

GRAPH BASED APPROACHES USED IN ASSOCIATION RULE MINING

Thesis submitted in partial fulfillment of the requirements for the award
of degree of

**Master of Engineering
in
Computer Science and Engineering**

**Hemant Kumar Sharma
(800832005)**

Under the supervision of:
**Dr. Deepak Garg
Assistant Professor, CSED**



**COMPUTER SCIENCE AND ENGINEERING DEPARTMENT
THAPAR UNIVERSITY
PATIALA – 147004**

JUNE - 2010

Certificate

I hereby certify that the work which is being presented in the thesis entitled, "**Graph Based Approaches used in Association Rule Mining**", in partial fulfillment of the requirements for the award of degree of Master of Engineering in Computer Science and Engineering submitted in Computer Science and Engineering Department of Thapar University, Patiala, is an authentic record of my own work carried out under the supervision of Dr. Deepak Garg and refers other researcher's works which are duly listed in the reference section.

The matter presented in this thesis has not been submitted for the award of any other degree of this or any other university.



(Hemant Kumar Sharma)

This is to certify that the above statement made by the candidate is correct and true to the best of my knowledge.



(Dr. Deepak Garg)

Assistant Professor

Computer Science and Engineering Department

Countersigned by



(Dr. Rajesh Bhatia)

Head

Computer Science and Engineering, Department
Thapar University, Patiala



(Dr. R. K. Sharma)

Dean (Academic Affairs)

Thapar University,
Patiala

Acknowledgement

No volume of words is enough to express my gratitude towards my guide **Dr. Deepak Garg**, Department of Computer Science and Engineering, Thapar University, Patiala. He has helped me to explore this vast topic in an organized manner and provided me with all the ideas on how to work towards a research-oriented venture.

I am also thankful to **Dr. Rajesh Bhatia**, Head of Department, Computer Science and Engineering Department and **Dr. Inderveer Chana**, P.G. Coordinator, for the motivation and inspiration that triggered me for the thesis work.

I would also like to thank the staff members and my colleagues who were always there at the need of the hour and provided with all the help and facilities.

Most importantly, I would like to thank my parents and the almighty for showing me the right direction out of the blue, to help me stay calm in the oddest of the times and keep moving even at times when there was no hope.



Hemant Kumar Sharma
(800832005)

The Thesis aims at in depth study of Graph based approach to find Association Rule Mining and Association Rule Mining without pre-assign weights and its implementation. In the past, several authors have proposed various association rule mining algorithms with and without pre-assigned weights. The Thesis consists of two parts. Part two consists of graph based association rule mining for primitive as well as items which have concept hierarchy. Complete implementation has been done in C++ on various real life datasets. Part one consisted of association rule mining without pre-assigned weights using HITS algorithm.

Contents	Page No.
Certification	i
Acknowledgement	ii
Abstract	iii
Contents	iv
List of figures	vi
List of Tables	vii
 Chapter 1 Introduction	 1
 Chapter 2 Association Rule Mining	 3
2.1 Overview	3
2.2 Notation and Basic Concepts	3
2.2.1 Definition of Support	4
2.2.2 Definition of Confidence	4
2.3 Market-Basket Analysis	4
2.3.1 Goals for Market-Basket Mining	5
2.3.1.1 Association Rules	5
2.3.1.2 Causality	5
2.3.1.3 Frequent Itemsets	6
2.4 Algorithm for Finding Frequent Itemsets	7
2.4.1 Apriori Algorithm	7
2.4.2 Apriori Candidate Generation and Pruning Function	8
2.5 Recent Advances in Association Rule Mining	9
2.5.1 Redundant Association Rules	9
2.5.2 Other Measures as Interestingness of an Association	10
	iv

2.5.3 Negative Association Rules	11
Chapter 3 Graphs Based Approach for Finding Frequent Itemsets	13
3.1 Primitive Association Rule Mining	15
3.1.1 Association Graph Construction	15
3.1.2 Primitive Association Pattern Generation	16
3.2 Generalized Association Rule Mining	20
3.2.1 Generalized Association Graph Construction	21
3.2.2 Generalized Primitive Association Pattern Generation	22
3.3 Comparison	27
Chapter 4 Link Analysis and Hits Algorithm	28
4.1 Link Analysis Ranking	28
4.2 Hits Algorithm	30
Chapter 5 Association Rule Mining without Pre-Assigned Weights	33
5.1 W-Support: A New Measurement Pre-Assigned Weight	34
5.2 Algorithm of Mining Significant Itemsets without Pre-Assigned Weight	35
5.3 Comparison	41
Chapter 6 Implementations and Results	42
Chapter 7 Conclusion and Future Scope	52
7.1 Conclusion	52
7.2 Future Scope	52
References	54

List of Figures

<u>Figure no.</u>	<u>Title</u>	<u>Page no.</u>
3.1	Concept Hierarchy of a Computer	13
3.2	Concept Hierarchy of Access	14
3.3	Association Graph of Table 3.1	19
3.4	Graph of Generalized Association Rule Mining	26
5.1	Bipartite Representation of a Database	34
6.1	Taxonomy of 20 Items	50

List of Tables

<u>Table no.</u>	<u>Title</u>	<u>Page no.</u>
3.1	Sample Database	18
3.2	Bit-Vector for Sample Database	18
3.3	Market Basket Data	23
3.4	Number Assignment with Bit-Vector	24
3.5	Transaction Items as Number	24
5.1	Data Representation	36

Chapter 1

INTRODUCTION

We are in an age often referred to as the information age. In this information age, because we believe that information leads to power and success, and thanks to sophisticated technologies such as computers, satellites, etc., we have been collecting tremendous amounts of information like business transactions, scientific data, medical data, satellite data, surveillance video & pictures, world wide web repositories to name a few. With the enormous amount of data stored in files, databases, and other repositories, it is increasingly important, if not necessary, to develop powerful means for analysis and perhaps interpretation of such data and for the extraction of interesting knowledge that could help in decision-making.

Data Mining, also popularly known as *Knowledge Discovery in Databases* (KDD), refers to the nontrivial extraction of implicit, previously unknown and potentially useful information from data in databases. While data mining and knowledge discovery in databases (or KDD) are frequently treated as synonyms, data mining is actually part of the knowledge discovery process.

The kinds of patterns that can be discovered depend upon the data mining tasks employed. By and large, there are two types of data mining tasks: *descriptive data mining* tasks that describe the general properties of the existing data, and *predictive data mining* tasks that attempt to do predictions based on inference on available data.

One of the popular descriptive data mining techniques is Association rule mining (ARM), owing to its extensive use in marketing and retail communities in addition to many other diverse fields. Mining association rules is particularly useful for discovering relationships among items from large databases.

Association rule mining deals with market basket database analysis for finding frequent itemsets and generate valid and important rules. Various association rules mining algorithms have been proposed in 1993 by Aggrawal et. al. [1,2] viz. Apriori, Apriori-TID and Apriori Hybrid. Other algorithms for finding frequent itemsets include pincer search [3], FP (frequent pattern) tree [4]. Apriori- generation function follows bottom- up approach. Pincer search algorithm o finds frequent itemsets but it follows both bottom-

up and top-down approach. Frequent pattern tree also generate frequent itemsets without candidate generation.

In all algorithms discussed above, all items and transaction have equal importance. However in practice this is not true. Given a large set of market basket transaction with number of items, different items might have different weights or importance, and also different transactions may have different weights. This is termed as in weighted association rule mining (WARM) [5]. In 2008, Sun et. al [6] proposed yet another algorithm for finding frequent itemsets without preassigned weights. The advantage of this approach is that weights for items and transactions can be found from data itself rather than preassigning weights.

Yen et. al [7] proposed a Graph-Based Approach for Discovering Various Types of Association Rules in which we can find association rule mining using graph, and it gives better result as compared to Apriori algorithms. The graph based approach has been extended to Generalized Association Rule Mining which is better than cumulate algorithm [8].

Graph based association rule mining uses bit vector data structure for storing datasets, which is better than any other approach to store datasets. In the second part of the Thesis, two Graph based approaches have been considered. First is Primitive association rule mining and other is generalized association rule mining. In both the approaches, graphs are constructed and frequent itemsets are found from graphs.

2.1 Overview

Association rule mining (Aggarwal et.al. [1], 1993) is one of the important problems of data mining. The goal of the Association rule mining is to detect relationships or associations between specific values of categorical variables in large data sets. This is a common task in many data mining projects. Suppose I is a set of items, D is a set of transactions, an association rule is an implication of the form $X \Rightarrow Y$, where X, Y are subsets of I , and X, Y do not intersect. Each rule has two measures, *support* and *confidence*. Association rule mining was originally proposed in the domain of market basket data. The association rule mining on Market "Basket Data" is Boolean Association Rule Mining in which only Boolean attributes are considered. In order to do association rule mining on quantitative data, such as Remotely Sensing Image data, some mapping should be done from quantitative data to Boolean data. The main idea here is to partition the attribute values into *Transaction Patterns*. Basically, this technique enables analysts and researchers to uncover hidden patterns in large data sets.

2.2 Notation and Basic Concepts

Let $\Omega = \{i_1, i_2 \dots i_m\}$ be a universe of items. Also, let $T = \{t_1, t_2 \dots t_n\}$ be a set of all transactions collected over a given period of time. To simplify a problem, we will assume that every item i can be purchased only once in any given transaction t . Thus $t \subseteq \Omega$ ("t is a subset of omega"). In reality, each transaction t is assigned a number, for example a transaction id (TID). Let now A be a set of items (or an itemset). A transaction t is said to contain A if and only if $A \subseteq t$. Now, mathematically, an association rule will be an implication of the form

$$A \Rightarrow B$$

Where both A and B are subsets of Ω and $A \cap B = \phi$ ("the intersection of sets A and B is an empty set").

2.2.1 Support

The *support* of an itemset is the fraction of the rows of the database that contain all of the items in the itemset. Support indicates the frequencies of the occurring patterns. Sometimes it is called *frequency*. Support is simply a probability that a randomly chosen transaction t contains both itemsets A and B . Mathematically,

$$\text{Support}(A \Rightarrow B) = P(A \subseteq t \wedge B \subseteq t)$$

We will use a simplified notation that

$$\text{Support}(A \Rightarrow B) = P(A \wedge B)$$

2.2.2 Confidence

Confidence denotes the strength of implication in the rule. Sometimes it is called *accuracy*. Confidence is simply a probability that an itemset B is purchased in a randomly chosen transaction t given that the itemset A is purchased. Mathematically,

$$\text{Confidence}(A \Rightarrow B) = P(B \subseteq t | A \subseteq t)$$

We will use a simplified notation that

$$\text{Confidence}(A \Rightarrow B) = P(B | A)$$

In general, a set of items (such as the antecedent or the consequent of a rule) is called an itemset. The number of items in an itemset is called the length of an itemset. Itemsets of some length k are referred to as k -itemsets. Generally, an association rules mining algorithm contains the following steps:

- The set of candidate k -itemsets is generated by 1-extensions of the large $(k - 1)$ - itemsets generated in the previous iteration.
- Supports for the candidate k -itemsets are generated by a pass over the database.
- Itemsets that do not have the minimum support are discarded and the remaining itemsets are called large k -itemsets.

This process is repeated until no more large itemsets are found[13].

2.3 Market-Basket Analysis

The market-basket problem assumes we have some large number of items, e.g., bread, milk. Customers fill their market baskets with some subset of the items, and we get to know what items people buy together, Even if we don't know who they are. Marketers use this information to position items, and control the way a typical customer traverses the store.

In addition to the marketing application, the same sort of question has the following uses:

1. Baskets = documents; items = words. Words appearing frequently together in documents may represent phrases or linked concepts and can be used for intelligence gathering.
2. Baskets = sentences, items = documents. Two documents with many of the same sentences could represent plagiarism or mirror sites on the Web.

2.3.1 Goals for Market-Basket Mining

2.3.1.1 Association rules:

We can define association rule as follows:

Let $I = \{i_1, i_2, \dots, i_m\}$ be a set of literals or items, $D = \{t_1, t_2, \dots, t_n\}$ be a set of transactions, where each transaction t_i is an itemset such that $t_i \subseteq I$. Each transaction, t , has a transaction-id ($t.id$) and an itemset ($t.Itemset$), i.e., $t = (t.id, t.Itemset)$. A transaction t contains an itemset X if X is a subset of $t.Itemset$. An Association rule, R , denoted by $R: X \rightarrow Y$, where X and Y are itemsets that don't intersect. Each rule R has two value measures, support and confidence, denoted by $sup(R)$ and $conf(R)$ respectively. The support of an item set, X , has support, s , in transaction set, D , if $s\%$ of transaction in D contain X . Then, $sup(R: X \rightarrow Y) = sup(X \cup Y)$, $conf(R: X \rightarrow Y) = sup(X \cup Y) / sup(X)$. Different transactions may contain same itemset, especially for remote sensed imagery. This suggests a way to eliminate duplicate calculation. Some concepts are given below. Let $U = \{t \mid t \text{ is any possible transaction}\}$, while $D = \{t \mid t \text{ is a transaction already happened}\}$.

