that looks for any rules that either contain both "Jackets" and "Shoes", or contain "Clothes" or any descendants of clothes and do not contain "Hiking Boots" or "Footwear".

Algorithm *Cumulate* is developed for mining association rules with taxonomies. This algorithm adds all ancestors of each item in the transaction to a new transaction, and then runs the Apriori algorithm over these "extended transactions". It also performs two optimizations. First, including the addition of only ancestors which are present in one or more candidates to the extended transaction. Second, not counting any itemset which includes both an item and its ancestor. The basic structure and operation of Cumulate are similar to those of Apriori, the performance was not affected much because of the present taxonomies. In this paper, the boolean expression only reveals the existence of items but it may not be interested enough for most of the users. In addition, the algorithms only deal with the binary attributes which limit the range of its usage.

2.5 Exploratory Mining and Pruning Optimizations of Constrained Association Rules

The Apriori algorithm suffers from the shortcomings such as the lack of user interaction and focus. As we mention before, users may only want to focus the generation of rules for a small subset of candidates. It seems that the traditional algorithm gives too much information. In [20], the CAP algorithm is developed to deal with the constrainted association queries. The algorithm shows how to optimize the mining process when the input constraints are either *anti-monotone* or *succinct* or both.

In the CAP algorithm, the *anti-monotone* constraints (see Table 2.1) share a similar property that can be incorporated in the pruning process just like the frequency constraint. The anti-monotone property is defined as the below.

Definition 1 Anti-monotonicity

A 1-var constraint C is anti-monotone iff for all sets S, S':

$$(S \supseteq S') \text{ and } (S \text{ satisfies } C) \Rightarrow S' \text{ satisfies } C$$

For any given anti-monotone constraints (C_{am}), an optimization can exploit the following property for better pruning of candidate itemsets.

min(S) $>$ constant
max(S) $<$ constant
count(S) $<$ constant

Table 2.1: Anti-Monotone 1-var constraints.

For a set S where $|S| = k$ which satisfies $C_{am} \Rightarrow$

$$\forall S' \subseteq S \text{ where } |S'| = k-1 \text{ and } S' \text{ satisfies } C_{am}$$

Besides the *anti-monotone* constraints, there is the *succinct* constraints that allows pruning to be done once-and-for-all before any iteration takes place. To explain how it works, we have to understand the next definition:

Definition 2 Succinct

1. *SAT_c(Item) is the set of itemsets that satisfy C. It represents the pruned space consisting of those item sets satisfying C.*

2. *SAT_c(Item) is a succinct powerset if $Item_1, \ldots, Item_k \subseteq Item$ such that SAT_c(Item) can be expressed in terms of the strict powersets of $Item_1, \ldots, Item_k$ using union and minus.*

3. *1-var constraint C is succinct provided SAT_c(Item) is a succinct powerset.*

It has been shown that succinctness is a sufficient condition for pre-counting prunability. A succinct constraint can be simply operated in a generate-only environment instead of in a generate-and-test environment. In fact, the algorithm CAP achieves a remarkable speedup of the mining process by using the two constraint properties in the pruning process. However, some questions, as those listed below, are remained opened.

1. CAP is dealing with 1-var constraint. It must be modified for other types of constraints, such as inequality constraints involving sums and differences.
2. For constraints that are not *succinct* nor *anti-monotone*, the effect brought by the algorithm may not be significant.
3. Most of the constraints containing the operators $(+, -, *, \div)$ are neither *anti-monotone* nor *succinct*.

2.6 Can we Push More Contraints into Frequent Pattern Mining?

In this paper [22], the authors extend their scope to integrate two mining methods, constrained mining and highly efficient frequent pattern mining and develop a constraint frequent pattern mining method. A new class of constraint, called convertible constraint is identified and handled. The paper also categorizes all the constaints into five classes, succinct, anti-monotone, monotone, convertible and inconvertible. It shows that the first four types can be pushed deep into the frequent pattern mining process. This covers most of the constraints popularly encountered and composed by SQL primitives.

Constrained Frequent Pattern Growth, CFG, an integrate new constraint-based frequent pattern mining method, is proposed for integration of multiple classes of constraint in frequent pattern mining. The performance study shows that CFG achieves significant performance improvement over previously proposed constraint based mining algorithm.

The paper integrates constraint pushing and frequent pattern growth mining into one unified framework. It leads to further improvement of mining efficiency. However, the framework still does not cover all constaints types. As the authors mentioned, the inconvertible constraints can be pushed into the mining process. In our approach, we focus to cover all the common of constraint that containing the operators $(+, -, *, \div)$. All those constraint can be pushed to the pruning process and speed up the whole mining process.

2.7 Mining Frequent Patterns without Candidate Generation

In [11], the authors suggests that the bottleneck of the *Apriori* method is at the *candidate set generation and test.* If one can avoid generating a huge set of candidate patterns, the performance of frequency pattern mining can be substantially improved. This problem is attacked in three aspects in [11].

First, a new data structure, called frequent pattern tree, is used to store quantitative information about the frequent patterns. Only frequent length-1 items will have nodes in the tree in order to ensure the tree is compact and informative. The tree nodes are specially arranged such that the more frequently occurring nodes will have better chance of sharing nodes than less frequently occurring ones. The frequent pattern tree offers the mining method a much smaller data set to work on. Then, the *FP-tree-based pattern fragment growth* mining method, FP-growth in short, is developed. It examines only its conditional pattern base, constructs its frequent pattern tree and performs mining recursively with such a tree. Since the frequent itemset in any transaction is always encoded in the corresponding path of the frequent pattern tree, pattern growth ensures the completeness of the result. The proposed method is not *restricted generation-and-test* but *restricted test only.* It is much less costly than candidate generation and pattern matching operation in most of mining algorithms. Third, it employs a partitioning-based, divide-and-conquer search technique. This dramatically reduces the size of conditional pattern base generated at the subsequent level of search as well as the size of its corresponding conditional frequent pattern tree. Moreover, it transforms the problem of finding long frequent patterns to look for shorter ones then concatenating the suffix.

This paper proposes a fast and effective mining method to avoid candidate itemset generation. Its idea speeds up the mining process but the generated result may not be of user interest. In our approach, we look for associations that are of users' interest and simplify the candidate generation at the same time as well.

2.8 Discovery of Multiple-Level association Rules from Large Databases

In this paper, a top-down progressive deepening method is developed for mining multiple level association rules [10]. The method uses a hierarchy information encoded transaction table, instead of the original transaction table, in iterative data mining. It has several ideas. (1) It reveals that most data mining queries are relevant to only the subsets of database transactions such as people are interested in can food items in the supermarket database. It is more effective to first collect the relevant set of data then query repeatedly on the relevant set. (2) Encoding can be performed during the collection of relevant data and there is no extra "encoding pass" required. (3) The encoded string which represents a specific node in a hierarchy requires less bits than the corresponding object-identifier. Encoding makes more items to be removed, which further reduces the size of the encoded

TID	Items
T_1	{111,121,211,221}
T_2	{111,211,222,323}
T_3	{112,122,221,411}
T_4	{111,121}
T_5	{111,122,211,221,413}
T_6	{211,323,524}
T_7	{323,411,524,713}

Table 2.2: Encoding Table

transaction table. A sample encoding table is shown in Table 2.2. The paper showed that it is often beneficial to use an encoded table. Also based on different sharing techniques, a group of algorithms, notably, ML_T2L1, ML_T1LA, ML_TML1 and ML_T2LA, have been developed.

The paper provides a fundamental idea in mining association rules with the multiple level concept. The encoding scheme mentioned simplies the mining process. However, in practice, users may require more specific knowledge of the database. In our preliminary work, we have developed the MMIC algorithm to handle constraints that contain the operators $(+, -, *, \div)$ for multi-level items.

Chapter 3

Inequality Constraints for Quantitative Relationships

In the past several years, many research works for association rule mining have been proposed. We found out that there are still one area which is less of concerned, the mining of associations which satisfy an user specified constraint over the quantitative database. We are interested in the inequality constraints which consider the quantitative relationships between items [16, 17]. The mining method we introduce will be the fundamental block for the mining of temporal and multi-level associations in later chapters.

3.1 Problem Statement

We now formally define our first problem. Let $V = I_1, I_2, \ldots, I_M$ be a set of quantitative items, and T be the transactions of a database D. For each transaction t, t[k] > 0 means that t contains item I_k with the value t[k], and t[k]=0 means that I_k does not exist in t. In our work, we are interested in finding all quantitative association patterns that satisfy the following inequality S:

$$(I_{i1} \oplus I_{i2} \oplus \ldots \oplus I_{im}) \ominus (I_{j1} \oplus I_{j2} \oplus \ldots \oplus I_{jn}) \; \nabla \; C$$

where $\oplus$ is $+$ $(*)$ and $\ominus$ is $-$ $(\div)$ correspondingly, $\nabla \in \{<, >, =, \leq, \geq\}$, C is a scalar value, $I_{i1}, I_{i2}, \ldots, I_{im}$ are the m items in X, and $I_{j1}, I_{j2}, \ldots I_{jn}$ are the n items in Y[1]. In summary, we have the following six parameters for the inequality constraint:

1. size m
2. size n
3. operator $\oplus$: $+$, $*$
4. operator $\ominus$: $-$, $\div$
5. relation ∇ $(<, >, =, \leq, \geq)$
6. constant C

The followings are the parameters representing 5 differet queries,

1. $(2, 0, +, -, <, 100)$: find the association of any two items where there total sum is less than a hundred.
2. $(3, 2, +, -, >, 400)$: find the association of any five items where the difference between the sum of first three items and the sum of the last two is greater four hundred.
3. $(1, 1, \text{nil}, /, =, 2)$: find the combination of any two items where the first item is two times of the second one.

[1] Items in X and Y are disjoint and they are not fixed during the mining process

3.2 The QMIC Approach

With a given inequality constraint, two properties are observed. That is

- integers m and n provide the minimal size of the desired itemsets
- the parameters ∇ and C together imply the possible range of values after operating on the desired itemsets

In developing our QMIC algorithm, we have exploited these two properties to improve and simplify the mining process. Before our discussion on the QMIC algorithm, we first introduce several terminologies below.

1. *Pre-mining process*: the process have been done before the generation of any candidate itemsets.
2. *Generation step*: the process including the generation of candidate itemsets C_i from L_{i-1} and its pruning process in iteration i where $i = 2, 3, \ldots$.
3. *Counting step*: the support counting process of candidate itemsets C_i to form the large itemsets L_i in iteration i where $i = 1, 2, \ldots$.
4. *Maximum value itemset list* ($maxlst_i$): A sorted list of all itemsets in L_i in a descending order according to the maximum value of the sum (or product) of the items within each itemset in L_i.
5. *Minimum value itemset list* ($minlst_i$): A sorted list of all itemsets in L_i in an ascending order according to the minimum value of the sum (or product) of the items within each itemset in L_i.

3.2.1 Generating Mining Sequence

Comparing with other processes, the *counting step* is straight forward but also most time-consuming. However, it is hard to simplify since the database scanning is an avoidable step. Therefore, most algorithms, including the *Apriori* algorithm, are designed to reduce the number of candidate itemsets. However, its effect on pruning decreases gradually when the itemset size gets larger. The *pre-mining process* is placed in our algorithm with the purpose to optimize not only the *generation step* but also the *counting step.* In general, we can speed up the mining process by achieving the following two objectives.

1. Reduce the number of candidate itemsets (by *generation step*).
2. Reduce the number of scannings of the entire database (by *pre-mining process* instead of *counting step*).

After observing the first property in Section 3.1, we notice that it is not necessary to generate all itemsets with different sizes. Only itemsets of sizes m and n are important as they will form the minimal itemsets. Hence, the idea is to generate C_k from $L_{k/2}$ if ever possible. With this approach, we can skip the generation steps of $L_{k/2+1}, L_{k/2+2}, \ldots,$ and L_{k-1}. There would only be $\log_2 k$ steps in finding L_k instead of k steps. When k is not a multiple of 2, we first find out the L's up to L_w where $2^w < k < 2^{w+1}$. We then find L_k by utilizing L_w, $L_{w/2}$, $L_{w/4}$, $\ldots$, etc. For example, if k is 24, we find out L_1, L_2, L_4, L_8 and L_{16}. We can then use the found L's to find L_{24} in an iterative fashion. As a result, instead of taking 24 steps to find out L_{24}, we will only take 6 steps as illustrated below.

$$\begin{array}{ll} L_1 \rightarrow L_2 \rightarrow L_4 \rightarrow L_8 \rightarrow L_{16} & \text{5 steps} \\ L_{16} + L_8 \rightarrow L_{24} & \text{1 step} \end{array}$$

This method also saves many computational effort when compared with the Apriori algorithm. However, we would need to store many previously calculated L's. For example, in finding L_{31}, we need L_1, L_2, L_4, L_8 and L_{16}. If the size of the inequality is large, there is a big demand on memory requirement. To alleviate the issue, we want to store only a few L's while skipping the unnecessary steps. Suppose L_k is a result of combining L_m and L_n where m and n are as specified in S. Two integer sequences, S_m and S_n, are calculated for m and n respectively. For a given integer value m, the sequence S_m: $(s_{m_1}, s_{m_2}, \ldots, 1)$ is obtained by the following formula

$$s_{m_{i+1}} = \begin{cases} s_{m_i}/2 & \text{if } s_{m_i} \text{ is even} \\ s_{m_i} - 1 & \textit{otherwise} \end{cases} \tag{3.1}$$

where s_{m_1} equals to m initially. After finding the sequences S_m and S_n, we find out sequence S_k which equals to $S_m \cup S_n$. The reverse of the resultant sequence is then used to guide the generation of the candidate itemsets. For example, if $n = 16$ and $m = 15$, the corresponding S_m will be a sequence of integers as $15 \rightarrow 14 \rightarrow 7 \rightarrow 6 \rightarrow 3 \rightarrow 2 \rightarrow 1$. The corresponding S_n will be $16 \rightarrow 8 \rightarrow 4 \rightarrow 2 \rightarrow 1$. We merge the two sequences to form S_k which is $31 \rightarrow 16 \rightarrow 15 \rightarrow 14 \rightarrow 7 \rightarrow 6 \rightarrow 3 \rightarrow 2 \rightarrow 1$. We only take the integer 16 from the sequence S_n since L_{16} can be generated from L_{15}. Therefore, L_8 and L_4 are not needed in the merged sequence. Figure 3.1 shows the procedure of merging the subsequences.

```
1. Let $S_m = \{ s_m, s_{m-1}, \ldots, s_1 \}$, $S'_n = \{ s'_n, s'_{n-1}, \ldots, s'_1 \}$, and $s_m > s'_n$

2. if $s'_n$ exists in $S_m$ then

   • $S_k = S_m$

   • return merged sequence: $S_k$

3. for i = $s'_n$ downto $s'_1$ do

   • Include $i$ in $S_m$

   • if $(i - 1)$ or $(i/2)$ exists in $S_m$ then

      – $S_k = S_m$

      – return merged sequence: $S_k$
```

Figure 3.1: Sequence merging.

When the merged sequence is used as the generation sequence, it can be shown that no more than 3 previous L's are required in the memory. We first introduce the Lemma 1.

Lemma 1 *Given any two subsequences $S_m : s_1, s_2, \ldots, s_m$ and $S'_n : s'_1, s'_2, \ldots, s'_n$ which are generated by using Formula (3.1). If we merge these two subsequences by using the sequence merging procedure, no more than two consecutive sequence items in the final sequence will be come from the same subsequence.*

Proof. We prove the lemma by contradiction. Assume that there are more than two consecutive sequence items in the merged sequence come from the same subsequence. Without loss of generality, suppose there are 3 of them.

- *Assumption: Three consecutive integer items S_i, S_{i+1}, S_{i+2} come from the same subsequence which is generated after using Formula (3.1).*

  i.e. $S_k : 1, 2, \ldots, s'_i, s_i, s_{i+1}, s_{i+2}, s'_{i+1}, \ldots, k$

Since S_m is generated after using formula (1), s_{i+1} is equal to either $s_i * 2$ or $s_i + 1$. If s_{i+1} takes the larger value, s_{i+2} will be either $s_i * 4$ or $s_i * 2 + 1$. Since S'_n is generated by the same formula, we conclude that the followings are true,

1. The minimum value of s_{i+2} is equal to $s_i * 2 + 1$
2. The maximum value of s'_{i+1} is equal to $s'_i * 2$

For the above, we can see if s'_i is smaller than s_i, s'_{i+1} must be smaller than s_{i+2}. However, this contradicts to what we have assumed before. Hence, the lemma is proven. □

Theorem 1 *In maximum, only three previous L's, large itemsets, are needed in order to generate the next level of large itemsets in the merged sequence.*

Proof: Let I be one of the elements in the generation sequence. From Lemma 1, either $I-1$ or $I/2$ will be one of the three elements ahead of I. Therefore, if we keep any three consecutive L's in the generation sequence, we will be able to generate the next set of candidate itemsets. $\square$

After we have generated the sequence, we can start the *generation step* and the *counting step* to generate the itemsets which are specified in the sequence. Unlike Apriori, the candidate itemsets in C_k are not necessarily formed by the large itemsets in L_{k-1} now. It can be also formed by the itemsets in $L_{k/2}$. The algorithm of generating candidate itemsets at the ${s_{i+1}}^{th}$-iteration is shown in Figure 3.2.

3.2.2 Max_Min Pruning

The performance of the Apriori algorithm is mainly depended on two integers X and Y, where X is the number of the database scannings and Y is the number of candidate itemsets in each iteration. In the previous section, we have discussed how to reduce the value of X rapidly. Here, we will discuss how to reduce the value of Y.

There are two pruning possibilities in reducing the size of the candidate itemsets. The first case is when C_k is generated from L_{k-1}. In this case, for any candidate itemset c that its $(k-1)$-subset is not in L_{k-1}, c should be removed from C_k. The other situation is when C_k is generated from $L_{k/2}$. Similarly, for any itemsets that some of its $k/2$-subset is not in $L_{k/2}$, c should be removed also. There is a trade off when C_k is generated by $L_{k/2}$.

Given the generating sequence $s_1, s_2, \ldots, s_k$, and let L_{s_i} be the current large itemsets and the previous two L's itemsets, $L_{s_{i-2}}$ and $L_{s_{i-1}}$, are currently saved.

```
1. Let $sprev = s_i$ and $curr = i$
2. if $((s_{i+1} - sprev) = 1)$ then
      • For any two itemsets $x,\ y \in L_{sprev}$ where $X = X_1X_2\ldots X_{i-1}X_i$ and $Y = Y_1Y_2\ldots Y_{i-1}Y_i$
        – if ($X_1 = Y_1$ and $X_2 = Y_2$ and ... $X_{i-1}$=$Y_{i-1}$) then
            { $c = x \cup y$; Insert $c$ into $C_{s_{i+1}}$; }
   else if $((s_{i+1} - sprev) = sprev)$ then
      • For any two itemsets $x, y \in L_{sprev}$
        – if $((x \cap y) = \phi)$ then
            { $c = x \cup y$; Insert $c$ into $C_{s_{i+1}}$; }
3. Let $curr$ = $curr$ - 1
4. Let $sprev = s_{curr}$
5. Repeat from step (2) above.
```

Figure 3.2: Candidate Itemset Generation: $C_{s_{i+1}}$.

It is possible that the number of candidate itemsets is larger when compare to the C_k which is generated by L_{k-1}. To remedy the problem, we exploit the inequality constraint again and developm the Max_Min pruning method for candidate itemset pruning. The **Max_Min pruning** method is designed to reduce the number of candidate itemsets in each iteration. Note that the general form of an input inequality E is

$$E : A \ominus B \nabla C$$

where A is $I_{i_1} \oplus I_{i_2} \oplus \ldots \oplus I_{i_m}$, B is $I_{j_1} \oplus I_{j_2} \oplus \ldots \oplus I_{j_n}$, m and n are the sizes of set A and B respectively. The method reduces the number of candidate itemsets by considering the upper and lower bounds of A and B in the pruning process. It prunes away the undesired itemsets based on the following two scenarios.

1. If $\nabla \in \{\geq, >\}$,the lower bound of A is C since B is a non-negative integer. The upper bound of B is the difference between the upper bound of A and C. We use the $maxlst_i$ of A and the $minlst_i$ of B in the *max_min pruning*.
2. If $\nabla \in \{\leq, <\}$, this scenario is symmetric to the first one and the strategy is similar only that we are using the $minlst_i$ of A and the $maxlst_i$ of B.

Max_Pruning for scenario 1

As the values of the itemsets in B are non-negative, the minimum total value of the itemsets in A is C. We apply this idea in the pruning process by using the $maxlst_i$. The procedure in Figure 3.3 shows how L_1 is pruned.

```
Given $maxlst_1$ as $\{I_{1_1}, I_{1_2}, \ldots, I_{1_k}.\}$ where $I_{1_1} > I_{1_2} > \ldots > I_{1_k}$

1. Find the sum of the $I_{1_1} + \ldots + I_{1_m}$ is greater than $C$ then

    • set $maxsum_1$ equal to the sum of $I_{1_1} + \ldots + I_{1_{m-1}}$ items

    • set counter $c = m - 1$

    • repeat

        – increase $c$ by 1;

        until $I_{1_c} \oplus maxsum_1 \leq C$

    • Remove all the items whose index is larger than $c$ from $L_1$

   else

    • No itemset in the database satisfies the input constraint
```

Figure 3.3: Max_Pruning for L_1.

```
Given $maxlst_i$ as { $I_{i_1}, I_{i_2}, \ldots, I_{i_k}$ },where $I_{i_1} > I_{i_2} > \ldots > I_{i_k}$

1. if the sum of the $I_{i_1} + \ldots + I_{i_m}$ is greater than $C$ then
    (a) set $c = (\lceil (m-i)/i \rceil) + 1$
    (b) set $maxsum_i$ equal to the sum of $I_{1_1} + \ldots + I_{1_{c-1}}$ items
    (c) repeat
        • increase $c$ by 1;
        until $I_{i_c} \oplus maxsum_i \leq C$

        • remove all the items whose index is larger than $c$ from $L_i$ and form $L_i$'
```

Figure 3.4: Max_Pruning at iteration i.

After the pruning process, we have a new L_1' which contains a smaller number of itemsets than L_1. Since the size of L_1' is smaller, the number of candidate itemsets in C_2 will be reduced as well.

In the general case, after the database scanning for support count of C_i, we can find the maximum values of each itemset in L_i to form the $maxlst_i$ for better pruning. However, we observe two new complications though. First, in the max_pruning for L_1, we evaluate an item by adding it to the other $m-1$ items in $maxlst_1$. This may not be always feasible since the combination formed by the itemsets in L_i may not have its size equal to m. To avoid the over-pruning situation, the $maxsum_i$ will use the value of the combination with size k where $k - m < i$. The itemsets may be under-pruned but it can guarantee the completeness of the generation. The procedure is shown in Figure 3.4.

The second complication is the possibility of the $maxlst_i$ containing non-disjoint itemsets. For example, when $i = 2$, $maxlst_2$ can be $\{AB, AC, AE, BC, EF \ldots\}$. One may suggest to skip the overlapped itemsets. However, this may loose some qualified candidate itemsets later. The situation is illustrated in the following example.

Example 3.1 Given that m is 6 and C is 17, and

$$maxlst_2 : \{AB(10), AC(9), BE(8), EF(4), XY(2)\}$$

If we skip the overlapping itemsets, $maxsum_2$ will be 14, itemset XY will be removed from L_2 since $2 + maxsum_2$ is equal to 16 which is smaller than C. As a result, ABCEXY, a qualified candidate itemset is missed because of the elimination of XY.

Min_Pruning for scenario 1

When compared to the *max_pruning*, the *min_pruning* takes more steps to implement although the idea behind is similar. Given that the upper bound of A and the constant C are 120 and 100 respectively, the upper bound value of B should be 20. Based on the similar argument discussed before, we can prune away the large itemsets in each iteration. During the process, the upper bound of A is a constant obtained from the sum of the first m items in $maxsum_1$. It is the largest possible maximum value that any group of m items can have. The pruning procedure of the min_pruning method is shown in Figure 3.5.

```
Given $minlst_i$ as $I_{i_1}, I_{i_2}, \ldots, I_{i_k}$ ,where $I_{i_1} < I_{i_2} < \ldots < I_{i_k}$

1. initialise $c = (\lfloor (m-i)/i \rfloor) + 1$

2. set $minsum_i$ equal to the sum of $I_{i_1} + I_{i_2} + \ldots + I_{i_{c-1}}$ items

   • repeat
       increase $c$ by 1;
     until $I_{i_c} \oplus minsum_i > C - maxsum_1$

   • remove all the items whose index is larger than $i$ from $L_i$ and form $L_i$'
```

Figure 3.5: Algorithm: General Min_pruning

In the QMIC algorithm, at each iteration, the min_pruning is invoked after max_pruning until the the itemsets of either size m or n are obtained. The detail is shown in Figure 3.6.

3.3 Summary

QMIC is the fundamental algorithm in this dissertation. It contributes our work in two areas, 1)handle the inequality constraint and, 2) mining over the quantitative database. The work in following chapters is more or less the extension of QMIC. The technique in QMIC will be further developed to handle more complicated problems.

In next chapter, after exploring the quantitative relationship between items, we enchance our existing methods to explore a more interested relationship in practice, the temporal relationship between items.

```
1. $L_1$ := frequent 1-pattern;

2. GMS := Sequence Merging (shown in Figure 3.1)

3. for k $\in$ sequence presented by GMS do

    • $C_k$ := Candidate Itemset Generation (shown in Figure 3.2)

    • $C_k$ := General Max_pruning (shown in Figure 3.4)

    • $C_k$ := General Min_pruning (shown in Figure 3.5)

    • forall transaction $t \in D$ do

        – Increment the count of all candidates in $C_k$ that are contained in $t$

    • $L_k$ := All candidates in $C_k$ with minimum support

4. result = $\bigcup_k L_k$;
```

Figure 3.6: The QMIC Algorithm

Chapter 4

Temporal Constraint for Sequential Relationship

The previous problem in the last chapter does not include the temporal aspect of the data. That is, in addition to a quantitative value, we can have a timestamp for each item in a given transaction. In this chapter, we consider both the quantitative and temporal relationships between items [15]. We will explain our work through its application in mining the web access patterns.

4.1 Mining Access Patterns

The use of World-Wide-Web(WWW) has grown exponentially in past several years and companies are interested to analyze access patterns of their users for better profit and services. One common query is the access patterns of the different Web pages. In a company, for example, it is important to know what product information that a user has browsed in its WWW server. However, only knowing the associations is no sufficient, it is helpful to find out the access order of the Web pages as well. The access order represents the traversal patterns of a user and how pages are effectively linked. In a proxy server, the access order can also help to predict user behavior for better caching and prefetching of WWW pages. With the massive amount of access logs on WWW pages, it is natural to consider how current data mining techniques can assist in solving the query.