2.3.1.2 Causality:

Ideally, we would like to know that in an association rule the presence of $X_1, \dots, X_m$ actually causes Y to be bought. However, "causality" is an elusive concept. Nevertheless, for market-basket data, the following test suggests what causality means. If we lower the price of diapers and raise the price of beer, we can lure diaper buyers, who are more likely to pick up beer while in the store, thus covering our losses on the diapers. That strategy works because diapers cause beer. However, working it the other way round, running a sale on beer and raising the price of diapers, will not result in beer buyers buying diapers in any great numbers, and we lose money.

2.3.1.3 Frequent Itemsets:

In many (but not all) situations, we only care about association rules or causalities involving sets of items that appear frequently in baskets. For example, we cannot run a good marketing strategy involving items that no one buys anyway. Thus, much data mining starts with the assumption that we only care about sets of items with high support; i.e., they appear together in many baskets. We then find association rules or causalities only involving a high-support set of items i.e., $\{X_1, \dots, X_m\}$. Y must appear in at least a certain percent of the baskets, called the support threshold.

The AIS algorithm was the first algorithm proposed for mining association rule [1]. In this algorithm only one item consequent association rules are generated, which means that the consequent of those rules only contain one item, for example we only generate rules like $X \cap Y \rightarrow Z$ but not those rules as $X \rightarrow Y \cap Z$. The main drawback of the AIS algorithm is too many candidate itemsets that finally turned out to be small are generated, which requires more space and wastes much effort that turned out to be useless. At the same time this algorithm requires too many passes over the whole database. Apriori is more efficient during the candidate generation process [2]. Apriori uses pruning techniques to avoid measuring certain itemsets, while guaranteeing completeness. These

are the itemsets that the algorithm can prove will not turn out to be large. However there are two bottlenecks of the Apriori algorithm. One is the complex candidate generation process that uses most of the time, space and memory. Another bottleneck is the multiple scan of the database. Based on Apriori algorithm, many new algorithms were designed with some modifications or improvements [13].

2.4 Algorithm for Finding Frequent Itemsets

1. Given support threshold s , in the first pass we find the items that appear in at least fraction s of the baskets. This set is called L_1 , the frequent items.

2. Pairs of items in L_1 become the candidate pairs C_2 for the second pass. We hope that the size of C_2 is not so large that there is not room for an integer count per candidate pair. The pairs in C_2 whose count reaches s are the frequent pairs, L_2 .

3. The candidate triples, C_3 are those sets $\{A; B; C\}$ such that all of $\{A; B\}$, $\{A; C\}$, and $\{B; C\}$ are in L_2 . On the third pass, count the occurrences of triples in C_3 ; those with a count of at least s are the frequent triples, L_3 .

4. Proceed as far as you like (or the sets become empty). L_i is the frequent sets of size i ; C_{i+1} is the set of sets of size $i + 1$ such that each subset of size i is in L_i .

2.4.1 Apriori Algorithm

Input: Transaction T that contains all transaction with itemsets in each transaction

Output: All frequent Itemsets that satisfy minimum threshold condition

Start

Initialize L_1 =large 1-itemsets that contains all items in beginning i.e. $L_1 = I$ (I is item sets)

$K=2$

```

While (  $L_{k-1} \neq \emptyset$  )
{
     $C_k$  = Apriori-gen ( $L_{k-1}$ ) //  $C_k$  contains K-itemsets using Apriori-gen function
    For all transactions  $t \in D$ 
    {
         $C_t$  = subset ( $C_k, t$ ) //check whether  $C_k$  belongs to Transaction  $t$ 
        For all candidates  $c \in C_t$ 
        {
             $C.count++$ 
        }
    }
    Add  $L_k$  to result list
     $L_k = \{c \in C_k \mid c.count \geq \text{minsup}\}$ 
    Increment  $k$  by 1
}
End

```

2.4.2 Apriori Candidate Generation and Pruning Function

The Apriori-gen function takes an argument L_{k-1} , the set of all large $(k - 1)$ -itemsets. It returns a superset of the set of all large k -itemsets. The function works as follows. First, in the join step, we join L_{k-1} with L_{k-1} :

```

Insert into  $C_k$ 
Select  $p.item_1, p.item_2, \dots, p.item_{k-1}, q.item_k$ 
From  $L_{k-1} p, L_{k-1} q$ 
Where  $p.item_1 = q.item_1, \dots, p.item_{k-2} = q.item_{k-2},$ 
 $p.item_{k-1} < q.item_k$ ;

```

Here, C_k is candidate k -itemsets, L_{k-1} is large $k-1$ itemsets.

Similarly, SQL statement of candidate generation is

```

SELECT  $b_1.item, b_2.item, COUNT (*)$ 
FROM  $L b_1, L b_2$ 
WHERE  $b_1.BID = b_2.BID$  AND  $b_1.item < b_2.item$ 
GROUP BY  $b_1.item, b_2.item$ 

```

HAVING COUNT (*) $\geq$ s;

Output of this query is candidate k-itemsets. For this query L is large k-1 itemsets, s is threshold.

After generation of candidate k itemsets then we have to prune itemsets from C_k which does not belongs to L_{k-1} itemsets for each k-1 subset of C_k .

For all itemsets $c \in C_k$

{

For all (k-1)-subsets s of c

{

If (s does not belongs L_{k-1}) then delete c from C_k ;

}

}

2.5 Recent advances in association rule discovery

A serious problem in association rule discovery is that the set of association rules can grow to be unwieldy as the number of transactions increases, especially if the support and confidence thresholds are small. As the number of frequent itemsets increases, the number of rules presented to the user typically increases proportionately. Many of these rules may be redundant.

2.5.1 Redundant Association Rules

To address the problem of rule redundancy, four types of research on mining association rules have been performed. First, rules have been extracted based on user-defined templates or item constraints [24]. Secondly, researchers have developed interestingness measures to select only interesting rules. Thirdly, researchers have proposed inference

*Referenced from infolab.stanford.edu/~ullman/mining/allnotes.pdf

rules or inference systems to prune redundant rules and thus present smaller, and usually more understandable sets of association rules to the user [14]. Finally, new frameworks for mining association rule have been proposed that find association rules with different formats or properties [15]. Ashrafi et al [16] presented several methods to eliminate redundant rules and to produce small number of rules from any given frequent or frequent closed itemsets generated. Ashrafi et al [17] present additional redundant rule elimination methods that first identify the rules that have similar meaning and then eliminate those rules. Furthermore, their methods eliminate redundant rules in such a way that they never drop any higher confidence or interesting rules from the resultant rule set. Moreover, there is a need for human intervention in mining interesting association rules. Such intervention is most effective if the human analyst has a robust visualization tool for mining and visualizing association rules. Techapichetvanich and Datta [18] presented a three-step visualization method for mining market basket association rules. These steps include discovering frequent itemsets, mining association rules and finally visualizing the mined association rules.

2.5.2 Other measures as interestingness of an association

Omiecinski [19] concentrates on finding associations, but with a different slant. That is, he takes a different view of significance. Instead of support, he considers other measures, which he calls all-confidence, and bond. All these measures are indicators of the degree to which items in an association are related to each other. With all-confidence, an association is deemed interesting if all rules that can be produced from that association have a confidence greater than or equal to a minimum all-confidence value. Bond is another measure of the interestingness of an association. With regard to data mining, it is similar to support but with respect to a subset of the data rather than the entire data set. The idea is to find all itemsets that are frequent in a set of user-defined time intervals. In this case, the characteristics of the data define the subsets not the end-user. Omiecinski [19] proved that if associations have a minimum all-confidence or minimum bond, then those associations will have a given lower bound on their minimum support and the rules produced from those associations will have a given lower bound on their minimum confidence as well. The performance results showed that the algorithm can find large

itemsets efficiently. In [15], the authors mine association rules that identify correlations and consider both the absence and presence of items as a basis for generating the rules. The measure of significance of associations that is used is the chi-squared test for correlation from classical statistics. In [20], the authors still use support as part of their measure of interest of an association. However, when rules are generated, instead of using confidence, they use a metric that they call conviction, which is a measure of implication and not just co-occurrence. In [21], the authors present an approach to the rare item problem. The dilemma that arises in the rare item problem is that searching for rules that involve infrequent (i.e., rare) items requires a low support but using a low support will typically generate many rules that are of no interest. Using a high support typically reduces the number of rules mined but will eliminate the rules with rare items. The authors attack this problem by allowing users to specify different minimum supports for the various items in their mining algorithm.

2.5.3 Negative Association Rules

Typical association rules consider only items enumerated in transactions. Such rules are referred to as positive association rules. Negative association rules also consider the same items, but in addition consider negated items (i.e. absent from transactions). Negative association rules are useful in market-basket analysis to identify products that conflict with each other or products that complement each other. Mining negative association rules is a difficult task, due to the fact that there are essential differences between positive and negative association rule mining. The researchers attack two key problems in negative association rule mining: (i) how to effectively search for interesting itemsets, and (ii) how to effectively identify negative association rules of interest. Brin et. al [15] mentioned for the first time in the literature the notion of negative relationships. Their model is chi-square based. They use the statistical test to verify the independence between two variables. To determine the nature (positive or negative) of the relationship, a correlation metric was used. In [22] the authors present a new idea to mine strong negative rules. They combine positive frequent itemsets with domain knowledge in the form of taxonomy to mine negative associations. However, their algorithm is hard to generalize since it is domain dependant and requires a pre-defined taxonomy. Wu et al

[23] derived a new algorithm for generating both positive and negative association rules. They add on top of the support-confidence framework another measure called *mininterest* for a better pruning of the frequent itemsets generated.

Association rule mining has a wide range of applicability such market basket analysis, medical diagnosis/ research, Website navigation analysis, homeland security and so on. The conventional algorithm of association rules discovery proceeds in two steps. All frequent itemsets are found in the first step. The frequent itemset is the itemset that is included in at least *minsup* transactions. The association rules with the confidence at least *minconf* are generated in the second step. End users of association rule mining tools encounter several well known problems in practice. First, the algorithms do not always return the results in a reasonable time. It is widely recognized that the set of association rules can rapidly grow to be unwieldy, especially as we lower the frequency requirements. The larger the set of frequent itemsets the more the number of rules presented to the user, many of which are redundant. This is true even for sparse datasets, but for dense datasets it is simply not feasible to mine all possible frequent itemsets, let alone to generate rules, since they typically produce an exponential number of frequent itemsets; finding long itemsets of length 20 or 30 is not uncommon. Although several different strategies have been proposed to tackle efficiency issues, they are not always successful [13].

CHAPTER 3

GRAPH BASED APPROACH FOR FINDING FREQUENT ITEMSETS

This chapter discusses graph-based approach for discovering various types of association rules viz

- Primitive Association Rules.
- Generalized Association Rules.

Yen et. Al [7] has purposed a way to find out association rule mining using graph. For finding association rule mining, the concept of graphs and bit vector model is used.

A primitive association rule is an association rule which describes the association among database items which appear in the database. A primitive association pattern is a large itemset in which each item is a database item. A generalized association rule describes the association among database items as well as items of concept hierarchy (taxonomy). Concept hierarchy of the items can usually be derived. An example of concept hierarchy is shown in Fig. 3.1 and 3.2, in which the terminal nodes are database items, and the non-terminal nodes are generalized items.

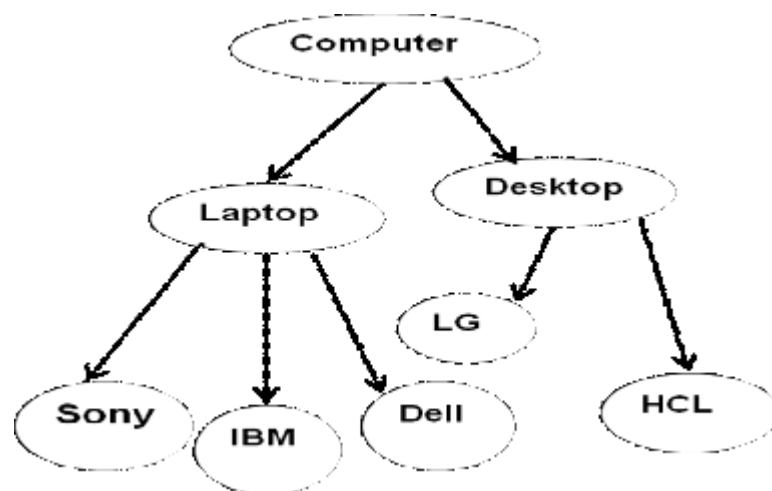


Fig: 3.1

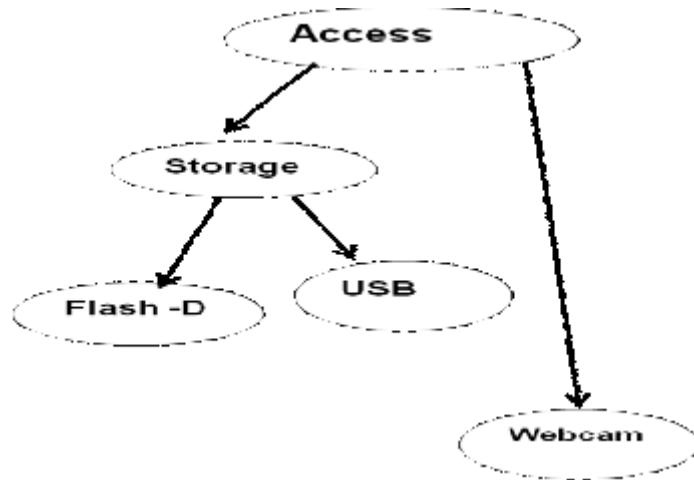


Fig: 3.2

In concept hierarchy, if there is a path between nodes y and x , where y is a “higher concept” of x , then y is called an ancestor of x and x a descendant of y . Here “higher concept” means more general items are created in concept hierarchy. Support of non-leaf item is the number of 1’s in bit-vector of that item. I.e if x is leaf node in concept hierarchy then support of item x is number of 1’s in bit-vector. Otherwise use logical OR operation over bit-vectors of children. Association rules may exist at higher level concepts if the itemsets at the lower level concepts cannot reach the minimum support. Hence, significant association rules may not be discovered if we only consider database items which are the lowest level concepts in the concept hierarchy. A generalized association rule describes the association among items which can be generalized items or database items. Here we are creating concept hierarchy after that we can find frequent large itemsets which is more general among the items. In traditional approaches only transactions and items are used for finding frequent itemsets, but using generalized association rule mining we can find frequent itemsets among different concept hierarchy also. This is very important for those types of transactions in which number of items in each transaction is less or most of the items appeared in few transactions. If number of items which appeared less in transaction or most of items appeared in few transactions then we can not find frequent itemsets but using concept hierarchy we can find large

number of frequent itemsets and we can find number of rules among different concept hierarchy and we can find large itemsets in which each item is a generalized item or database item.

There are few steps to find frequent itemsets and association rules. These are:

- Numbering phase: In this phase, all items are assigned an integer number.
- Large item generation phase: This phase generates large items and records related information. A large item is an item whose support is no less than a user specified minimum support.
- Association graph construction phase: This phase constructs an association graph to indicate the associations between large items.
- Association pattern generation phase: This phase generates all association patterns by traversing the constructed association graph.
- Association rule generation phase: The association rules can be generated directly according to the corresponding association patterns.

3.1 Primitive Association Rule Mining

In this section, an algorithm PAPG (Primitive Association Pattern Generation) is presented to generate primitive association patterns. Here, I describe the first four phases discussed in previous section for the algorithm PAPG.

3.1.1 Association Graph Construction

In the numbering phase, the algorithm PAPG arbitrarily assigns each item a unique integer number. In the large item generation phase, PAPG scans the database and builds a bit vector for each item. The length of each bit vector is the number of transactions in the database. If an item appears in the i th transaction, the i th bit of the bit vector associated with this item is set to 1. Otherwise, the i th bit of the bit vector is set to 0. The bit vector associated with item i is denoted as BV_i . The number of 1s in BV_i is equal to the number of transactions which support the item i , that is, the support for the item i .

Property 1. The support for the itemset $\{i_1; i_2; \dots; i_k\}$ is the number of 1s in $BVi_1 \wedge BVi_2 \wedge \dots \wedge BVik$, where the notation $\wedge$ is a logical AND operation.

In the association graph construction phase, PAPG applies the algorithm AGC (Association Graph Construction) to construct association graph. The AGC algorithm is described as follows: For every two large items i and j , $i < j$, if the number of 1s in $BVi \wedge BVj$ achieves the user-specified minimum support, a directed edge from item i to item j is created. Also, itemset $(i; j)$ is a large 2-itemset.

3.1.2 Primitive Association Pattern Generation

The large 2-itemsets are generated after the association graph construction phase. In the association pattern generation phase, the algorithm LGDE (Large itemset Generation by Direct Extension) is proposed to generate large k -itemsets ($k > 2$), which is described as follows:

For each large k -itemsets ($k \geq 2$), the last item of the k -itemset is used to extend the large itemset into $k+1$ -itemsets.

Lemma 1. If an itemset is not a large itemset, then any itemset which contains this itemset cannot be a large itemset.

Lemma 2. For a large itemset $(i_1; i_2; \dots; i_k)$, if there is no directed edge from item i_k to an item v , then itemset $(i_1; i_2; \dots; i_k; v)$ cannot be a large itemset.

Suppose $(i_1; i_2; \dots; i_k)$ is a large k -itemset. If there is no directed edge from item i_k to an item v , then the itemset need not be extended into $k+1$ -itemset, because $(i_1; i_2; \dots; i_k; v)$ must not be a large itemset according to Lemma 2. If there is a directed edge from item i_k to an item u , then the itemset $(i_1; i_2; \dots; i_k)$ is extended into $k+1$ -itemset $(i_1; i_2; \dots; i_k; u)$. The itemset $(i_1; i_2; \dots; i_k; u)$ is a large $k+1$ -itemset if the number of 1s in $BVi_1 \wedge BVi_2 \wedge \dots \wedge BVik \wedge BVu$ achieves the minimum support. If no large $k+1$ -itemsets can be generated, the algorithm LGDE terminates.

Algorithm of Primitive Association Rule Mining

Input: Transaction T that contains all transaction with itemsets in each transaction

Output: All frequent Itemsets that satisfy minimum threshold condition

Start

Assign numbers to each item

For each item i, Convert transaction table into bit vector

For each item i

{

If (number of 1's achieve minimum support condition of item i)

{

Add this item in large 1-itemsets

}

}

Generate candidate 2-itemsets using large 1- itemsets

For each candidate 2-itemsets (i, j)

{

If(number of 1's is achieve minimum support condition after logical AND operation of item I and j)

{

Add it to large 2-itemsets and create an edge between vertex i to j

}

}

For each large k-itemsets (i1; i2; . . . ; ik)

{

If (there is an edge item ik to any vertex (item) v and number of 1's is achieve minimum support condition after logical AND operation of items i1, i2,..., ik and v)

{

Add (i1; i2; . . . ; ik, v) as large k+1 itemsets.

}

}

Stop

Example 1:- Consider the database in Table 3.1. Each record is a <TID, Itemset> pair, where TID is the identifier of the corresponding transaction, and Itemset records the items purchased in the transaction. Assume that the minimum

TID	ITEMSETS
100	ABC
200	ABCD
300	ABCDE
400	BD
500	ACF
600	CDE

TABLE 3.1

ITEMSETS (ITEM NO.)	BIT-VECTOR(BV)
A(1)	111010
B(2)	111100
C(3)	111011
D(4)	011101
E(5)	001001
F(6)	000010

TABLE 3.2 (Bit vector for corresponding items)

Support is 50 percent (i.e., 3 transactions). After the numbering phase, the numbers of the items A, B, C, D, E and F are 1, 2, 3, 4, 5 and 6, respectively. In the large item generation phase, the large items found in the database are items 1, 2, 3 and 4, and BV1, BV2, BV3, BV4, BV5 and BV6 are 111010, 111100, 111011, 011101, 001001 and 000010, respectively.

For finding large two items use logical AND operation between large 1 itemsets.

$$BV1 \wedge BV2 = 111010 \wedge 111100 = 111000 \text{ (3)}$$

$$BV1 \wedge BV3 = 111010 \wedge 111011 = 111010 \text{ (3)}$$

$$BV1 \wedge BV4 = 111010 \wedge 011101 = 011000 \text{ (2)}$$

$$BV2 \wedge BV3 = 111100 \wedge 111011 = 111000 \text{ (3)}$$

$$BV2 \wedge BV4 = 111100 \wedge 011101 = 011100 \text{ (3)}$$

$$BV3 \wedge BV4 = 111011 \wedge 011101 = 011001 \text{ (3)}$$

So large two items are { (1,2), (1,3), (2,3), (2,4), (3,4)}. And create an edge between these vertices, which is shown in figure 3.3.

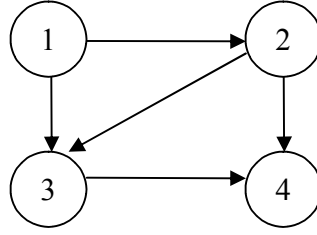


Fig 3.3: Association Graph of Table-3.1

After the association graph construction phase, the large 2-itemsets (1, 2), (1, 3), (2, 3), (2, 4), and (3, 4) are generated. For the large 2-itemset (1, 2), there is a directed edge from the last item 2 of the itemset (1, 2) to items 3 and 4 in the association graph shown in Fig. 2. Hence, the 2-itemset (1, 2) is extended into 3-itemset (1, 2, and 3) and (1, 2, 4). The number of 1s in $BV1 \wedge BV2 \wedge BV3$ ($111010 \wedge 111100 \wedge 111011 = 111000$) is 3. Hence,

The 3-itemset (1, 2, and 3) is a large 3-itemset, since the number of 1s in its bit vector is no less than the minimum support threshold.

The number of 1s in $BV1 \wedge BV2 \wedge BV4$ ($111010 \wedge 111100 \wedge 011101 = 011000$) is 2. Hence, the 3-itemset (1, 2, and 4) is a not large 3-itemset, since the number of 1s in its bit vector is less than the minimum support threshold.

For the large 2-itemset (1, 3), there is a directed edge from the last item 3 of the itemset (1, 3) to items 4 in the association graph shown in Fig. 2. Hence, the 2-itemset (1, 3) is extended into 3-itemset (1, 3, 4). The number of 1s in $BV1 \wedge BV3 \wedge BV4$ ($111010 \wedge 111011 \wedge 011101 = 011010$) is 3. Hence, the 3-itemset (1, 3, and 4) is a large 3-itemset, since the number of 1s in its bit vector is no less than the minimum support threshold.

So large 3-itemsets $\{(1, 2, 3), (1, 3, 4)\}$

For the large 3-itemset (1, 2, 3), there is a directed edge from the last item 3 of the itemset (1, 2, 3) to items 4 in the association graph shown in Fig. 2. Hence, the 3-itemset (1, 2, 3) is extended into 4-itemset (1, 2, 3, 4). The number of 1s in $BV1 \wedge BV2 \wedge BV3 \wedge BV4$ ($111010 \wedge 111100 \wedge 111011 \wedge 011101 = 011000$) is 2. Hence, the 4-itemset (1, 2, 3, and 4) is a no large 3-itemset, since the number of 1s in its bit vector is less than the minimum support threshold.

The LGDE algorithm terminates because no large 5-itemsets can be further generated.

3.2 Mining Generalized Association Rules

The algorithm GAPG (Generalized Association Pattern Generation) use to discover all generalized association patterns. In the following, we also describe the four phases for the algorithm GAPG.

To generate generalized association patterns, one can add all ancestors of each item in a transaction to the transaction and then apply the algorithm PAPG on the extended transactions. However, because if an item is a large item, then the 2-itemset which contains the item and its ancestor is also a large 2-itemset, the number of the edges in the association graph can be very large, and the LGDE algorithm needs to take much more time to traverse the association graph to generate all large itemsets.

In the numbering phase, GAPG applies the numbering method PON (Post order Numbering method) to number items at the concept hierarchies. For each concept hierarchy, PON numbers each item according to the following order: For each item at the concept hierarchy, after all descendants of the item are numbered, PON numbers this item immediately, and all items are numbered increasingly. After all items at a concept hierarchy are numbered, PON numbers items at another concept hierarchy.