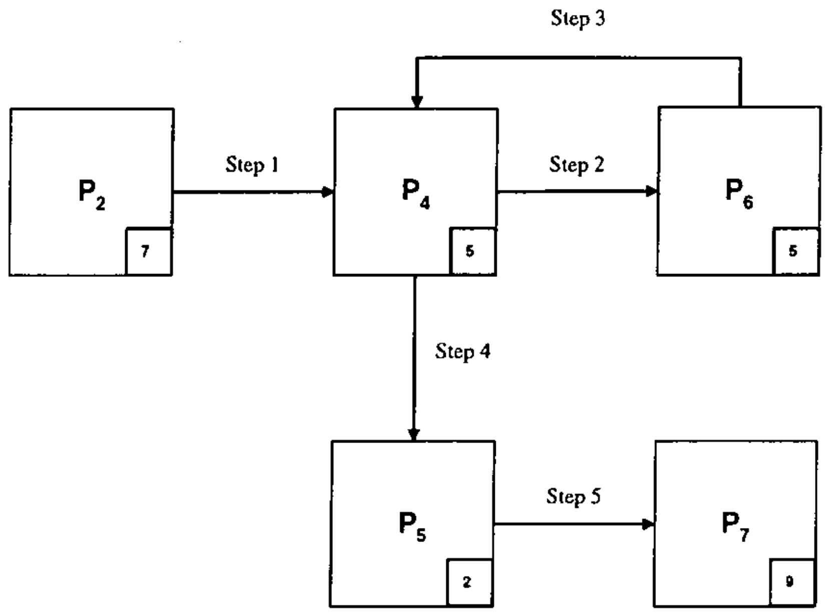
Recently, different researches of association rule have been done in the area of WWW because of its rapid growth. Some of the work [7, 6] explores the new data mining capability which involves mining path patterns in WWW. Different hashing and pruning techniques have been introduced to optimize the mining of traversal patterns. To measure a user's interest in the web pages, the viewing time has also been considered [24]. Most of the work was done for the better organization of web pages but not focused on the improvement of mining efficiency. Note that although some effort have been elaborated upon the traversal behavior, not all results are really interesting. Instead of generating all the frequent patterns, we decide to let a user defining his own interest patterns. This will not only provide the flexibility but also improve the mining efficiency significantly. Similar problem has been addressed in [5] which proposed an algorithm for efficiently extracting only the maximal frequent itemsets, where an itemset is maximally frequent if it has no superset that is frequent. In other words, it generates only the largest size of frequent itemsets which are considered as interesting in their case. However it will be more flexible and effective for the users if they can get the frequent itemsets with specified itemset size. We proposes the **Order Pattern Mining** *OPM* algorithm for directly mining the access patterns of a number of web pages while considering constraint with respect to size, orders and simple arithmetic inequalities.

```
158.132.24.196 - - [07/Oct:12:37:34] ĞET /icons/blank.gif HTTP/1.0" 200 148
158.132.24.196 - - [07/Oct:12:37:34] ĞET /icons/back.gif HTTP/1.0" 200 216
158.132.24.196 - - [07/Oct:12:37:34] ĞET /icons/folder.gif HTTP/1.0" 200 225
158.132.24.196 - - [07/Oct:12:37:34] ĞET /icons/image2.gif HTTP/1.0" 200 309
158.132.24.196 - - [07/Oct:12:37:34] ĞET /icons/text.gif HTTP/1.0" 200 229
158.132.24.196 - - [07/Oct:12:42:43] ĞET /yuen.htm HTTP/1.0" 403 288
158.132.24.196 - - [07/Oct:12:42:57] ĞET /index.html HTTP/1.0" 403 290
158.132.24.196 - - [07/Oct:12:45:12] ĞET /index.html HTTP/1.0" 403 290
158.132.24.196 - - [07/Oct:12:45:15] ĞET /yuen.htm HTTP/1.0" 403 288
158.132.24.187 - - [07/Oct:14:12:59] ĞET /_vti_in.html HTTP/1.0" 403 293
158.132.24.187 - - [07/Oct:14:13:00] ĞET /_vti_in.html HTTP/1.0" 403 293
158.132.24.187 - - [07/Oct:14:13:00] ĞET / HTTP/1.0" 403 280
158.132.24.196 - - [07/Oct:14:28:38] ĞET / HTTP/1.0" 403 7122
158.132.24.196 - - [07/Oct:14:28:39] ĞET /Images/news.jpg HTTP/1.0" 200 3507
```

Figure 4.1: WWW access log

There are different types of user paramters when we are mining the associations from the access logs on WWW pages, which is a collection of quantitative transactions. Obviously, the viewing times of the pages can be employed to measure a user's interest in the web pages. The pruning and hashing technique of OPM for the handling of these kind value constraints will be presented in later sections. Different papers on mining the quantitative association [1, 3, 8, 14, 12] have been proposed in the past several years. In our work, we focus on the mining of user access pattern in WWW pages. An example of access pattern is shown in Figure 4.1.

In the OPM algorithm, we embedded the handling of the item ordering so that unnecessary candidate itemsets can be pruned by exploiting the ordering property. Also, specific itemset size is a user constraint that OPM want to satisfy. The purpose is to reduce the



Database Transaction
(2,7), (4,5), (6,5), (4,5), (5,2), (7,9)

Figure 4.2: An Access Pattern

number of database scannings by adopting the idea of GMS which we discussed in last chapter.

4.2 Problem Statement

Before defining our second problem, we would like to introduce the concept of *equivalent patterns* and *frequent patterns*. The notations are shown in Table 4.1.

Definition 1 *S_k^i and S_k^l are both access patterns of k pages, they are equivalent iff* $P(i,a) = P(l,a) where a = 0,1,2,\ldots,k-1$.

Notation	Meaning
$d(i,j)$	duration of viewing j^{th} page in transaction i
d_m	duration of viewing page m
P_m	page m
S_k^i	access pattern i with k pages
$P(i,j)$	j^{th} page in the transaction i
$\|S_k^i\|$	number of equivalent access patterns of S_k^i in all transactions

Table 4.1: Notations for the OPM algorithm

Definition 2 *S_k^i is frequent if*

$$\frac{|S_k^i|}{\textit{Total number of transactions}} > \textit{minimum support}$$

When the number of equivalent patterns of an access pattern exceeds the minimum support, it is refered as the frequent pattern.

It is known that not all the access patterns are of interest. There can be two constraints specified by a user. They are i) the size of the patterns, and ii) the minimum/maximum total viewing time of each pattern. Specifing the size of patterns avoids the generation of non-interesting itemsets, while the viewing time reveals the interest of users in different pages. A pattern may be considered as non-interesting if its total viewing time is less than (or more than) the threshold. We define our second problem as followed.

Consider a transactions T_i,

$$\begin{aligned} T_i &= W_{i1}, W_{i2}, \ldots, W_{ik} \\ W_{ij} &= (P(i,j), d(i,j)) \end{aligned}$$

We would like to extract all the frequent patterns $S_k : W_{i,j}, W_{i,j+1}, \ldots, W_{i,j+k-1}$ with the specified size k under the viewing time constraints such that $\sum d_(i,j) \geq$ threshold or $(\sum d_(i,j) \leq$ threshold)

4.3 Order Pattern Approach

We have developed the OPM algorithm for finding all frequent access patterns. The algorithm is divided into two phases.

1. Find all access patterns whose support are greater than the minimum support. These patterns are called frequent patterns.
2. Generate the association patterns, In this phase, ABC and CAB are treated as two different patterns. We evaluate the total viewing time of the mined patterns if they exceed (or less than) the threshold.

Obviously, the first phase is the core of the algorithm. To obtain the large pattern L_k, we need to generate the candidate set C_k with the consideration of the access orders of the pages. In the Apriori algorithm, the order of the items within the itemsets are ignored during the candidate itemset generation. However in OPM, it ensures that C_k is the superset of L_k with the consideration of the item orderings. The procedure is shown in Figure 4.3, in which C_{k+1} is formed by merging the itemsets in L_k.

After C_{k+1} is generated, we have to prune away those patterns whose support are less than the minimum requirement. This counting process is very similar to the one in QMIC since it is also guided by GMS. The detail is shown in Figure 4.4. The rest of the work is simply

Suppose $S_k^i = W_{i1}W_{i2}\dots W_{ik}$ and

$S_k^l = W_{l1}W_{l2}\dots W_{lk}$ where S_k^i, $S_k^l \in L_k$

Two candidate patterns will be formed in C_{k+1}

$C_{k+1}^i = W_{i1}W_{i2}\dots W_{ik}W_{lk}$ and $C_{k+1}^l = W_{i1}W_{i2}\dots W_{lk}W_{ik}$ **iff**

$P(i,1) = P(l,1), P(i,2) = P(l,2), \dots, P(i,k) = P(l,k)$.

Figure 4.3: Candidate pattern generation

by checking the $\sum d_{i,j}$ of each pattern S_k in L_{k+1} against the viewing time constraint.

```
1. $L_1$ := frequent 1-pattern;
2. for i $\in$ sequence presented by GMS do
    • $C_k$ := New candidates of size $k$ generated from $L_{k-1}$
    • forall log transaction $t \in D$ do
        – Increment the count of all itemsets in $C_k$ that can be found in $t$
    • $L_k$ := All candidates in $C_k$ with minimum support
3. Interest access pattern = $\bigcup_k L_k$;
```

Figure 4.4: OPM: frequent pattern generation

Chapter 5

Taxonomy Constraint for Multi-Layer Relationship

In the last two chapters, we consider the associations in single level databases. However, very often, we have to deal with the multiple levels data relationships. In order to handle this kind of data, we work on the **MMIC** algorithm which considers the multi-level and quantitative relationship between items. Here, we also present how to integrate the inequality constraints into the mining process over the multiple levels database in order to improve the mining efficiency by two new techniques: *IET* and *EGMS* [17].

5.1 Taxonomy Relationship

The mining at a multi-level database is widely needed. For example, users may interest in association rules such as "there are 10% of customers who made long distance calls to London (England) last month and 70% of those customers also called New York (USA)." This type of mining has to deal with the taxonomy relationship among the data. An-

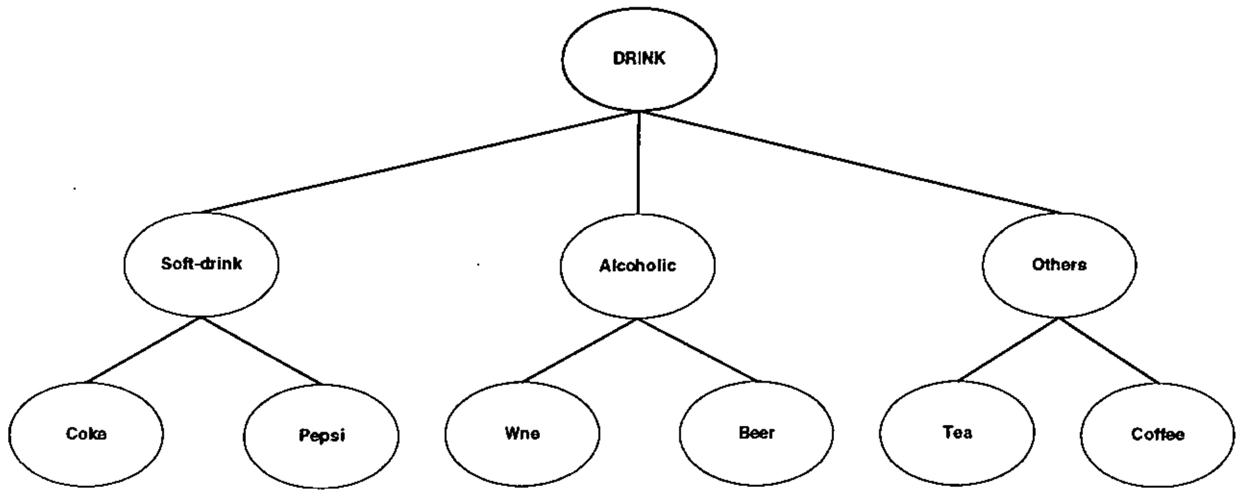


Figure 5.1: A hierarchy of supermarket.

other typical example is the multiple level database in a supermarket. A hierarchy of a supermarket's products is shown in Figure 5.1.

5.2 Problems Statement

Given the quantitative database with a taxonomy, we are interested in finding all the associations which satisfy a given inequality constraint. Unlike our first problem statement, we can now handle the constraints of arithmetic inequalities with any combination of $(+, -, *, \div)$.

Let $V = I_1, I_2, \ldots, I_M$ be a set of quantitative items, TR be their taxonomy relationship and T be the transactions of database D. For each transaction t, t[k] > 0 means that t contains item I_k with the value t[k], and t[k]=0 means that I_k does not exist in t. In our work, we are interested in finding all quantitative association rules that satisfy the user specified constraint S which consist of an *arithmetic expression*(E), an *inequality operator*(∇) and a *constant*(C). Therefore, a given constraint can be specified as $E \nabla C$, where

- E is any arithmetic expression composing of a number of unknown quantitative items and the operators $(+, -, *, \div)$ with or without brackets,
- $\nabla \in \{<, >, =, \leq, \geq\}$,
- C is a scalar value.

Some examples of the inequality constraints are given below.

1. $A + B + C > 500$ - Find any three items in the database where their sum of values is larger than 500.
2. $A = 2 * B$ - Find any pair of items where one's value is the double of the other. Or,
3. $A + (B - C) * D < 100$ - Some ambiguous users input constraint.

5.3 The MMIC Approach

In this section, we present our MMIC algorithm into two parts. First, we explain the ideas of Enhanced Generating Mining Sequence (EGMS) and Inequality Expression Tree (IET) which simplify the mining process as well as reduce the number of database scannings.

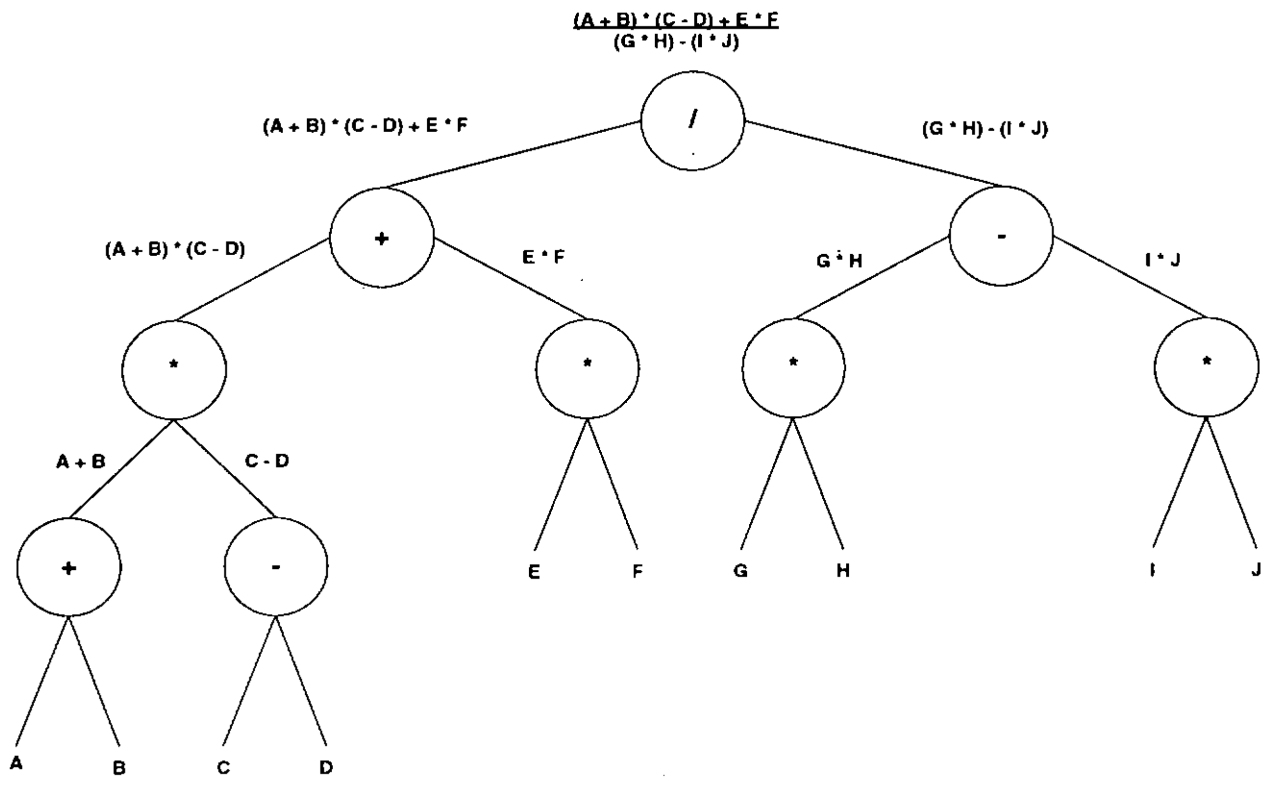
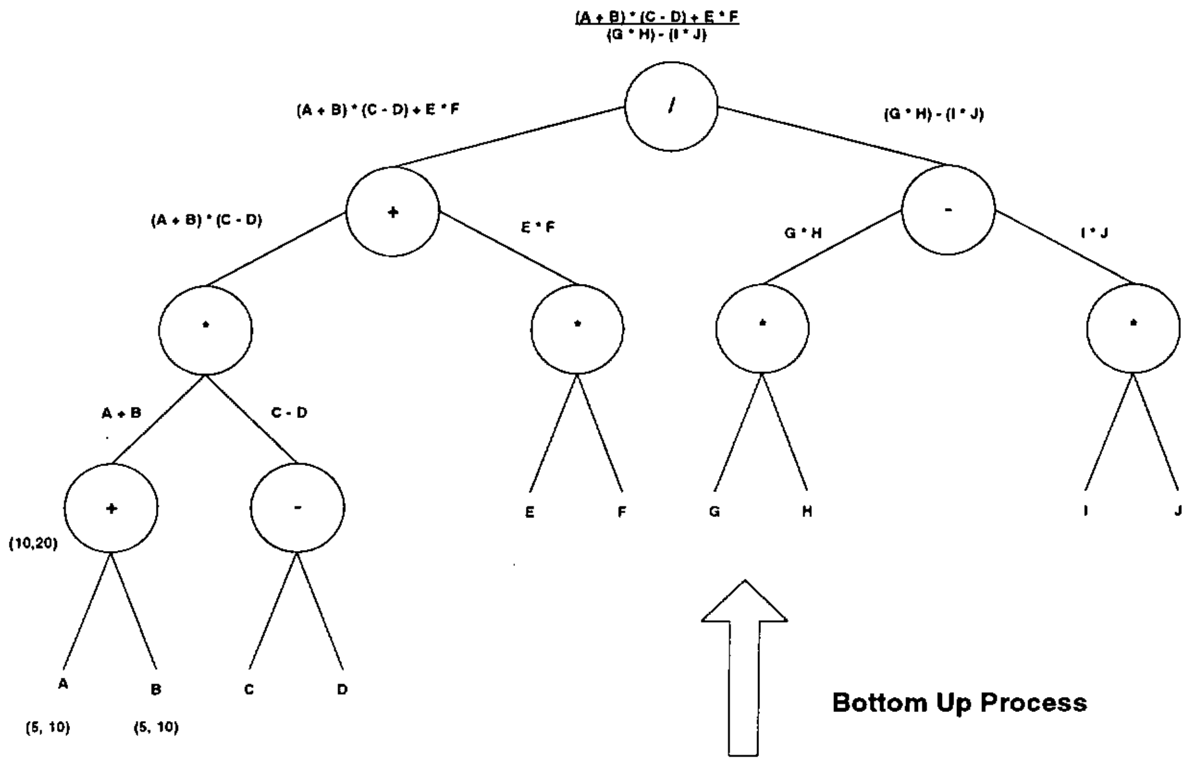


Figure 5.2: Exampe of an expression tree

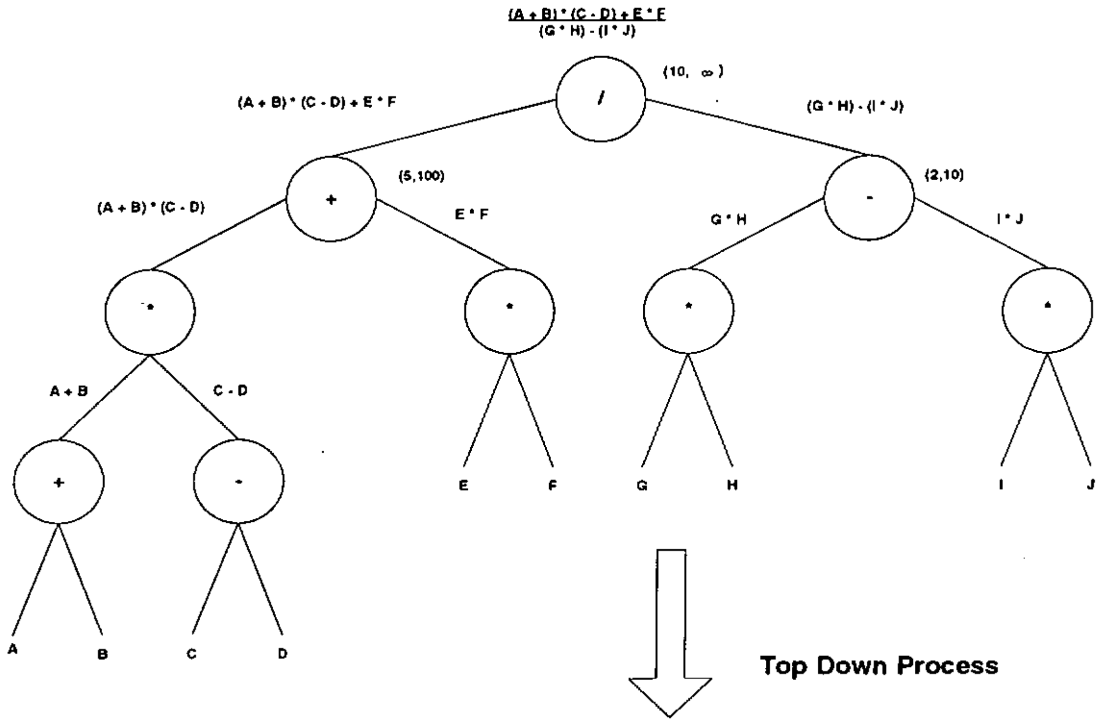
Second, we integrate these ideas with the taxonomy relationship, TR, among the data and show the details of the MMIC algorithm.

5.3.1 Inequality Expression Tree (IET)

Frequency constraint, which considers the support count of an itemset is a common criterion widely used in today's Apriori-like mining algorithm. In our MMIC, we like to consider the value constraint of the quantitative itemsets as the criterion in the pruning process. The idea is to divide the pruning process into two stages. Stage 1 defines the acceptance range (AR) for the candidate itemsets in each iteration. Whenever the value



(a) bottom-up assignment of min-max range



(b) top-down assignment of AR assumed min-max ranges of childs are (0,100) and (2,30) respectively

Figure 5.3: Building an IET.

range (VR) [1] of a candidate itemset overlaps with any AR, it can be passed to stage 2. Otherwise, it will be pruned away without the support counting. Stage 2 is simply the original support counting which makes use of the frequency constraint. We implement stage 1 by constructing an IET.

Expression Tree

There are three steps in constructing an IET. The first step is to convert an expression E of the user specified constraint into the expression tree structure. There are two interesting properties about the built expression tree in MMIC.

- A leaf node represents any possible item in the database.
- Each internal node represents a sub-expression rooted at itself. The *itemset size* of the node represents the number of items that this sub-expression has.

For example, an internal node rooted with the expression $x + y - z$ has node size equal to three. A more detail example is shown in Figure 5.2 which shows the expression tree with size equals to 10.

Minimum and Maximum Range

The second step is to assign each node a *min-max range* (Ei_{min}, Ei_{max}) represents the minimum and maximum values of the associated expression of node i. The calculation of the min-max ranges is a bottom-up process. That is, it starts at the leaf level and processes up until the root. The min-max ranges of the leaf nodes which represent the

[1]The VR represents the possible values after evaluating the itemset with the sub-expression associated with the node.

single can be simply obtained by scanning the database once. Based on the operators in the intermediate nodes, all the min-max ranges of the intermediate nodes can be calculated. In Figure 5.4 and 5.5, we show the four general formulae to calculate the ranges under the bottom-up process. All min-max ranges of the leaf nodes, which are size of 1, are the same at the beginning. The example of an expression tree with min-max ranges is shown in Figure 5.3.

Acceptance Range

The third step is the assignments of the acceptance range (AR), (E_{low}, E_{up}) to the nodes. The AR of a node is the valid range of values that an itemset can be accepted after evaluating with the sub-expression at the node. Unlike the min-max range, the AR assignment is a top-down process which starts from the root. The AR of the root can be found easily since it represents the whole left-hand-side expression of the input constraint. In our continuing example, it is $(10, \infty)$. Once the root's AR is defined, other AR's can be calculated. We show the four general formulae for the $(\geq)$ condition in Figure 5.7. The other four formulae for the $(\leq)$ are shown in Figure 5.9.

After the calculation of the ARs, the initial IET for the expression is completed as in Figure 5.3. It has a number of properties,

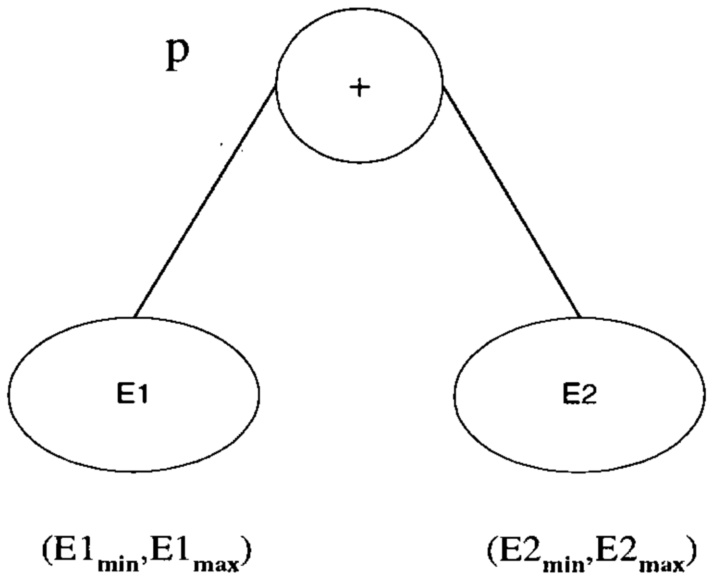
1. If the value range of a candidate itemset overlaps with any AR's of the leaf node, it is passed to stage 2 for support counting.
2. If the value range of a candidate itemset does not overlap any AR's, it is pruned.

3. During mining, each leaf node is associated with a set of large itemsets whose size is the same length as of the sub-expression.
4. The structure, especially the depth information, of the IET determines the Enchanced Generating Mining Sequence, EGMS, which will be discussed next.

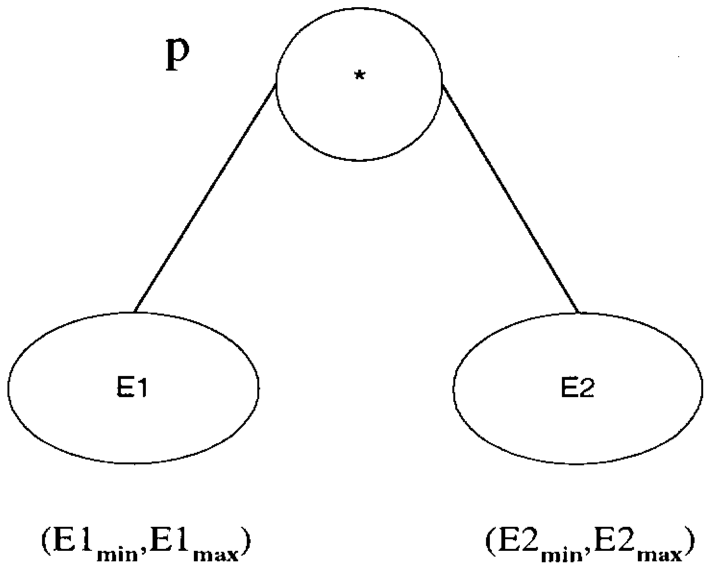
The first two properties are guaranteed by the general formulae. The rest will be useful in defining the EGMS of the mining process.