Lemma 3. If the numbering method PON is adopted to number items, and for every two items i and j ($i < j$), item Θ is an ancestor of item i but not an ancestor of item j , then $\Theta < j$.

Rationale. According to PON numbering method, after all descendants of an item are numbered, this item is numbered immediately, and these items are numbered increasingly. Hence, for an item i , if it is numbered, then its ancestor Θ must be numbered before the other item j which is not a descendant of item Θ is numbered. So, $\Theta < j$.

Large Item Generation

In the large item generation phase, GAPG builds a bit vector for each database item, and finds all large items (include database items and generalized items). Here, we assume that all database items are specific items.

Lemma 4 Suppose items $\{i_1; i_2; \dots; \text{and } i_m\}$ are all specific descendants of the generalized item i_n . The bit vector BV_{i_n} associated with item i_n is $BV_{i_1} \vee BV_{i_2} \vee \dots \vee BV_{i_m}$, and the number of 1s in $BV_{i_1} \vee BV_{i_2} \vee \dots \vee BV_{i_m}$ is the support for item i_n , where the notation " $\vee$ " is a logical OR operation.

From Lemma 4, the bit vector associated with a generalized item is obtained by performing logical OR operations on the bit vectors associated with all specific descendants of the generalized item.

3.2.1 Generalized Association Graph Construction

In the association graph construction phase, GAPG applies the algorithm GAGC (Generalized Association Graph Construction) to construct a generalized association graph to be traversed. The algorithm GAGC is described as follows: For every two large items i and j ($i < j$), if item j is not an ancestor of item i and the Number of 1s in $BV_i \wedge BV_j$ achieves the user-specified minimum support, a directed edge from item i to item j is created. Also, itemset $(i; j)$ is a large 2-itemset.

Lemma 5. If an itemset X is a large itemset, then any itemset generated by replacing an item in itemset X with its ancestor is also a large itemset.

Lemma 6. If $(\text{the number of 1s in } BV_i \wedge BV_j) \geq \text{minimum-support}$, then for each ancestor u of item i and for each ancestor v of item j , $(\text{the number of 1s in } BV_u \wedge BV_j) \geq \text{minimum-support}$ and $(\text{the number of 1s in } BV_i \wedge BV_v) \geq \text{minimum-support}$.

From Lemma 6, if an edge from item i to item j is created, the edges from item i to the ancestors of item j , which are not ancestors of item i , are also created. According to Lemma 3, the numbers of the ancestors of item i , who are not the ancestors of item j , are

all less than j . Hence, if an edge from item i to item j is created, the edges from the ancestors of item i , which are not ancestors of item j , to item j is also created.

3.2.2 Generalized Association Pattern Generation

In the association pattern generation phase, GAPG applies to generate all generalized association patterns by traversing the generalized association graph.

Algorithm of Generalized Association Rule Mining

Input: Taxonomy (concept hierarchy) and Transaction T that contains all transaction with itemsets in each transaction.

Output: All frequent Itemsets that satisfy minimum threshold condition.

Start

Assign numbers in post order for to each item in taxonomy

For each database item i , Convert transaction table into bit vector.

For non leaf items of taxonomy, construct bit vector using logical OR operation of its children

For each item i

{

If (number of 1's achieve minimum support condition of item i)

{

Add this item in large 1-itemsets

}

}

//Generate candidate 2-itemsets using large 1- itemsets

For each candidate 2-itemsets (i, j)

{

If (number of 1's is achieve minimum support condition after logical AND operation of item i and j and item i is not descendent of item j)

{

Add it to large 2-itemsets and create an edge between vertex i to j

}

}

For each large k-itemsets (i1; i2; . . . ; ik)

{

If (there is an edge item ik to any vertex (item) v and number of 1's is achieve minimum support condition after logical AND operation of items i1, i2,..., ik and v and there is parent-child relationship in this item)

{

Add (i1; i2; . . . ; ik, v) as large k+1 itemsets.

}

}

Stop

Example- Consider the database in Table 3.1 and the concept hierarchies in Fig. 3.1, 3.2. Assume that the minimum support = 40 percent.

TID	Items
100	SONY, USB, WEBCAM
200	IBM, WEBCAM
300	DELL, IBM, USB, WEBCAM
400	IBM, LG, FLSH-D
500	IBM, DELL, HCL, USB
600	USB, FLASH-D, LG
700	SONY, HCL, FLASH-D
800	DELL , LG, USB, WEBCAM
900	USB, FLASH-D
1000	HCL, WEBCAM

Table 3.3- Market Basket Data

Item names	Items number	Bit Vector (BV)
SONY	1	1000001000
IBM	2	0111110000
DELL	3	0010100100
LAPTOP	4	1111101100
LG	5	0001010100
HCL	6	0000101001
DESKTOP	7	0001111001
COMPUTER	8	1111111101
FLASH-D	9	0001011010
USB	10	1110110110
STORAGE	11	1111111110
WEBCAM	12	1010000101
ACCESS	13	1111111111

Table 3.4- Assigning No. and Bit Vector

TID	Items
100	1, 10, 12
200	2, 10
300	2, 3, 10, 12
400	2, 5, 9
500	2, 3, 6, 10
600	5, 9, 10
700	1, 6, 9
800	3, 5, 10, 12
900	9, 10
1000	6, 12

Table 3.5- Transaction items as Number

Large 1-itemsets are those which achieve minimum number of 1's in BV_i (bit vector of item i). So large 1-itemsets are {2, 4, 7, 8, 9, 10, 11, 12, 13}.

For generating large 2-itemsets, use bitwise AND operation between large 1-itemsets.

And also do not use AND operation between those item which has parent child relation.

So these are logical AND operation between large 1-itemsets:

BV2^BV7= 0111110000^ 0001111001=0001100000
BV2^BV9= 0111110000 ^ 0001011010 =0001000000
BV2^BV10= 0111110000^ 1110110110=0110100000
BV2^BV11= 1111111110^ 1111111110=0111100000
BV2^BV12= 1111111110 1010000101=0010000000
BV2^BV13= 0111110000^ 1111111111=0111100000
BV4^BV7= 1111101100^ 0001111001=0001101000
BV4^BV9= 1111101100^ 0001011010=0001001000
BV4^BV10= 1111101100^ 1110110110=1110100100
BV4^BV11= 1111101100^ 1111111110=1111101100
BV4^BV12= 1111101100^ 1010000101=1010000100
BV4^BV13= 1111101100^ 1111111111=1111101100
BV7^BV9= 0001111001^ 0001011010=0001011000
BV7^BV10= 0001111001^ 1110110110=000011000
BV7^BV11= 0001111001^ 1111111110=0001111000
BV7^BV12= 0001111001^ 1010000101=0000000000
BV7^BV13= 0001111001^ 1111111111=0001111001
BV8^BV9= 1111111101^ 0001011010=0001011000
BV8^BV10= 1111111101^ 1110110110=1110110100
BV8^BV11= 1111111101^ 1111111110=1111111100
BV8^BV12= 1111111101^ 1010000101=1010000101
BV8^BV13= 1111111101^ 1111111111=1111111101
BV9^BV10= 0001011010^ 1110110110=0000010010
BV9^BV12= 0001011010^ 1010000101=0000000000
BV10^BV12= 1110110110^ 1010000101=1010000100
BV11^BV12= 1111111110^ 1010000101=1010000100

If number of 1's in $BV_i \wedge BV_j$ ($i > j$) achieve minimum support and item i is not ancestor of j then draw an edge between vertices i to j .

So figure 3.4 is graph and edges between items are large 2-itemsets.

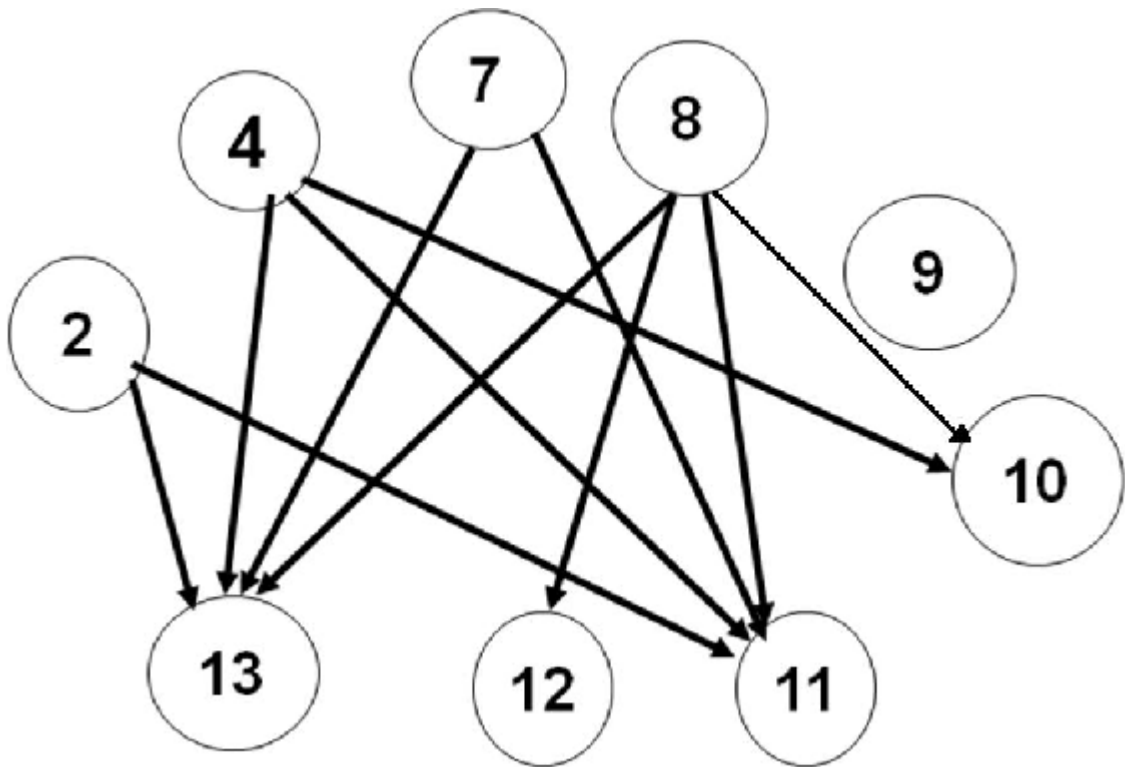


Fig 3.4: graph of generalizes association rule mining

Large 2-itemsets for graph construction as

$\{\{2, 11\}, \{2, 13\}, \{4, 10\}, \{4, 11\}, \{7, 11\}, \{7, 13\}, \{8, 10\}, \{8, 11\}, \{8, 12\}, \{8, 13\}\}$ And create edges between these vertices.

Candidate 3-itemsets are $\{\{8, 10, 12\}, \{8, 11, 12\}\}$

Logical AND operation among these items as

$$\mathbf{BV8^{\wedge}BV10^{\wedge}BV12 = 1111111101^{\wedge}1110110110^{\wedge}1010000101 = 1010000100}$$

$$\mathbf{BV8^{\wedge}BV11^{\wedge}BV12 = 1111111101^{\wedge}1111111110^{\wedge}1010000101 = 1010000100}$$

So no more itemsets are frequent, because all candidate 3-itemsets do not satisfy minimum support condition.

3.3 Comparison

Comparison of the algorithms, Apriori and Primitive Association Rule Mining is given in this section. There are many advantages of Primitive Association Rule Mining over Apriori.

Apriori uses candidate Generate function for generating every candidate k -itemsets and it takes enormous amount of time to generate candidate $k+1$ -itemsets from large k itemsets. However, Primitive Association Rule Mining does not use this function; instead it uses graph based approach after generating of large 2-itemsets.

In primitive association, a graph is constructed with large two itemsets. Using graph, large three itemsets can be generated easily without scanning the database.

At each pass in primitive association, it is enough to use graph with k large itemsets for generating $k+1$ candidate itemsets. Traversal of one link list (adjacency list) takes less time as compared to Apriori generation function.

Secondly in Apriori approach we are accessing transaction as a whole or we can divide into parts but it takes lot of memory whereas in Primitive Association Rule Mining, transactions are converted into bit vector which is based on items. Bit vector representation takes very less times as well as memory, theoretically 32 times less. Primitive Association Rule Mining takes less time since transactions are represented in bit vector form, and we are using logical AND, OR operation which is very fast. Further the bit representation consumes less memory also.

Generalized association rule mining (GARM) using graphs works better than cumulate approach [8] which is a traditional GARM approach. This is due to the fact that it works on bit vector approach, and also there is no need to scan transaction several times and also there is no need to prepare parent-child list at each & every iteration.