5.3.2 Enhanced Generating Mining Sequence

Just like the GMS in QMIC, EGMS is to determine the sizes of the iterations during the mining process of MMIC. In MMIC, we further shorten the generating sequence of mining. The depth of the IET determines the EGMS. Because the generation of large itemsets is done by traversing from the leaf nodes of IET to its root. For the example in Figure 5.2, the sequence from EGMS is $\{1, 2, 4, 6, 10\}$ and it is only half the length of the one in Apriori $\{1, 2, 3, \ldots, 10\}$.

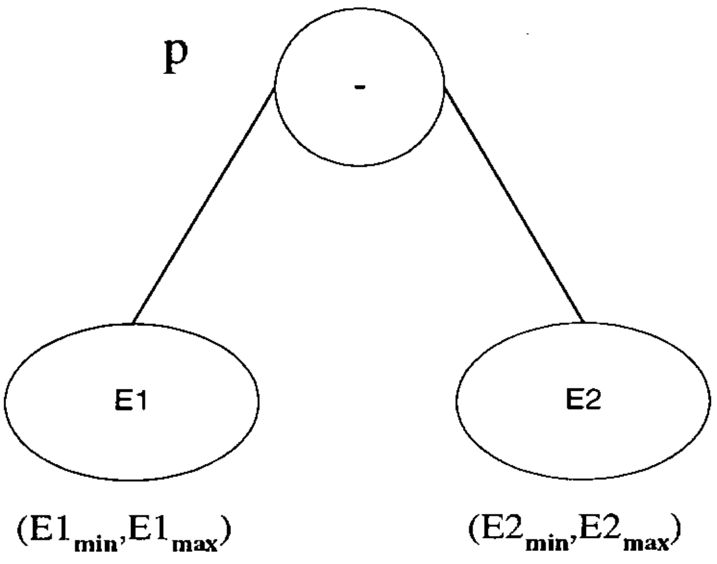


The min-max range of the parent node p is $(E1_{min} + E2_{min}, E1_{max} + E2_{max})$ respectively.

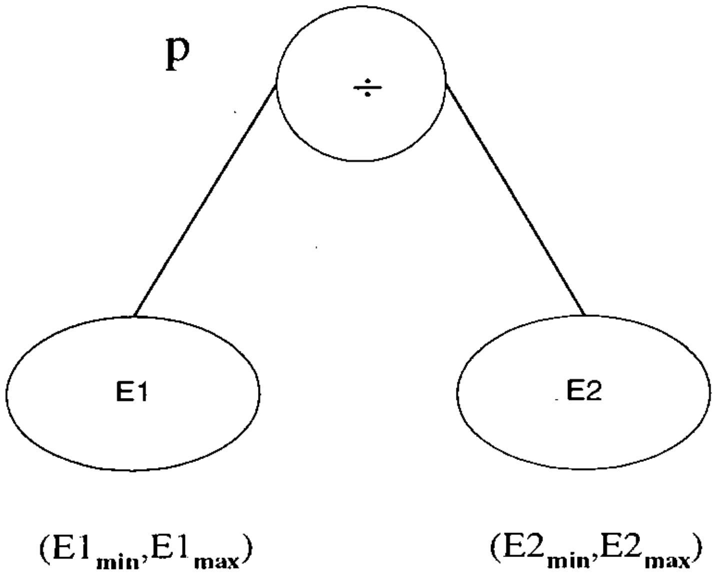


The min-max range of the parent node p is $(E1_{min} * E2_{min}, E1_{max} * E2_{max})$ respectively.

Figure 5.4: The general formulae for $(+, *)$ in the bottom-up process

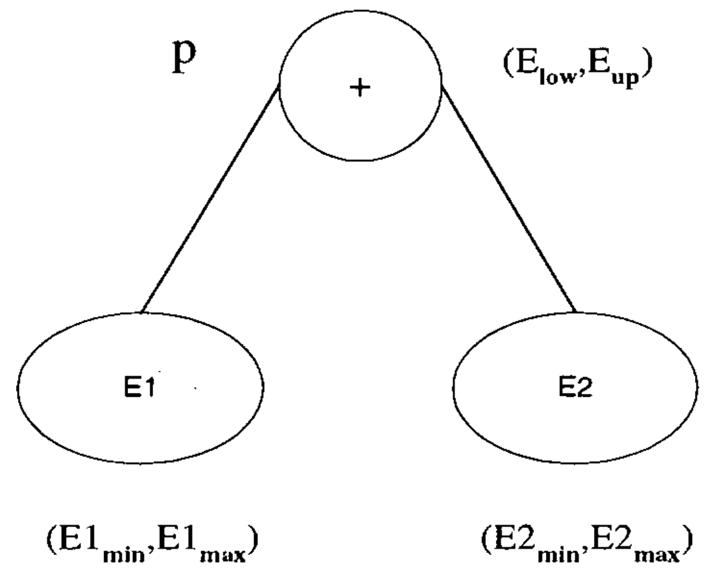


The min-max range of the parent node p is $(E1_{min} - E2_{max}$, $E1_{max} - E2_{min})$ respectively.

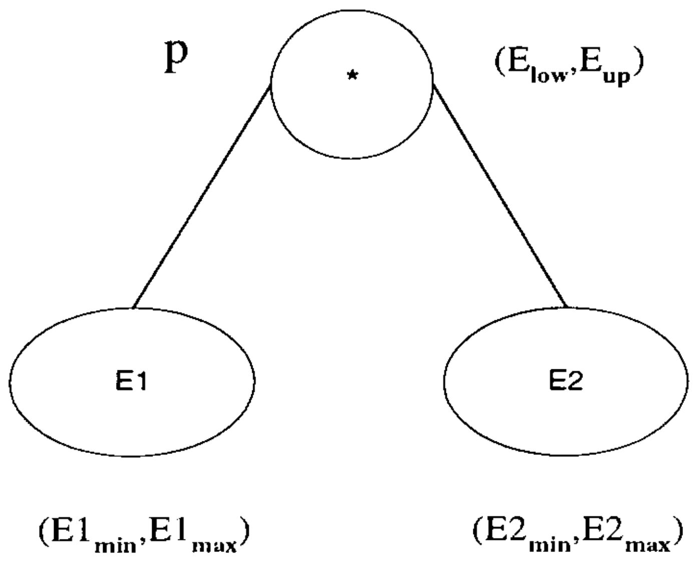


The min-max range of the parent node p is $(E1_{min} / E2_{max}$, $E1_{max} / E2_{min})$ respectively.

Figure 5.5: The general formulae for $(-, \div)$ in the bottom-up process

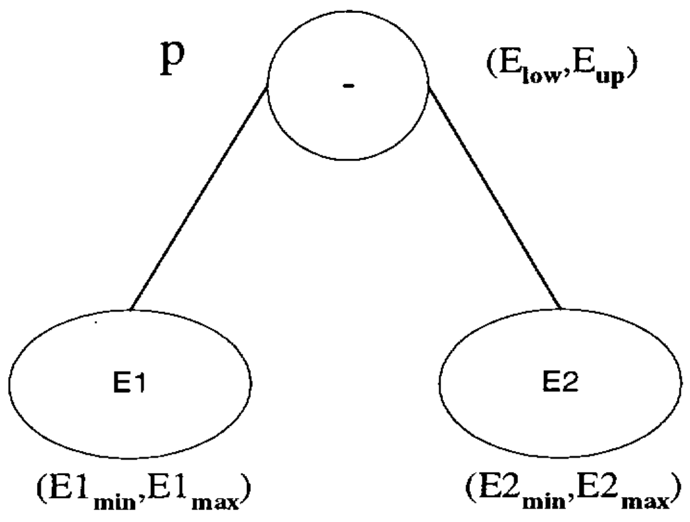


The AR range of E1 and E2 are $(E_{low} - E2_{max}, E1_{max})$ and $(E_{low} - E1_{max}, E2_{max})$ respectively.

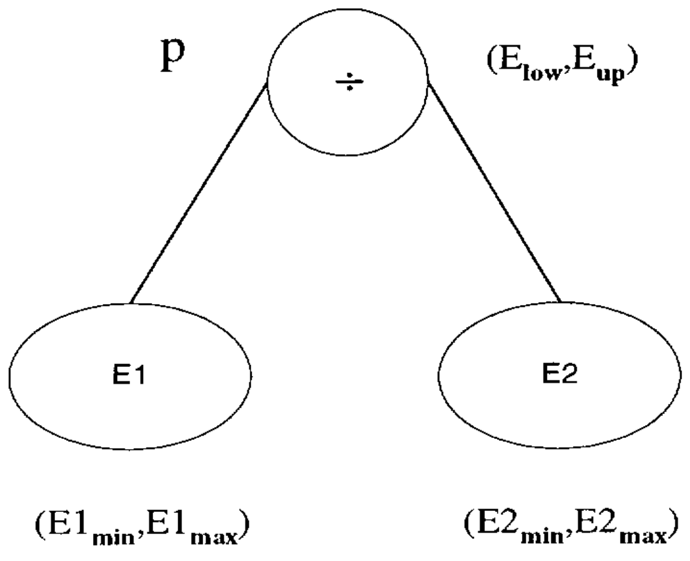


The AR range of E1 and E2 are $(E_{low} / E2_{max}, E1_{max})$ and $(E_{low} / E1_{max}, E2_{max})$ respectively.

Figure 5.6: The $(+, *)$ AR calculations for the $\geq$ condition

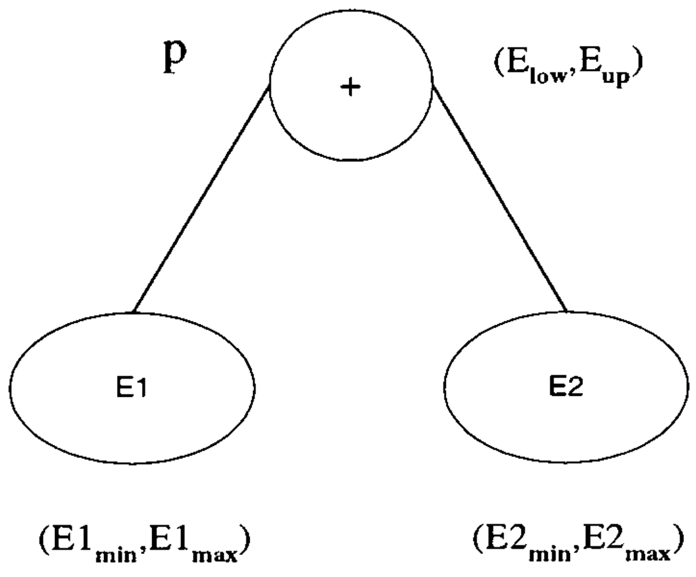


The AR range of E1 and E2 are $(E_{low} + E2_{min}, E1_{max})$ and $(E2_{min}, E1_{max} - E_{low})$ respectively.

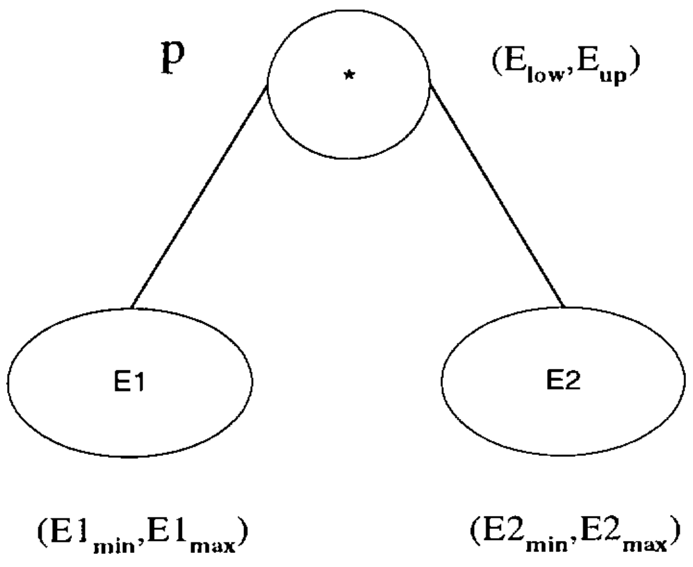


The AR range of E1 and E2 are $(E_{low} * E2_{min}, E1_{max})$ and $(E2_{min}, E1_{max} / E_{low})$ respectively.

Figure 5.7: The $(-, \div)$ AR calculations for the $\geq$ condition.

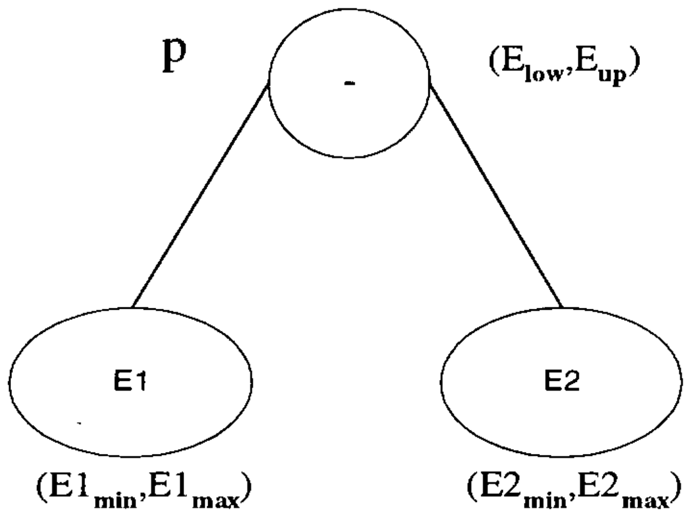


The AR range of E1 and E2 are $(E1_{min}, (E_{up} - E2_{min}))$ and $(E2_{min}, (E_{up} - E1_{min}))$ respectively.