CHAPTER 4

LINK ANALYSIS AND HITS ALGORITHM

The explosive growth and the widespread accessibility of the Web has led to a surge of research activity in the area of information retrieval on the World Wide Web. The seminal papers of Kleinberg [1998, 1999] and Brin and Page [1998] introduced *Link Analysis Ranking*, where hyperlink structures are used to determine the relative *authority* of a Web page and produce improved algorithms for the ranking of Web search results.

4.1 Link Analysis Ranking

Ranking is an integral component of any information retrieval system. In the case of Web search, because of the size of the Web and the special nature of the Web users, the role of ranking becomes critical. It is common for Web search queries to have thousands or millions of results. On the other hand, Web users do not have the time and patience to go through them to find the ones they are interested in. It has actually been documented that most Web users do not look beyond the first page of results. Therefore, it is important for the ranking function to output the desired results within the top few pages; otherwise the search engine is rendered useless. Furthermore, the needs of the users when querying the Web are different from traditional information retrieval. For example, a user that poses the query “microsoft” to a Web search engine is most likely looking for the homepage of Microsoft Corporation, rather than the page of some random user that complains about the Microsoft products. In a traditional information retrieval sense, the page of the random user may be highly relevant to the query. However, Web users are most interested in pages that are not only relevant, but also authoritative, that is, trusted sources of correct information that have a strong presence in the Web. In Web search, the focus shifts from *relevance* to *authoritativeness*. The task of the ranking function is to identify and rank highly the authoritative documents within a collection of Web pages. To this end, the Web offers a rich context of information which is expressed through the hyperlinks. The hyperlinks define the “context” in which a Web page appears. Intuitively, a link from page p to page q denotes an endorsement for the quality of page q . We can

think of the Web as a network of recommendations which contains information about the authoritativeness of the pages. The task of the ranking function is to extract this latent information and produce a ranking that reflects the relative authority of Web pages. Building upon this idea, the seminal papers of Kleinberg [1998], and Brin and Page [1998] introduced the area of *Link Analysis Ranking*, where hyperlink structures are used to rank Web pages. Here, we work within the hubs and authorities framework defined. A link analysis ranking algorithm starts with a set of Web pages. Depending on how this set of pages is obtained, we distinguish between *query independent* algorithms, and *query dependent* algorithms. In the former case, the algorithm ranks the whole Web. The PAGERANK algorithm by Brin and Page [1998] was proposed as a query independent algorithm that produces a PageRank value for all Web pages. In the latter case, the algorithm ranks a subset of Web pages that is associated with the query at hand. Kleinberg [1998] describes how to obtain such a query dependent subset. Using a text-based Web search engine, a Root Set is retrieved consisting of a short list of Web pages relevant to a given query. Then, the Root Set is augmented by pages which point to pages in the Root Set, and also pages which are pointed to by pages in the Root Set, to obtain a larger BaseSet of WebPages. This is the query dependent subset of WebPages on which the algorithm operates. Given the set of Web pages, the next step is to construct the underlying hyperlink graph. A node is created for every Web page, and a directed edge is placed between two nodes if there is a hyperlink between the corresponding Web pages. The graph is *simple*. Even if there are multiple links between two pages, only a single edge is placed. No self-loops are allowed. The edges could be weighted using, for example, content analysis of the Web pages. Usually links within the same Web site are removed since they do not convey an endorsement; they serve the purpose of navigation. Isolated nodes are removed from the graph.

Link analysis ranking gives ranking of web pages. Rank of web pages depends on the number of access, items available etc. After assigning the weights using an algorithm we can give priority to web pages in which if a user wants to get information through network then rank of web page will decide as to which related web pages will be provided to the user. As the size increases, its complexity grows so enormously that we can no longer grasp the whole picture. However, within small, local areas, the Web is still

structured orderly because the link structure is built upon a considerable effort of human annotation. Among the algorithms proposed for this purpose, HITS (Hyperlink-Induced Topic Search) algorithm is studied most widely. This algorithm models communities as interconnection between 'authorities' and 'hubs.' Despite the theoretical foundations, HITS algorithm is reported to fail in some real situations.

4.2 HITS (Hyperlink-Induced Topic Search) Algorithm

An overview of the HITS algorithm:

Independent of Brin and Page [1998], Kleinberg [1998] proposed a different definition of the importance of Webpages. Kleinberg argued that it is not necessary that good authorities point to other good authorities. Instead, there are special nodes that act as hubs that contain collections of links to good authorities. He proposed a two-level weight propagation scheme where endorsement is conferred on authorities through hubs, rather than directly between authorities. In his framework, every page can be thought of as having two identities. The *hub* identity captures the quality of the page as a pointer to useful resources, and the *authority* identity captures the quality of the page as a resource itself. If we make two copies of each page, we can visualize graph G as a bipartite graph where hubs point to authorities. There is a mutual reinforcing relationship between the two. A good hub is a page that points to good authorities, while a good authority is a page pointed to by good hubs. In order to quantify the quality of a page as a hub and an authority, Kleinberg associated with every page a hub and an authority weight. Following the mutual reinforcing relationship between hubs and authorities, Kleinberg defined the hub weight to be the sum of the authority weights of the nodes that are pointed to by the hub, and the authority weight to be the sum of the hub weights that point to this authority. HITS algorithm mines the link structure of the Web and discovers the thematically related Web communities that consist of 'authorities' and 'hubs.' Authorities are the central Web pages in the context of particular query topics.

For a wide range of topics, the strongest authorities consciously do not link to one another. Thus, they can only be connected by an intermediate layer of relatively

anonymous hub pages, which link in a correlated way to a thematically related set of authorities. These two types of Web pages are extracted by iteration that consists of following two operations.

$$X_p = \sum_{q, q \rightarrow p} Y_q$$

$$Y_q = \sum_{p, p \rightarrow q} X_p$$

For a page p , the weight of X_p is updated to be the sum of Y_q over all pages q that link to p where the notation $q \rightarrow p$ indicates that q links to p . In a strictly dual fashion, the weight of Y_q is updated to be to the sum of X_p . Therefore, Authorities and hubs exhibit what could be called mutually reinforcing relationships: a good hub points to many good authorities, and a good authority is pointed to by many good hubs.

After getting authority value we can estimate that, which web pages have what priority. According to it we can select some top priority pages, and send it to user when query comes.

Kleinberg [1998] proposed the following iterative algorithm for computing the hub and authority weights. Initially all authority and hub weights are set to 1. At each iteration, the operations O (“out”) and I (“in”) are performed. The O operation updates the authority weights, and the I operation updates the hub weights. A normalization step is then applied, so that the vectors X_p and Y_q become unit vectors in some norm. The algorithm iterates until the vectors converge. This idea was later implemented as the HITS (Hyperlink Induced Topic Search) algorithm. The algorithm is summarized below:

HITS Algorithm

Initialize all weights to 1

Repeat until the weights converge

Do

For every hub p

$$X_p = \sum_{q, q \rightarrow p} Y_q$$

For every authority q

$$Y_q = \sum_{q,p \rightarrow q} X_p$$

Normalize

End

The convergence of the H ITS algorithm does not depend on the normalization. Indeed, for different normalization norms, the authority weights are the same, up to a constant scaling factor. The relative order of the nodes in the ranking is also independent of the normalization.

CHAPTER 5

ASSOCIATION RULE MINING WITHOUT PREASSIGNED WEIGHTS

Ke Sun and Fengshan Bai [6] has purposed a way to find out weights of items and weights of transactions without pre-assigned weights in the database. For finding weights of items and weights of transactions, Ke Sun and Fengshan Bai [6] have taken concept of HITS model which is based on Link Ranking Analysis that provides a way to find out weights of items and weights of transactions.

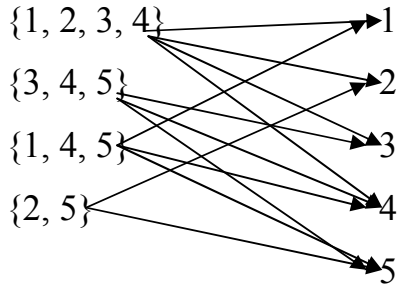
All algorithms that we have discussed earlier, there is not weights to items and transactions. Whereas, using weighted association rule mining, we have to assign weights to items and/or transactions. In Weighted association rule mining, we have to assign weight to items and/or transactions at the beginning in the database. But according to Ke Sun [1], we can find weights of items and weights of transactions. Method to assign weights to items on the basis of items belongs to transactions, and similarly we can find weight of transaction on the basis of items, that are available in transactions. And also, if an item belongs in more transactions, then weight or importance of that item is high. Similarly, if a transaction that contains many items, then weight or importance of that transaction is also high. i.e., a good transaction, which is highly weighted, should contain many good items; at the same time, a good item should be contained by many good transactions. The reinforcing relationship of transactions and items is just like the relationship between hubs and authorities in the HITS model.

So, we can find weights of items and weights of transactions by HITS model. We can assume that every transaction as a link/hub (which contain many items) and items belong to the transaction as an authority (item belongs to many link).

Wang and Su [9] proposed a novel approach on item ranking. A directed graph is created where nodes denote items and links represent association rules. A generalized version of HITS is applied to the graph to rank the items, where all nodes and links are allowed to have weights [10].

TID	Transaction
1	{1, 2, 3, 4}
2	{3, 4, 5}
3	{1, 4, 5}
4	{2, 5}

(a)



(b)

Figure 5.1: The bipartite graph representation of a database. (a) Database.
(b) Bipartite graph

5.1 W-Support: A New Measurement

Item set evaluation by support in classical association rule mining is based on counting. In this section, we will introduce a link-based measure called w-support and formulate association rule mining in terms of this new concept. The previous section has demonstrated the application of the HITS algorithm to the ranking of the transactions. As the iteration converges, the authority represents the “significance” of an item i . accordingly; we generalize the formula of auth to depict the significance of an arbitrary item set, as the following definition shows:

Definition 1. The w-support of an item set X is defined as

$$Wsupp(X) = \frac{\sum_{T: X \text{ subset of } T \wedge T \in D} hub(T)}{\sum_{T: T \in D} hub(T)}$$

Where, $hub(T)$ is the hub weight of transaction T . An item set is said to be significant if its w-support is larger than a user specified value.

5.2 Algorithm for Mining Significant Itemsets without Pre-Assigned Weight

Start

For all items i initialize $auth(i) = 0$

For ($l=0; l < num_it; l++$)

{

For each i set $auth'(i) = 0$

For all transaction t belongs to database

{

Hub(t) = sum of all $auth(i)$ where i belongs to t

Auth'(i) += hub(t) for each item i belongs to t

}

Auth(i) = auth'(i) for each i

Normalize auth

}

$L1 = \{ \{i\} \mid wsupp(i) > minsup \}$

$K=2$

While ($L_{k-1} \neq \emptyset$)

{

$C_k = \text{Apriori-gen}(L_{k-1})$ // C_k contains K -itemsets using Apriori-gen function

For all transactions $t \in D$

{

$C_t = \text{subset}(C_k, t)$ // check whether C_k belongs to Transaction t

 For all candidates $c \in C_t$

{

$c.wsupp += \text{hub}(t)$

}

$H += \text{hub}(t)$

}

Add L_k to result list

$L_k = \{ c \in C_k \mid c.wsupp/h \geq minsup \}$

Increment k by 1

}

End

Example of Mining weighted association rule without preassigned weights

Transactions are:

1=Bread, Milk
 2=Bread, Beer, Diaper, Eggs
 3=Milk, Beer, Diaper, Coke
 4= Bread, Milk, Beer, Diaper
 5=Bread, Milk, Diaper, Coke

Items→ Transactions ↓	Auth(0) Bread (B)	Auth(1) Milk (M)	Auth(2) Beer (Be)	Auth(3) Diaper (D)	Auth(4) Eggs (E)	Auth(5) Coke (C)
1 or Hub(0)	1	1	0	0	0	0
2 or Hub(1)	1	0	1	1	1	0
3 or Hub(2)	0	1	1	1	0	1
4 or Hub(3)	1	1	1	1	0	0
5 or Hub(4)	1	1	0	1	0	1

Table 5.1 Data Representation

Initialize all auth (i) =1, auth' (i) =0;

L=0

Hub (0) = auth (0) +auth (1) = 1+1=2

Auth' (0) =2

Auth' (1) =2

Hub(1)= auth(0)+auth(2)+ auth(3)+auth(4)=1+1+1+1=4

Auth' (0)=6

Auth'(2)=4

Auth'(3)=4

Auth'(4)=4

Hub(2)= auth(1)+auth(2)+ auth(3)+auth(5)=1+1+1+1=4

Auth'(1)=2+4=6

Auth'(2)=4+4=8

Auth'(3)=4+4=8

Auth'(5)=4

Hub(3)= auth(0)+auth(1)+ auth(2)+auth(3)=1+1+1+1=4

Auth'(0)=6+4=10

Auth'(1)=6+4=10

Auth'(2)=8+4=12

Auth'(3)=8+4=12

$\text{Hub}(4) = \text{auth}(0) + \text{auth}(1) + \text{auth}(3) + \text{auth}(5) = 1 + 1 + 1 + 1 = 4$
 $\text{Auth}'(0) = 10 + 4 = 14$
 $\text{Auth}'(1) = 10 + 4 = 14$
 $\text{Auth}'(3) = 12 + 4 = 16$
 $\text{Auth}'(5) = 4 + 4 = 8$

Therefore,

$\text{Auth}'(0) = 14$; $\text{Auth}'(1) = 14$; $\text{Auth}'(2) = 12$; $\text{Auth}'(3) = 16$; $\text{Auth}'(4) = 4$; $\text{Auth}'(5) = 8$

Total = $14 + 14 + 12 + 16 + 4 + 8 = 68$

Normalize,

$\text{Auth}(0) = 14/68 = 0.206$; (i.e. $\text{Auth}'(0)/\text{Total}$)
 $\text{Auth}(1) = 14/68 = 0.206$;
 $\text{Auth}(2) = 0.176$
 $\text{Auth}(3) = 0.235$
 $\text{Auth}(4) = 0.059$
 $\text{Auth}(5) = 0.118$

L=1 (After normalization)

$\text{Total} = 2.431$
 $\text{Auth}(0) = 0.209$
 $\text{Auth}(1) = 0.213$
 $\text{Auth}(2) = 0.174$
 $\text{Auth}(3) = 0.234$
 $\text{Auth}(4) = 0.053$
 $\text{Auth}(5) = 0.117$

L=2 (After normalization)

$\text{Total} = 2.459$
 $\text{Auth}(0) = 0.209$
 $\text{Auth}(1) = 0.214$
 $\text{Auth}(2) = 0.174$
 $\text{Auth}(3) = 0.234$
 $\text{Auth}(4) = 0.052$
 $\text{Auth}(5) = 0.117$

L=3 (After normalization)

$\text{Total} = 2.462$
 $\text{Auth}(0) = 0.209$

Auth(1)=0.215
Auth(2)=0.174
Auth(3)=0.234
Auth(4)=0.053
Auth(5)=0.117

L=4 (After normalization)

Same as L=3

Hence converged.

$$\text{Hub}(0)=\text{auth}(0)+\text{auth}(1) = 0.209+0.215= 0.424$$

$$\text{Hub}(1)= \text{auth}(0)+\text{auth}(2)+\text{auth}(3)+\text{auth}(4)= 0.209+0.174+0.234+0.117= 0.669$$

$$\text{Hub}(2)= \text{auth}(1)+\text{auth}(2)+\text{auth}(3)+\text{auth}(5)= 0.215+0.174+0.234+0.117=0.74$$

$$\text{Hub}(3)= \text{auth}(0)+\text{auth}(1)+\text{auth}(2)+\text{auth}(3)=0.209+0.215+0.174+0.234=0.832$$

$$\text{Hub}(4)= \text{auth}(0)+\text{auth}(1)+\text{auth}(3)+\text{auth}(5)= 0.209+0.215+0.234+0.117= 0.775$$

Therefore,

$$\mathbf{Hub(0)=0.424}$$

$$\mathbf{Hub(1)=0.669}$$

$$\mathbf{Hub(2)=0.74}$$

$$\mathbf{Hub(3)=0.832}$$

$$\mathbf{Hub(4)=0.775}$$

Squares of Hub's

$$[\text{Hub}(0)]^2 = 0.1798$$

$$[\text{Hub}(1)]^2 = 0.4476$$

$$[\text{Hub}(2)]^2 = 0.5476$$

$$[\text{Hub}(3)]^2 = 0.6922$$

$$[\text{Hub}(4)]^2 = 0.6006$$

$$\begin{aligned} \text{Sum} &= [\text{Hub}(0)]^2 + [\text{Hub}(1)]^2 + [\text{Hub}(2)]^2 + [\text{Hub}(3)]^2 + [\text{Hub}(4)]^2 \\ &= 2.468 \end{aligned}$$

Normalize squares of Hub's

$$\text{Hub}'(0)= [\text{Hub}(0)]^2/2.468 = 0.0728$$

Hub'(1)= 0.18136

Hub'(2)= 0.2219

Hub'(3)= 0.2804

Hub'(4)= 0.2434

Final Hub Weights

Hub(0)=√ Hub'(0)=0.27

Hub(1)=0.426

Hub(2)=0.471

Hub(3)=0.53

Hub(4)=0.493

Tabular Representation

Transaction ID	Transaction	Hub Weights
1	Hub(0)	0.27
2	Hub(1)	0.426
3	Hub(2)	0.471
4	Hub(3)	0.53
5	Hub(4)	0.493

Total Hub Weight= 2.19

1-Itemset	Support	w-support
B	0.8	0.78
M	0.8	0.81
Be	0.6	0.65
D	0.8	0.88
E	0.2	0.19
C	0.4	0.44

NOTE:-

Add hub weights of transactions in which item whose w-support need to be calculated is available

W-support=
$$\frac{\text{Sum of Hub weights of transactions containing the item}}{\text{Total Hub weights}}$$

e.g. w-support(B)= (0.27+0.426+0.53+0.493)/2.19= 0.78

Using Apriori approach

Let, $\text{min_w_support}=50\%$ (i.e. 0.5)

Items Selected= B,M, Be,D

1-itemsets

	{B},	{M},	{Be},	{D}
w-support=	0.78	0.81	0.65	0.88

2-itemsets

$$\begin{aligned}\{B,M\} &= 0.27 + 0.53 + 0.493 = 1.293 \\ \{B,Be\} &= 0.426 + 0.53 = 0.956 \\ \{B,D\} &= 0.426 + 0.53 + 0.493 = 1.449 \\ \{M,Be\} &= 0.471 + 0.53 = 1.001 \\ \{D,M\} &= 0.471 + 0.53 + 0.493 = 1.494 \\ \{D,Be\} &= 0.426 + 0.471 + 0.53 = 1.427\end{aligned}$$

Total of all Hub weights= 2.19

$$\begin{aligned}\{B,M\} &= 1.293 / 2.19 = 0.59 \\ \{B,Be\} &= 0.956 / 2.19 = 0.437 \\ \{B,D\} &= 1.449 / 2.19 = 0.662 \\ \{M,Be\} &= 1.001 / 2.19 = 0.457 \\ \{D,M\} &= 1.494 / 2.19 = 0.682 \\ \{D,Be\} &= 1.427 / 2.19 = 0.652\end{aligned}$$

Selected frequent 2-itemsets are:-

$$\{B,M\}, \{B,D\}, \{D,M\}, \{D,Be\}$$

3-itemsets:

$$\begin{aligned}\{B,M,D\} &= 0.53 + 0.493 = 1.023 \\ \{D,M,Be\} &= 0.471 + 0.53 = 1.001\end{aligned}$$

Total of all Hub weights= 2.19

$$\begin{aligned}\{B,M,D\} &= 1.023 / 2.19 = 0.47 \\ \{D,M,Be\} &= 1.001 / 2.19 = 0.457\end{aligned}$$

Selected frequent 3-itemsets are:-

NONE, Since support value < 0.5 (i.e. min_w_support)

Frequent itemsets selected are:-

$\{B\}, \{M\}, \{Be\}, \{D\}, \{B,M\}, \{B,D\}, \{D,M\}, \{D, Be\}.$

5.3 Comparison

There are so many variations of Apriori algorithms that have been developed. We are comparing two algorithms, which are Apriori with their variations and association rule mining without pre-assign weights

AprioriTid and AprioriHybrid are just variations of Apriori. In Apriori, at every step, we have to find candidate k-itemsets, and we have to scan whole database at each k, which is time consuming. So, AprioriTid algorithm has given a solution for finding candidate k-itemsets without scanning whole transaction. This algorithm works on the basis of transaction Id that is associated with every transaction. Apriori Hybrid is combined approach of Apriori and Apriori Tid, in which if some part of transactions (which is stored in other place) do not fit into the memory then use Apriori algorithm, otherwise swap Apriori algorithm to Apriori Tid.

In general, using Apriori AprioriTid and AprioriHybrid algorithms we can find frequent itemsets, whereas we assume that items and transactions have equal weights. Sometimes, it is important to know that whether every items have equal weights or not, if all items have equal weights then Apriori and their variation can do good job for finding frequent itemsets, and if weights of items are not equal, then Apriori and their variations do not work. So, to solve this problem, we have two solutions, whether we can assign weights to items and transactions, or to use some algorithms, so that it can give weights of items and weights of transactions. If we have weights of items in the beginning in the database then we can find frequent itemsets using weighted association rule of mining, otherwise we can use association rule of mining without pre-assign weights, which gives weights of items and weights of transactions using HITS algorithm.

CHAPTER 6

IMPLEMENTATIONS AND RESULTS

There are four implementations have been done in this Thesis, all these implementation have been compared on different datasets. Datasets that has taken which is real life datasets as well as computer generated datasets (IBM Synthetic data generator). There are so many real life datasets which were taken, these are

Kosarak- The kosarak dataset comes from the click-stream data of a Hungarian online news portal, Number of Instances =990,002, Number of Attributes= 41,270.

Chess- A game datasets. Attribute Information:

Classes (2): -- White-can-win ("won") and White-cannot-win ("nowin"). It believes that White is deemed to be unable to win if the Black pawn can safely advance. Number of Instances= 3196, Number of Attributes=36.

Retail- this is retail datasets, Number of Instances =16470, Number of Attributes= 88162

Mushroom- This data set includes descriptions of hypothetical samples corresponding to 23 species of gilled mushrooms. Each species is identified as definitely edible, definitely poisonous, or of unknown edibility and not recommended. This latter class was combined with the poisonous one. The Guide clearly states that there is no simple rule for determining the edibility of a mushroom. Number of Instances = 8124, Number of Attributes = 22.

Connect-This database contains all legal 8-ply positions in the game of connect-4 in which neither player has won yet, and in which the next move is not forced. Number of Instances= 67557, Number of Attributes=42.

PlumbsStar - Number of Instances =49046, Number of Attributes= 49046.

Some implementations have been done on IBM Synthetic data generator datasets, with different sizes, with different transaction, with different number of items in each transaction.

Implementation of Apriori and combined approach of Apriori and HITS algorithms

In Thesis part one; we have implemented one of the new approaches to find frequent itemsets in which items have not pre-assigned weights. This implementation has combined approach of Apriori and HITS algorithm. It has implemented up to 62MB of database. These implementations have been done with various datasets with varying sizes of datasets, with varying of transactions, with varying number of itemsets.