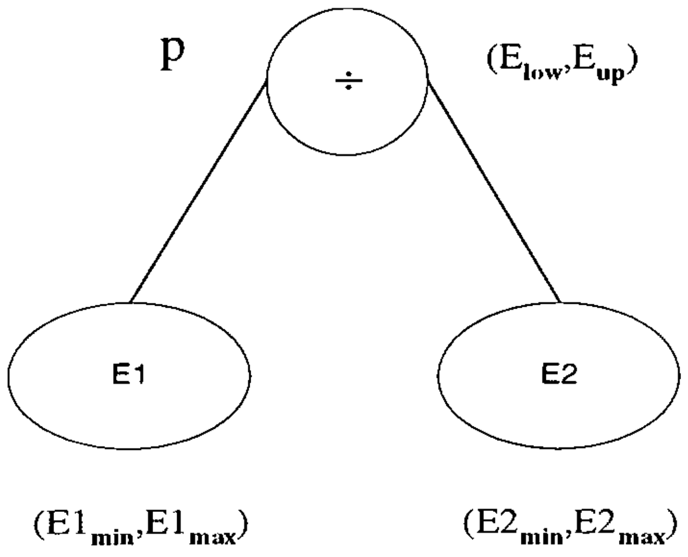


The AR range of E1 and E2 are $(E1_{min}, (E_{up} / E2_{min}))$ and $(E2_{min}, (E_{up} / E1_{min}))$ respectively.

Figure 5.8: The $(+, *)$ AR calculations for the $\leq$ condition.



The AR range of E1 and E2 are $(E1_{min}, E_{up}+E2_{max})$ and $(E1_{min} - E_{up}, E2_{max})$ respectively.



The AR range of E1 and E2 are $(E1_{min}, E_{up} * E2_{max})$ and $(E1_{min}/E_{up}, E2_{max})$ respectively.

Figure 5.9: The $(-, \div)$ AR calculations for the $\leq$ condition.

AR Pruning

In MMIC algorithm, we start from the leaf nodes of IET and use the sequence from EGMS to guide the candidate itemset generation steps. At each iteration, once the candidate itemsets with C_k are generated, their itemsets' VRs are verified with AR's in each leaf node. If the VRs of the itemsets in C_k does not overlap with any ARs, it is pruned away because it fails to satisfy the inequality constraint. By finding the support counts of the rest, we can determine L_k. Unlike Apriori, the next iteration of QMIC is not necessary L_{k+1}. It will be L_{k+l} where l is the size of the siblings of the current node. In other words, if both large itemsets associated with child nodes are available, the next iteration is to generate the large itemsets to be associated with their parent. That is why the depth of the IET defines the EGMS.

5.3.3 Multiple level pruning

In order to further improve the mining process, we consider the taxonomy relationship among the data in multiple levels database. As we mentioned before, two factors are affecting the performance of the mining process. They are the number of database scannings and candidate itemsets. The former is reduced by introducing the idea of EGMS. The latter can be also achieved implementing of IET. However, a more effective strategy is to remove the uninterested items before the real mining process takes place. The idea is based on the taxonomy relationship among the data and two simple facts.

1. In the taxonomy structure, a category item satisfies the frequency constraint if and only if all its sub-category items satisfy the same frequency constraint.

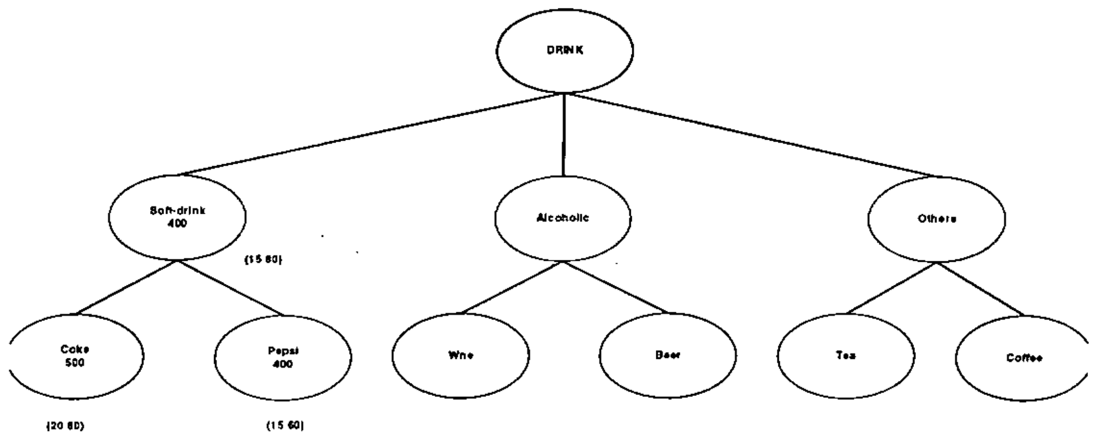


Figure 5.10: A Sample TR'.

2. A category item satisfies the value constraint if and only if all its sub-category items satisfy the same value constraint. For example in above figure, If the category item Soft-drink doesn't satisfied either the value or frequency constraint, both Coke and Pepsi are uninterested and should be pruned away before the mining process.

Base on these facts, the item pruning process of the unintested itemsets can be started even before the real mining process take place. First, we start from the bottom of the taxonomy relationship which represents the single items. Their *min-max ranges* and the support counts can be found from the first database scanning. These information can be propagated up to its parent categories until it reaches the top level. One result *TR*' is show in Figure 5.10.

Notation	Meaning
LN_m	leaf nodes with node size equal to m
LN_{mn}	specify nth leaf node in LN_m
AR_m	acceptance range set of the leaf nodes LN_m
AR_{mn}	acceptance range of the specified leaf node LN_{mn}
VR_i	value range set of all the itemsets with size i
VR_{ij}	value range of a specify jth itemset in VR_i
L_i	large itemsets with size i
C_i	candidate itemsets with size i
C_{ij}	jth candidate itemset in C_i
C_i'	candidate itemsets which satisfy the user constraint
N_k	tree node which represent the large itemsets

Table 5.1: Notations for the MMIC Algorithm.

The MMIC Algorithm

In this section, we show the complete MMIC algorithm which makes use the idea of EGMS, IET and TR' in Figure 5.11. Note that the notations used are shown in Table 5.1. The details of candidate itemset generation and its support counting is omitted because their functions are similar to those in Apriori and QMIC.

```
1. Given the database D, its $TD'$ and the completed IET, scan the database for $L_1$ and $VR_1$.
2. Find the size of input constraint, $m$
3. Define the target level from $TD'$, with the size $\geq m$ and let all the category items in that level be $L_1$
4. Repeat {
5. Itemset generation of $C_2$ from $L_1$
6. i:=2
7. Repeat {
    • delete the $LN_i$ from the IET
    • Itemset generation for $C_i$
    • for x = 1 to sizeof($C_i$)
      for y= 1 to sizeof($LN_i$)
        – if $((VR_{ix} \cap AR_{iy}) \neq \phi)$ append $C_{ix}$ into $C_i'$
        – else remove $C_{ix}$ and all its sub-categories item from $TD'$
    • Support counting of the $C_i'$ to obtain the large itemsets $L_i$
    • /*adjust the AR of the parent node of $LN_i$*/
        – for any node associated with the $L_i$
          * update the min-max range of their parent node
          * update $AR$ of parent since their min-max range has been updated
    • /*update the IET and its parameter*/
        – for any pair of leaf nodes $(LN_i, LN_j)$ which are siblings
          * i= min (sum of the node size of each pair)
    • Itemset generation of $C_i$ from the large itemsets which are represented by $LN_i$ and $LN_j$
  } while depth of the IET > 1
8. Let $L_1$ = the category items in the next level of $TD'$ } while $L_1 \neq$ empty
```

Figure 5.11: The MMIC Algorithm.

Chapter 6

Experiments

6.1 Background

In our work, we have developed three algorithms for mining different kinds of item relationships. In this chapter, we test their performance and discuss the results. We have implemented the Apriori, QMIC, OPM and MMIC algorithm in C++. We ran our preliminary experiments on a SUN workstation with 32 Mbytes of main memory. There are three sets of experiments. The first set focuses on the comparsion of Apriori and QMIC. The second set investigates the performance of OPM. The third set reveals the performances of MMIC regarding different parameters such as the change of taxonomy structure, confidence level, etc.

6.2 QMIC Performance

To evaluate the accuracy and performace of QMIC, we compare it to the Apriori algorithm. We found out that the candidate itemsets in QMIC are the superset of those in Aprioir. That prove the accuracy of the algorithm. For the performance, we notice that it is not sufficient to determine the efficiency of the algorithms by only counting the number of steps in generating the candidate itemsets. For the QMIC algorithm, it takes fewer steps to find L_k, but at the same time, it takes extra effort for Max_Min pruning process of the large itemsets at each iteration. For example, in pruning L_k, we need to verify if all of the (k/2)-subsets for an element in C_k are in $L_{k/2}$. If items are totally associated with each other, the number of comparisons will be ${}_kC_{k/2} \times_n C_{k/2}$ at each mining step. On the other hand, with Apriori algorithm, there will only be ${}_kC_{k-1} \times_n C_{k-1}$ comparisons at each iteration.

There are three groups of experiments in this set. The first set is based on a fixed number of transactions (10,000) generated by a program modified from the pseudo-data generation program in [3]. The parameters for the inequality I_E is (4,4,+,-,$\geq$,1000) and the support is set to 5%. We are interested in measuring the time to discover the constrained association rules under different numbers of items among the transactions. The results are shown in Figure 6.1. The second group of experiments is similar to the first group, except that we have fixed the number of items in the database as 1000 and experimented with different database sizes. The corresponding results is shown in Figure 6.2. We have also compare the number of candidate itemset in those two algorithms and the result is showed in Figure 6.3. Although QMIC generates more candidate itemsets, the results show us that the number of transactions has a larger impact than the number of items of the database. This is reasonable since we have to scan the whole database once for each size-k of the large itemset L_k. The QMIC algorithm skips the generation of many unnecessary candidate

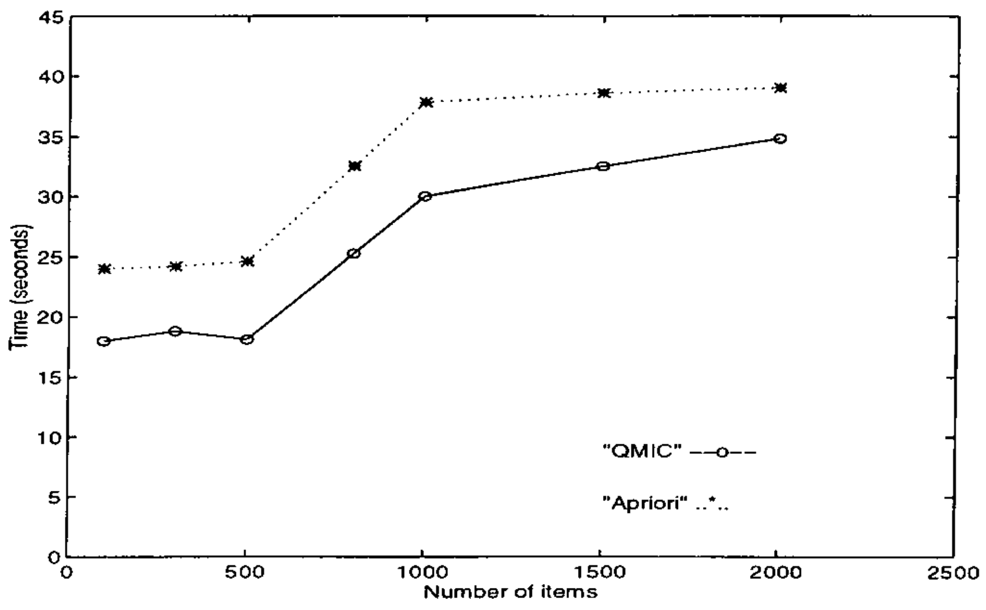


Figure 6.1: QMIC discovery times for a database of 10,000 transactions but varying numbers of items.

itemsets. So, the more transactions in the database, the algorithm can save more time. To further explain the situation, we have calculated the time saving, in percentage, of the QMIC over Apriori for inequality I_E in Figure 6.4. From the result of those experiments, we can conclude that with a larger number of transactions, the more effective of the QMIC algorithm will be. In practice, usually, the number of transactions in the database is large when compare to the number of items. Therefore, the algorithm can really speed up the mining process a lot.