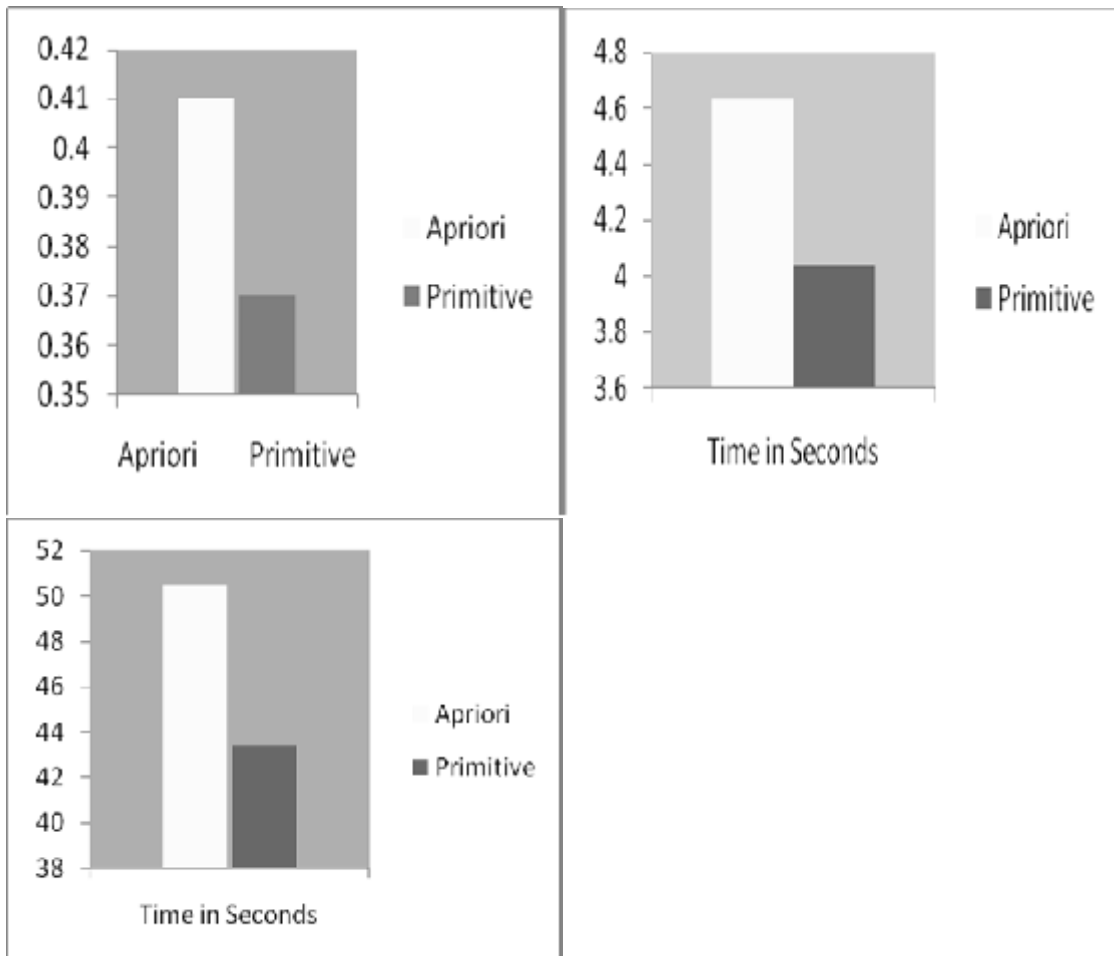
Implementation of Apriori and Primitive Association rule mining algorithms

Implementation in the part two of the Thesis is to find frequent itemsets using graph based algorithm.

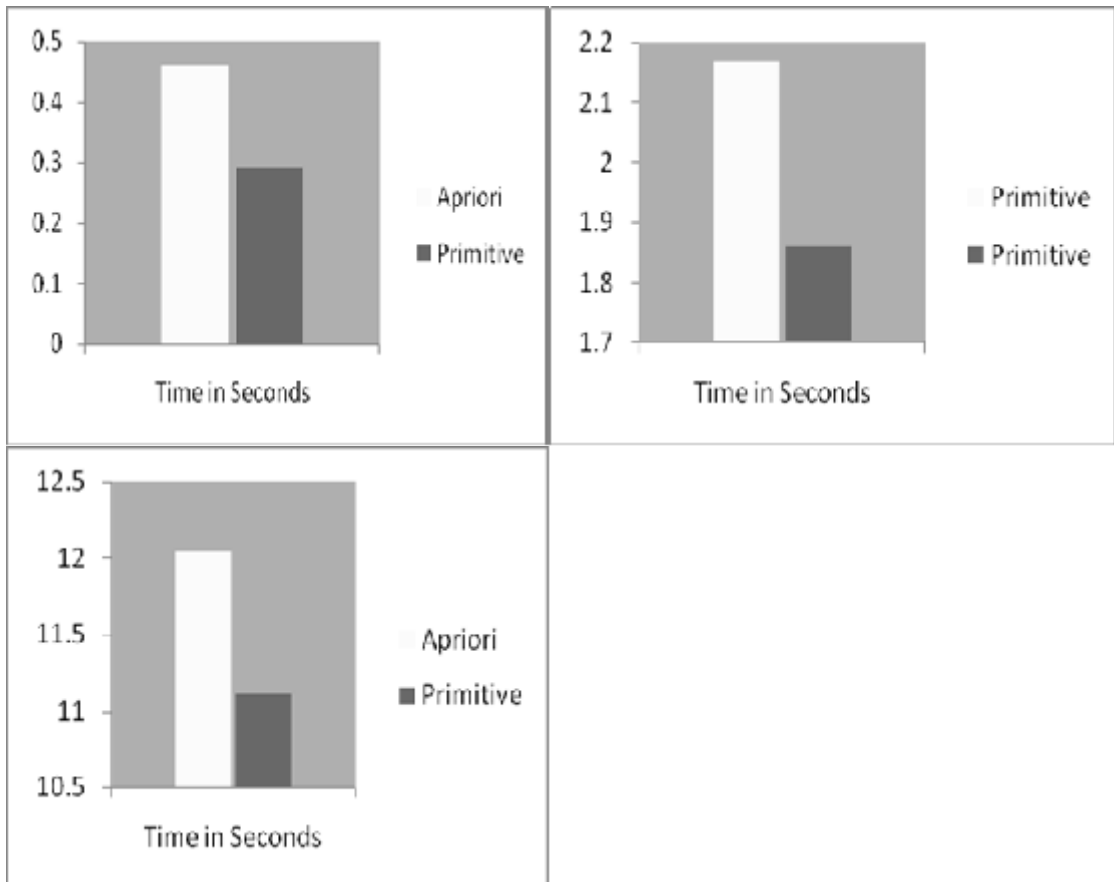
This implementation has been compared with Apriori approach. Here we have given comparison chart for many real life datasets with varying sizes. There are three charts for each dataset. Here X- axis represents dataset names and Y-axis represents time in seconds which is taken by algorithms.

Some results of this algorithm are:

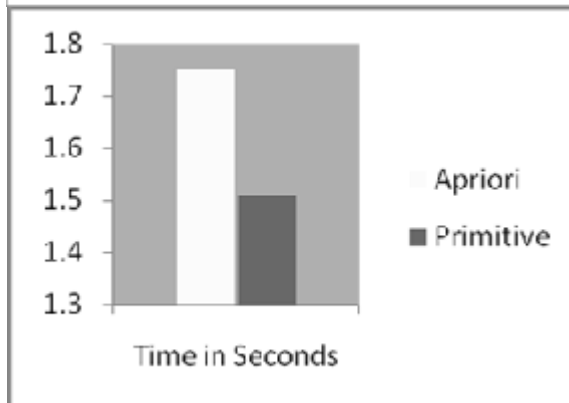
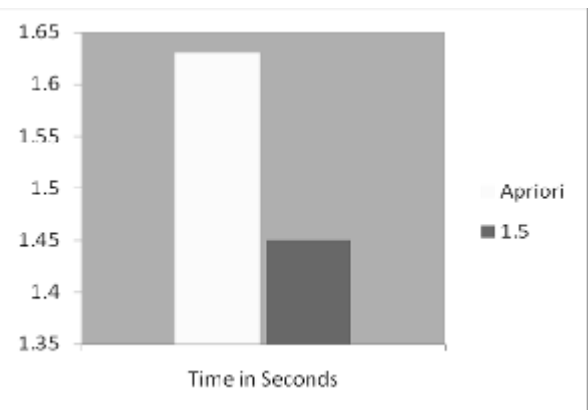
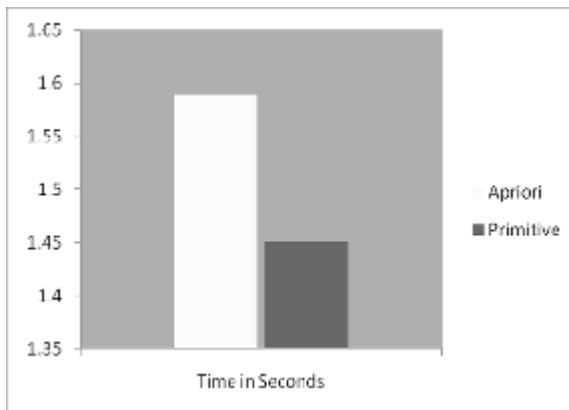
Database name	Chess		
Large itemsets	Time taken by Apriori	Time taken by Primitive	
3	0.41	0.37	
4	4.64	4.04	
5	50.43	43.43	



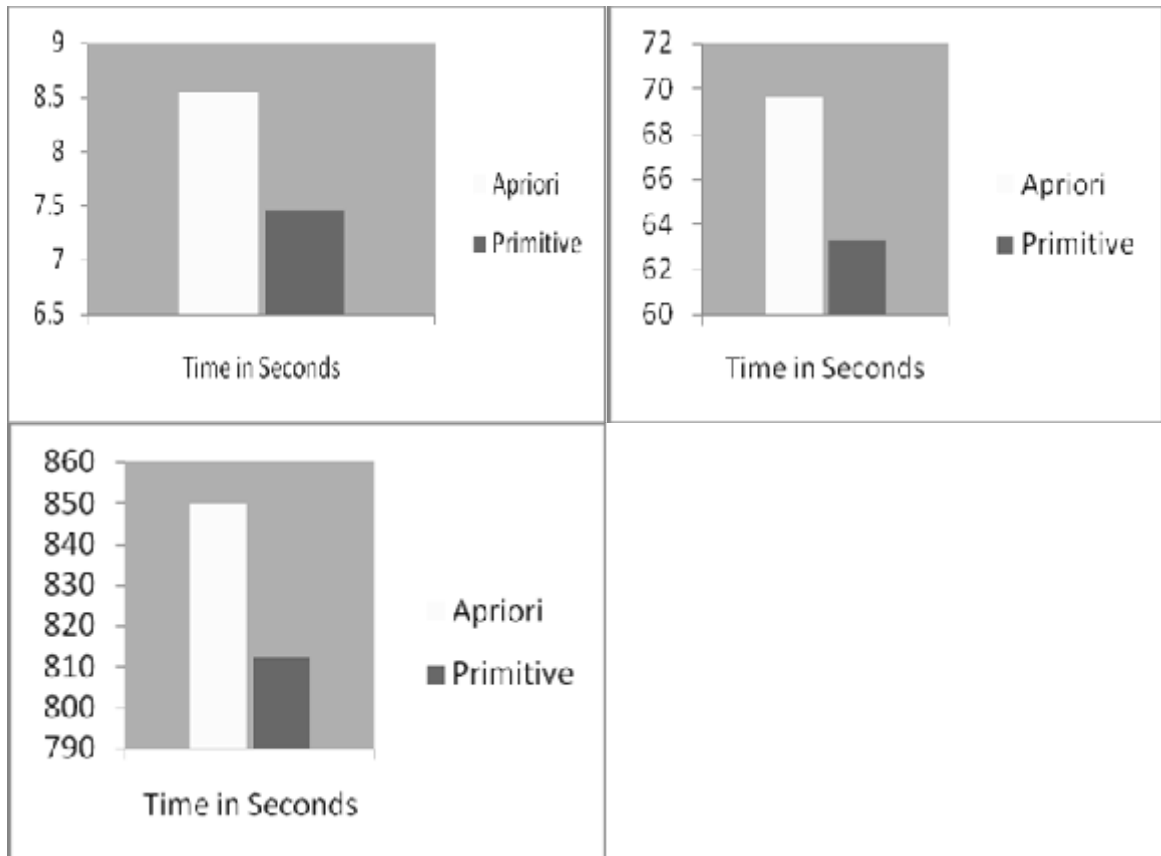
Database name	Mushroom	
Large itemsets	Time taken by Apriori	Time taken by Primitive
3	0.46	0.29
4	2.166	1.86
5	12.04	11.11



Database name	Retail	
Large itemsets	Time taken by Apriori	Time taken by Primitive
3	1.59	1.45
4	1.63	1.5
5	1.75	1.51