6.3 OPM Performance

In this section, we would like to investigate the performance of OPM regarding to the length of the access pattern and the viewing time of the pages. There are two groups of experiments in this set. The first group focuses on the relationship between the access

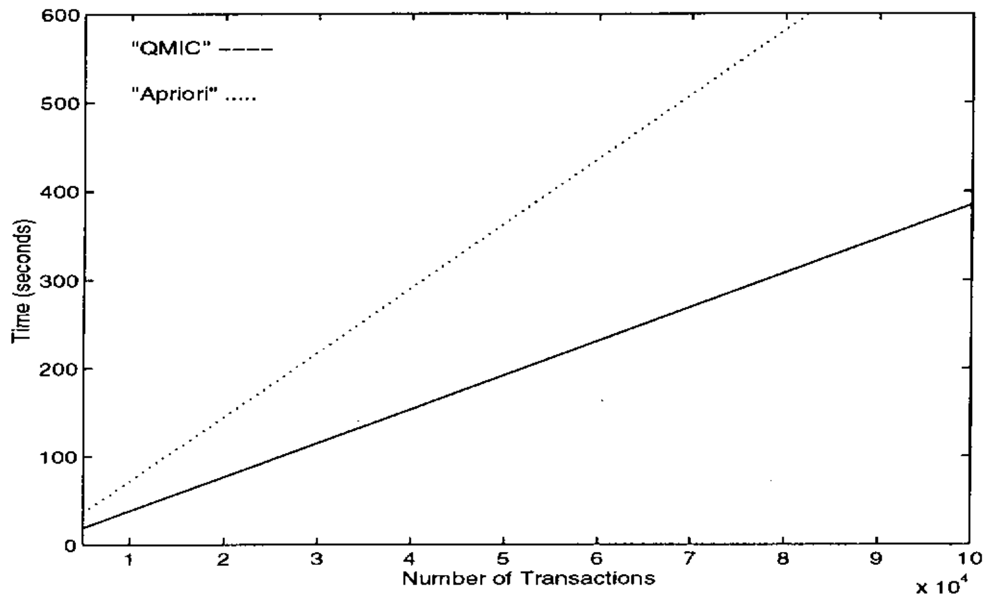


Figure 6.2: QMIC discovery times for different number of transactions for I_E.

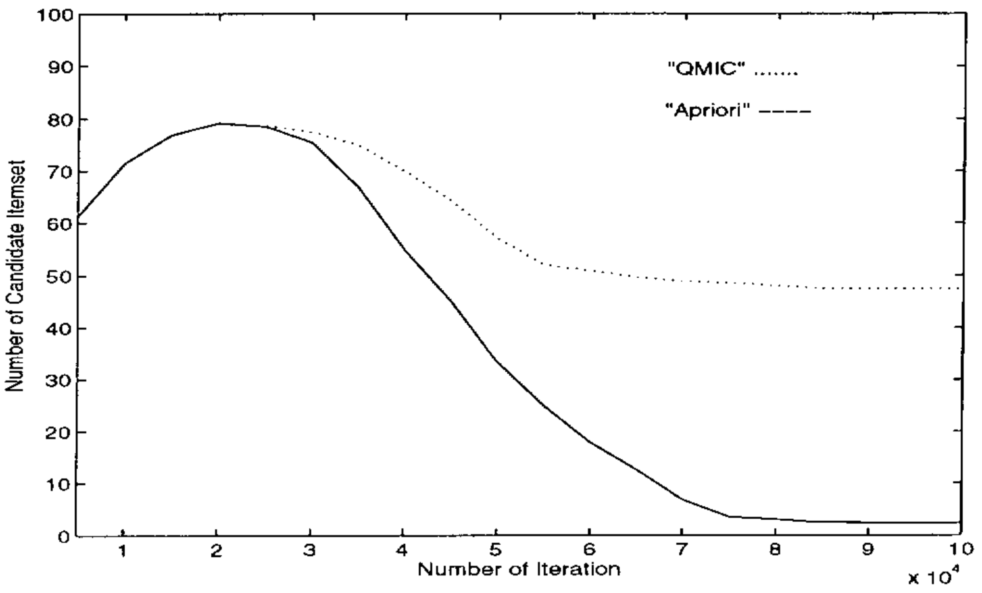


Figure 6.3: Comparison of the Number of Candidate Itemsets between Apriori and QMIC

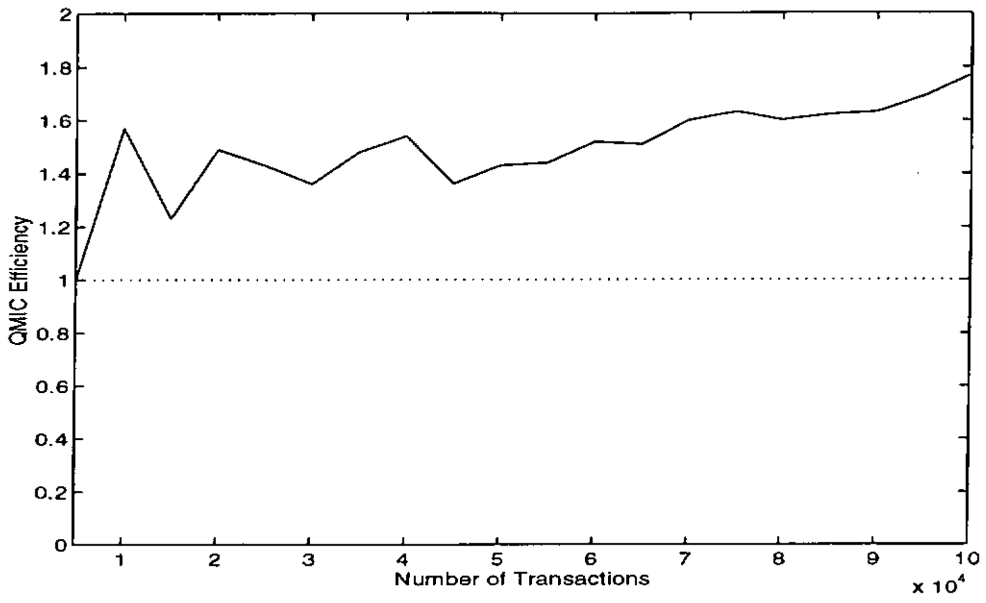


Figure 6.4: QMIC performance over Apriori for I_E.

pattern lengths and OPM performance. Suppose the minimum viewing time of each page in the patterns are 5, 10 and 20 seconds respectively. The other parameters for the experiment are:

- size of the WWW log: 100,000 transaction
- number of distinct pages : 342
- minimum viewing time for each page: 5, 10 and 20 seconds
- minimum support: 5%

With these parameters, we are interested in measuring the time to discover the temporal relationship between items under different length of access pattern items. The results are shown in Figure 6.5. From the result, we found that, OPM has better performance when mining the longer access patterns. The effect become more significant when the patterns

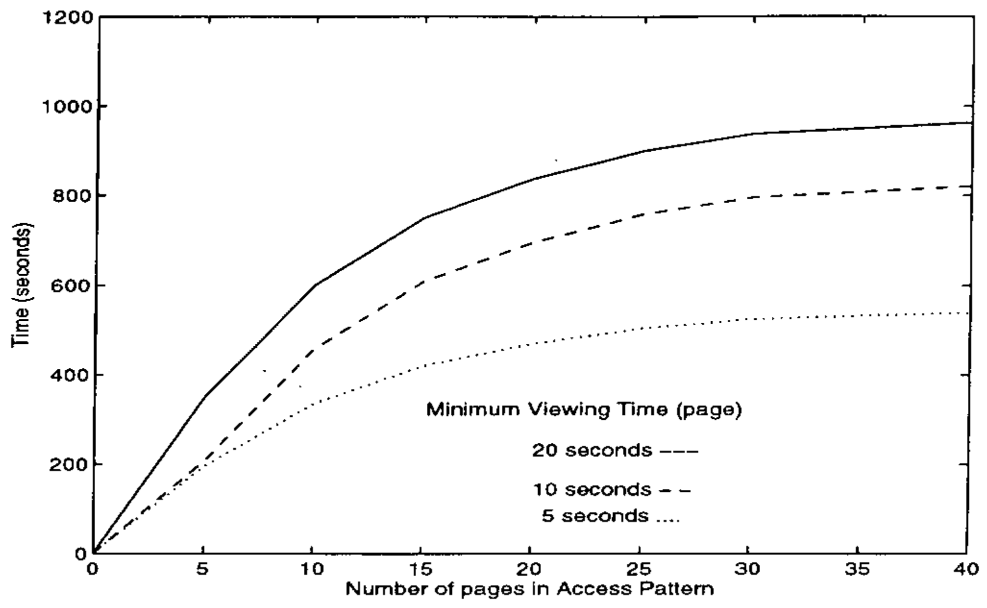


Figure 6.5: OPM discovery times for different access pattern length

have 20 pages or above.

The second group of experiment investigates the OPM performance regarding to the minimum viewing time required for each access pattern. We set the database size be 100,000, 200,000 and 500,000 transactions respectively. The parameters for the experiment are:

- size of the WWW log: 100,000 , 200,000 and 500,000 transactions
- number of distinct pages: 342
- target access pattern size: 8 pages
- minimum support: 5%

The corresponding results are shown in Figure 6.6. In addition to the frequency constraint, the minimum viewing time works as the second constraint for the pattern pruining. Obviously, the shorter viewing time allowed to each page, the more patterns will be pruned.

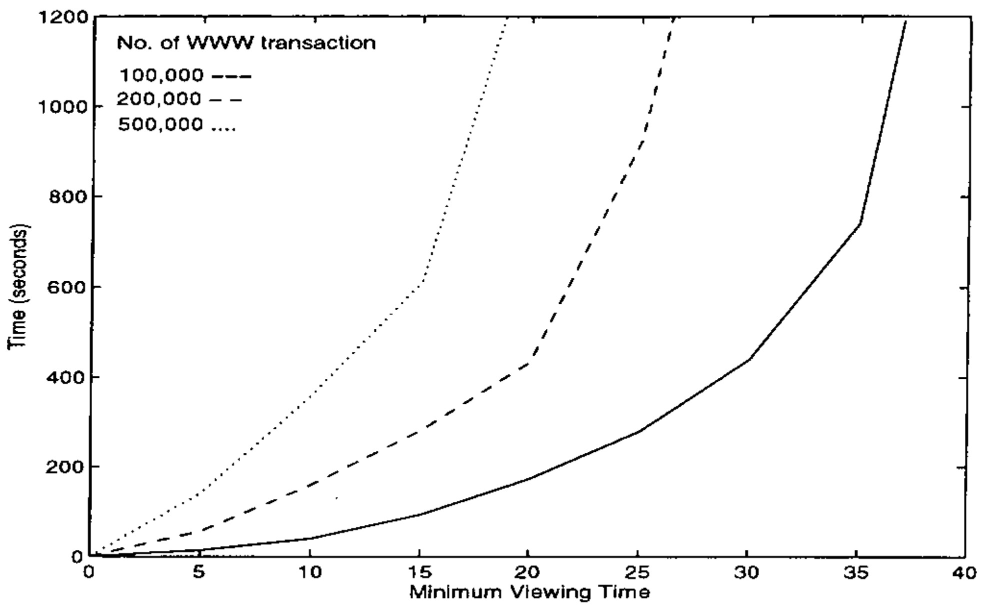


Figure 6.6: OPM discovery times for different minimum viewing required for each page

6.4 MMIC Performance

The performance of MMIC depends on different parameters such as the size of a database, number of items, length of the interested itemset, taxonomy structure and the value distribution of the items. The effect of the former three on the mininig process have been discussed in the experiments of QMIC and OPM. In this section, we would like to investigate the performance of MMIC regarding to different taxonomy structures and try to compare its performance with Apriori. Totally, there are two groups of experiments in this set.

The first group of experiments investigate the relationship between the depth of the taxonomy, *TR*, and the performance of MMIC. Regarding to different types of databases, we have different *TRs*. For example, in long distance call business, the depth of *TR* can be 3 only (ie. continent⟶country⟶city). However, the depth of *TR* in a su-

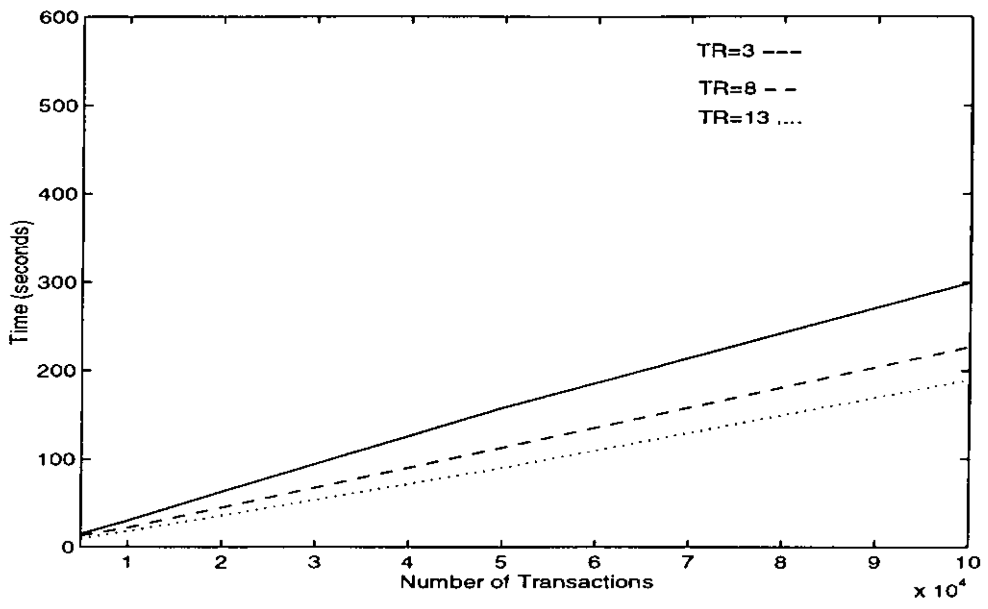


Figure 6.7: MMIC performance on TR with depth = 3, 8 and 13

permarket may not be that simple. It can be 10 or even more (ie. drink⟶ non-alcoholic⟶softdrink⟶ ... ⟶pepsi). The experiment investigated the performance of MMIC on the long distance call data regarding to different sizes of the database. Three simulating TRs have been tested, the depths are 3, 8 and 13 respectively. The experiment parameters are shown below:

- depth of the TR: 3, 8 and 13 respectively
- number of distinct items: 500
- user input constraint $(I_1 + I_2 * I_3 - I_4 \div I_5 \geq 100)$
- minimum support: 5%

We found out that, MMIC has a better performance in mining the supermarket-like database. In other word, the more levels we have in TR, the more efficency will be gained

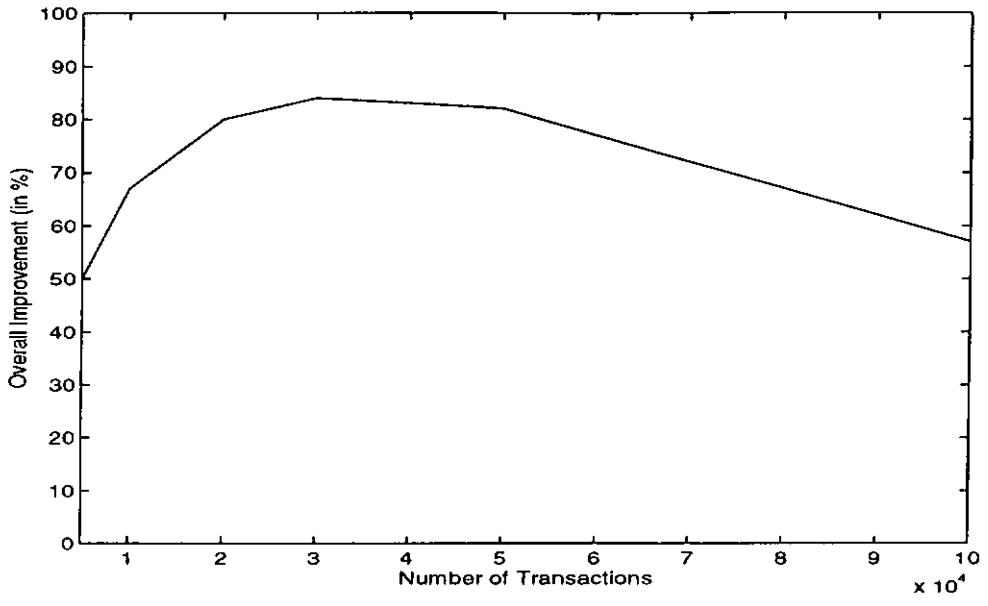


Figure 6.8: Comparison of MMIC performance with TR=3 and TR=13

from MMIC. It is because many non-interested categories are pruned earlier in the process and all their subsets are eliminated at the same time without examination. This reduces the number of candidate itemsets and further speeds up the mining process. The result will be more significant when we compare the TR with depth equal to 3 and the TR with depth equal to 13. The result of the comparsion is shown in Figure 6.8.

The second group of the experiments compared the performance between MMIC and Apriori. In mining the taxonomy data, the support count of the category is evaluated in each iteration of MMIC. We compared MMIC and Apriori regarding to different TRs in the database. The parameters of the experiment are shown below and the result is given in Figure 6.9.

- number of transactions in the database: 100,000
- number of distinct item: 500

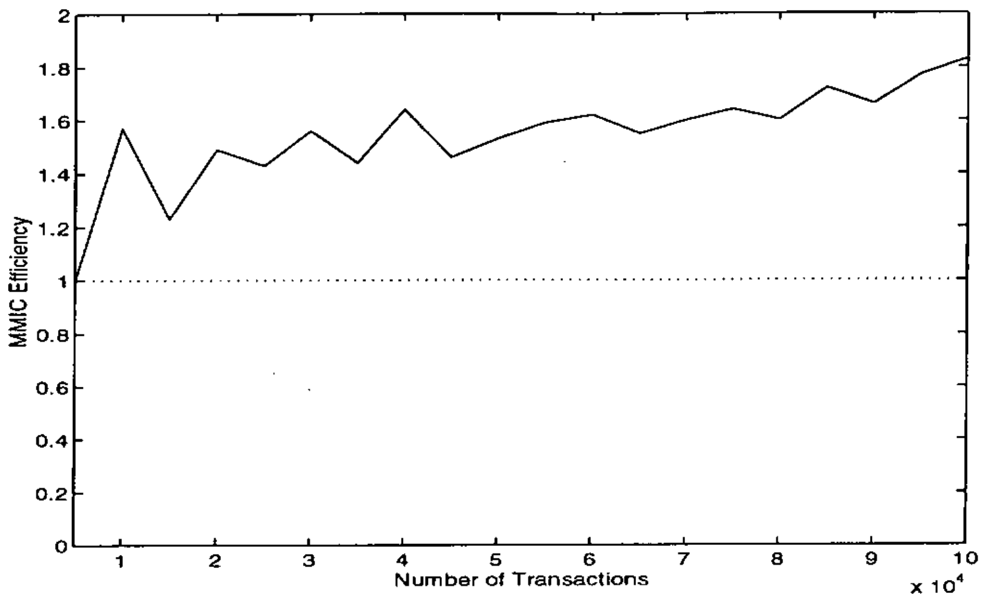


Figure 6.9: MMIC performance over Apriori

- user input constraint ($I_1 + I_2 * I_3 - I_4 \div I_5 \geq 100$)
- minimum support: 5%

The result showed that MMIC brings more efficency than Apriori in mining taxonomy data especially when the depth of the TR is getting larger. The result is expected since MMIC made use of different techniques such as the EGMS, IET and support count propogation.

Chapter 7

Conclusion

In this dissertation, we have presented three mining algorithms to extract subtle and embedded knowledge in a database satisfying the inequality constraint. We consider three types of constraints for different data item relationships. 1) Inequality constraints which consider the quantitative relationship. 2) Temporal constraints which additionally consider the temporal relationship. 3) Taxonomy constraints which consider the multi-layer relationship. The arithmetic inequalities constraints we consider are composed of common operators $(+, -, *, \div)$ and can be easily extended to other queries such as max(),min() and avg().

In our work, we improve the mining efficency by achieve the following two objectives.

- reduce the number of database scans for support counting.
- reduce the number of candidate itemsets in each iteration.

The techniques GMS and EGMS are developed in QMIC and MMIC respectively to achieve the first objective. The basic idea is to skip those unnecessary database scanning and define a new generating sequence. The Max_Min pruning in QMIC, the new candidate itemset generation in OPM and the IET structure in MMIC are the techniques which define new criteria for itemset pruning. Those techniques improve the pruning efficiency by further reducing the number of candidate itemsets. As a conclusion, these proposed algorithms have been verified by experiments that there are improvements in mining effeciency and finding different data item relationships.

7.1 Further Work

Here, we propose some possible further work:

1. **Dynamic Inequality Expression Tree** In MMIC algorithm, candidate itemsets are pruned away when their value are out of the Acceptance Range(AR). The ARs are calculated and propagated only once after the first database scanning then keep as constant in the whole mining process. To further improve the pruning efficiency, the AR can be adjusted in each iteration. It is because some of the itemsets, the AR calculation based on, have been pruned in last iterations. A shorter AR' will be more effective pruning criteria. Since ARs are changed in each iteration and also does the IET. We call this the dynamic inequality tree.

2. **Cross Level Association** In our work, we proposed the effective algorithms for mining the item association in same level. For the taxonomy data, it will be also meaningful for mining the cross level association. A typical example can be the association of milk and Pepsi coke. In order to find such an association, the information passing in IET

should not be only up propagation. Indeed, we need a new structure for the IET. This can be achieved by further extending the MMIC algorithm.

3. **Advanced User Constraint** As we mentioned in our work, not all the the associations are interested. Although the current inequality constraint represents the most common user query, QMIC and MMIC can be further extended to handle more specified input constraint such as *max()*, *min()* and *avg()*.

4. **Further reduce the number of database scanning**. The repeated database scans is the most costly operation in the mining process. The 0proposed GMS and EGMS technique can be further developed to reduce the number of database scanning as lowest as possible.

Bibliography

[1] R. Agrawal and R. Srikant, *Fast Algorithm for Mining Association Rules in Large Databases*, In Proc. VLDB'94, Santiago, Chile, pp. 487-499, September 1994.

[2] R. Agrawal and R. Srikant, *Mining Sequential Patterns,* In Proc. Int. Conf. on Data Engineering, pp. 3-14, 1995.

[3] R. Agrawal, T. Imielinski and A. Swami, *Mining Association Rules between Sets of Items in Large Databases*, In Proc. ACM-SIGMOD Int. Conf. on Management of Data, pp. 207-216, 1993.

[4] RJ Bayardo Jr., R. Agrawal and D. Gunopulos, *Constraint-Based Rule in Large, Dense Database*, In Proc. of the 15th Int. Conf. on Data Engineering, pp. 188-197, 1999.

[5] Roberto J. Bayardo Jr. *Efficiently Mining Long Patterns from Databases*, In Proc. of the ACM-SIGMOD, 27(2), pp. 85-90, 1998.

[6] J. Borges and M. Levene, *Mining Association Rules in Hypertext Database*, American Association for Artificial Intelligence, Knowledge Discovery and Data Mining (KDD),

pp. 149-153, 1998.

[7] Ming-Syan Chen, Jong Soo Oark, Philip S.Yu. *Data Mining for path Traversal Patterns in a Web Environment*, In Proc.16th ICDCS, IEEE, Hong Kong, pp. 385-393, May 1996.

[8] David W. Cheung, JW Han, Vincent TY Ng and CY Wong, *Maintenance of Discovered Association Rules in Large Databases: An Incremental Updating Technique*, In Proc. Int. Conf. on Data Engineering, New Orleans, USA, pp. 106-114, 1996.

[9] L. Feng, HJ Lu, Jeffrey X. Yu and JW Han, *Mining Inter-Transaction Associations with Templates* In Proc. of the 8th Int. Conf. on CIKM-99, ACM Press, pp. 225-233, November 2000.

[10] JW Han and YJ Fu, *Discovery of Multiple-Level Association Rules from Large Databases*, In Proc. of the 21st VLDB-95 Conf., Zurich, Swizerland, pp. 420-431, September 1995.

[11] JW Han, J. Pei, and YW Yin, *Mining Frequent Patterns without Candidate Generation*, ACM-SIGMOD Int. Conf. Management of Data, 29(2), pp. 1-8, 2000.

[12] M. Houstma and A. Swami, *Set-Oriented Mining of Association Rules*, Technical Report RJ 8567. IBM Almaden Research Lab., San Jose, CA, October 1993.

[13] J. Kahng, WH Kevin Liao and Dennis Mcleod, *Mining Generalized Associations: Count Propagation Algorithm*, American Association for Artifical Intelligence, pp. 203-206, 1997.

[14] M. Klemettinen, H. Mannila, P. Ronkainen, H. Toivonen and AI Verkamo, *Finding Interesting Rules from Large Sets of Discovered Association Rules,* In Proc. 3rd Int. Conf. on Information and Knowledge Management, Gaithersburg, Maryland, November, pp. 401-408, 1994.

[15] Charles C Lo, Vincent TY Ng, *Discovering Web Access Orders with Association Rules*, Proc. the IEEE SMC'99 Conference, Vol. IV, pp. 99-104.

[16] Charles C Lo, Vincent TY Ng, *Mining Quantitative Association Rules under Inequality Constraints* Proc. the IEEE KDEX'99 Conference, pp. 53-59.

[17] Vincent TY Ng, D Cheung, Charles C Lo, Efficent Mining of Association Rules under Inequality Constraints in Distributed Database. In book Advances in Distributed and Parallel Knowledge Discovery, published by AAAI, pp. 211-230.

[18] H. Mannila, H. Toivonen and AI Verkamo, *Discovering Frequent Episodes in Sequences,* the 1st Int. Conf. on Knowledge Discovery and Data Mining (KDD-95), Montreal, Canada, AAAI Press, pp. 210-215, August 1995.

[19] RJ Miller and Y. Yang, *Association Rules over Interval Data,* In Proc. ACM-SIGMOD, pp. 452-461, 1997.

[20] R. Ng, L. Lakshmanan, JW Han and A. Pang, *Exploratory Mining and Pruning Optimizations of Constrained Association Rules,* In Proc. ACM-SIGMOD, 27(2), ACM Press, pp. 13-24, 1998.

[21] J. Pei, JW Han, B. Mortazavi-Asl, and H. Zhu, *Mining Access Patterns efficiently from Web logs*, In Proc. 2000 Pacific-Asia Conf. on Knowledge Discovery and Data Mining (PAKDD'00), Kyoto, Japan, April 2000.

[22] J. Pei and JW Han, *Can We Push More Constraints into Frequent Pattern Mining?*, In Proc. Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'2000), (335), Boston, MA, August 2000.

[23] R. Rastogi and K Shim, *Mining Optimized Association Rules with Categorical and Numeric Attributes*, IEEE, pp. 503-512, 1998.

[24] Cyrus Shahabi, Amir M.Zarkesh, Jafar Adibi, Vishal Shah. *Knowledge Discovery from User's Web-Page Navigation*, In IEEE Proc. of the 7th Int. Workshop on RIDE-97, pp. 20-31, April 1997.

[25] R. Srikant and R. Agrawal, *Mining Quantitative Association Rules in Large Relational Tables*, In Proc. ACM-SIGMOD, 25(2), ACM Press, pp. 1-12, June 1996.

[26] R. Srikant, Q. Vu and R Agrawal, *Mining Association Rules with Item Constraints*, KDD'97, pp. 67-73, 1997.

[27] Anthony KH Tung, HJ Lu, JW Han and L. Feng, *Breaking the Barrier of Transactions: Mining Inter-Transaction Association Rules* In Proc. of the 5th ACM-SIGKDD, ACM Press, pp. 297-301, August 1999.

[28] AM Zarkesh, J. Adibi, C. Shahabi, R. Sadri and V. Shan, *Analysis and Design of Server Informative WWW-sites,* ACM 6th Int. Conf. on Information and Knowledge Management, Las Vegas, Nevada, pp. 254-261, November 1997.

[29] T. Zhang, R. Ramakrishnan and M. Livny, *BIRCH: An Efficient Data Clustering Method for Very Large Databases,* In Proc. of the ACM-SIGMOD, Montreal, Canada, ACM Press, pp. 103-114, 1996.