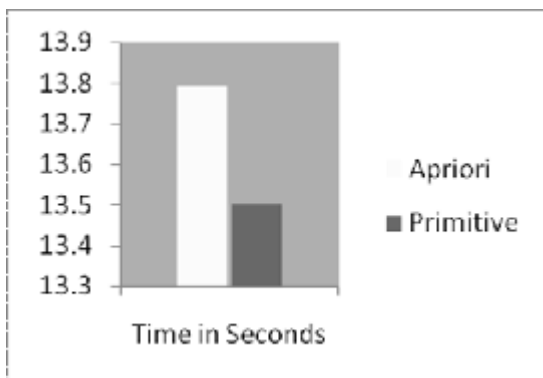
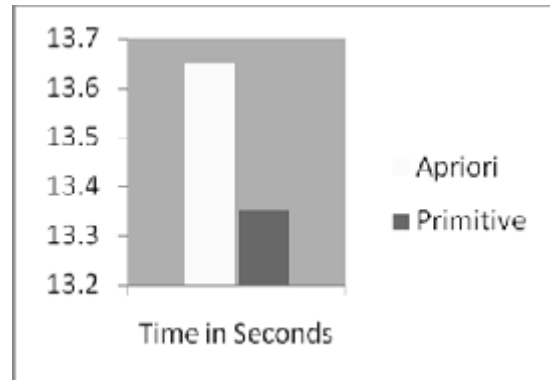
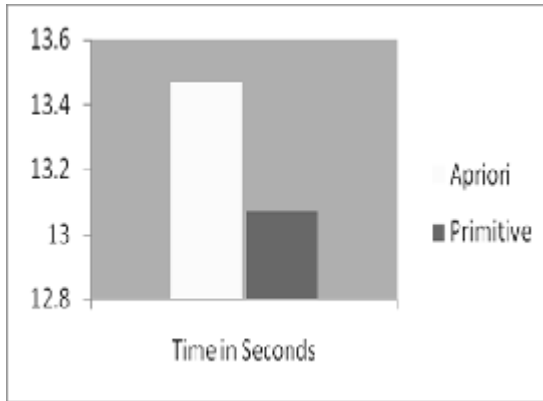


Database name	Connect	
Large itemsets	Time taken by Apriori	Time taken by Primitive
3	8.55	7.46
4	69.58	63.25
5	849.53	812.15



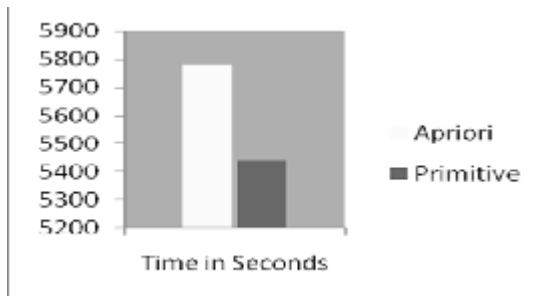
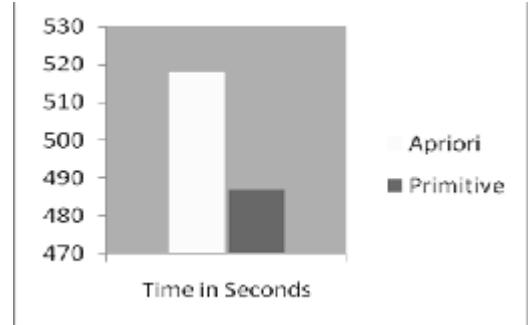
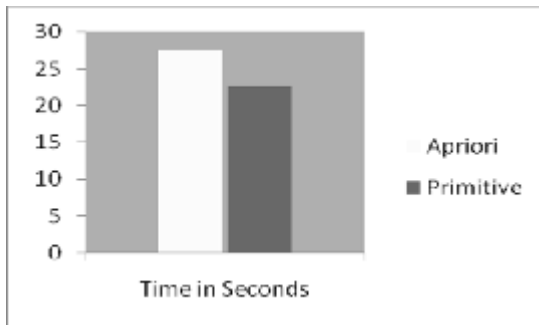
Database name **Kosarak**

Large itemsets	Time taken by Apriori	Time taken by Primitive
3	13.47	13.07
4	13.65	13.35
5	13.79	13.5



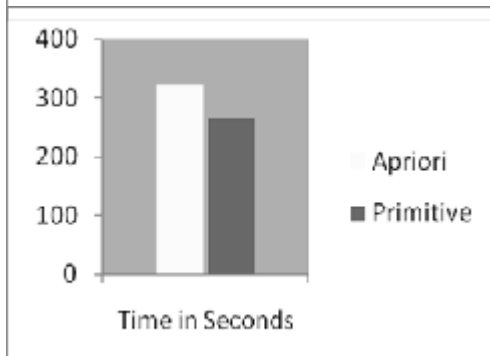
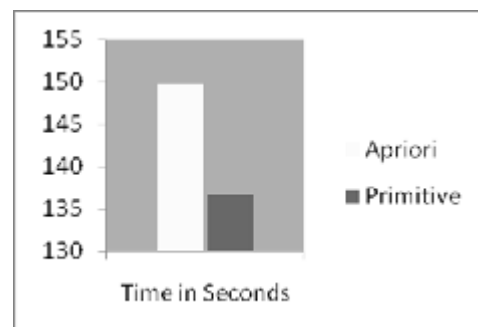
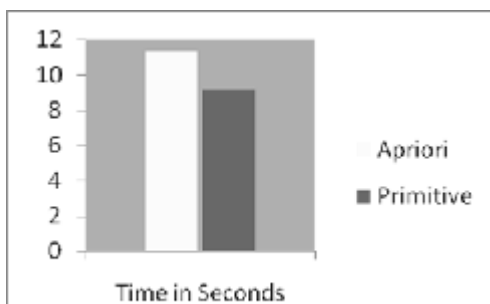
Database name **Pumbs**

Large itemsets	Time taken by Apriori	Time taken by Primitive
3	27.59	22.59
4	517.94	486.94
5	5781.67	5439.89



Database name **PumbsStar**

Large itemsets	Time taken by Apriori	Time taken by Primitive
3	11.37	9.19
4	149.64	136.64
5	322.62	265.62



Implementation of Generalized Association rule mining algorithms

In this implementation, we found large itemsets using graph and transactions. It works better than cumulate approach[8,11] because it uses bit vector approach to store datasets and it need not use to scan transaction many times, and also it need not to prepare parent-child list at each iteration.

Here we have been represented a taxonomy in figure-6.1, and according to this taxonomy we found frequent itemsets on different datasets. These are

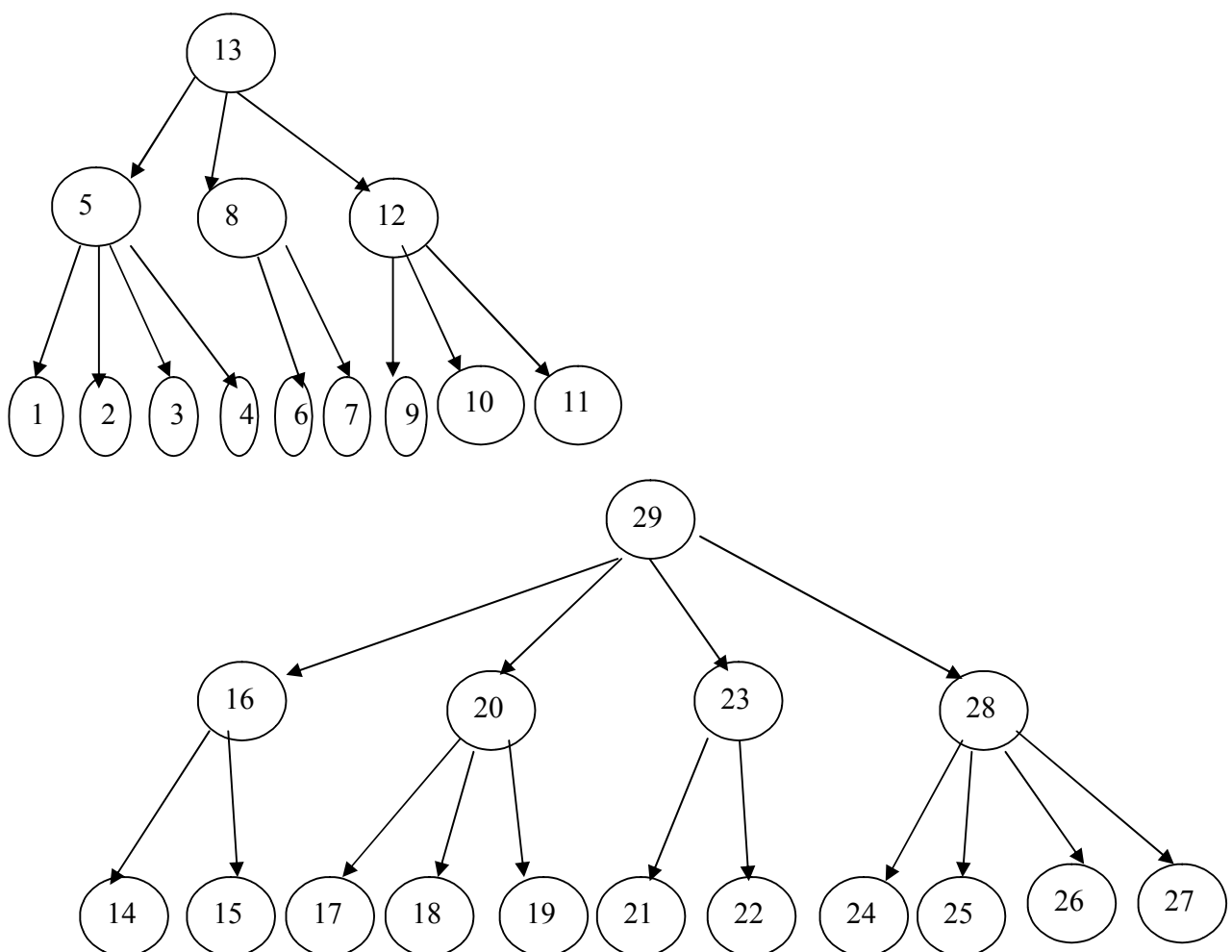


Figure 6.1: Taxonomy of 20 items

Number of transaction 1000, Number of different items 20, frequent itemsets are

{3, 4, 5, 8, 11, 12, 13, 22, 23, 26, 28, 29, (3, 8), (3, 12), (3, 13), (3, 29), (5, 8), (5, 12), (5, 13), (5, 29), (8, 12), (8, 28), (8, 29), (12, 29), (13, 28), (13, 29), (5, 12, 29)}

Number of transaction 500, Number of different items 20, frequent itemsets are

{4, 5, 8, 11, 9, 12, 13, 22, 18, 20, 26, 28, 29, (5, 12), (5, 20), (5, 29), (8, 20), (8, 28), (8, 29), (12, 28), (12, 29), (13, 18), (13, 20), (13, 23), (13, 28), (13, 29), (20, 23), (20, 28), (23, 28), (5, 12, 28), (5, 12, 29) }

7.1 Conclusion

An efficient way for discovering the frequent set can be very useful in various data mining problems, such as discovery of association rules. In this Thesis, new approaches to association rule mining has been explored in depth. In part one of the Thesis, weighted association rule mining without pre-assigned weights was discussed and implementation was done on real life datasets. Comparison of the algorithms, Apriori and Primitive Association Rule Mining was done in this section and there we found many advantages of Primitive Association Rule Mining over Apriori.

In part two of the Thesis, graph based association rule mining for primitive and generalizes approach with taxonomies has been discussed in depth. The efficiency of the approach has been compared with apriori with regard to computational time. This has been demonstrated on number of real life datasets. Here we compared variations of Apriori with and without pre-assigned weights. We found that it was better to use a hybrid approach to fulfill our task in hand but a bit complex to implement as it needs swapping of approaches on the fly.

The link-based model is useful in adjusting the mining results given by the traditional techniques. Some interesting patterns may be discovered when the hub weights of transactions are taken into account. Moreover, the transaction ranking approach is precious for estimating customer potential when only binary attributes are available, such as in Web log analysis or recommendation systems.

7.2 Future Scope

For our approach, the related information may not fit in the main memory when the size of the database is very large. In the future, we shall consider this problem by reducing the memory space requirement. Also, we shall apply our approach on different applications, such as document retrieval and resource discovery in the World Wide Web environment. Best part of previously known algorithms can be combined with to develop hybrid

approaches which perform best for all cases. Number of solutions has been presented, but still a lot of research is possible in this particular area. Descriptive data mining techniques were discussed in the thesis which can be further extended to explore various other approaches. Besides that, the work can be extended to perform predictive data mining task.

And last but not the least; here also we are dealing with the time-space tradeoff problem. As the size of frequent itemset increases, computational time for the initial phases increases exponentially with increase in the requirement in memory space. So, a better way to consider only the relevant transaction or items can be possible field of research. If data cannot fit in the memory than more page faults may occur resulting in the decrease in the performance of the system.

REFERENCES

- [1]. Rakesh Aggarwal, Tomasz Imielinski, Arun Swami, " Mining Association Rules between Sets of Items in Large Databases" ACM Sigmod Conference Washington DC, May 1993.
- [2]. Rakesh Aggarwal , Ramakrishanan Srikant, "Fast Algorithm for mining Association Rules", IBM Almaden Research Centre, Proceedings of 20th VLDB Conference, Santiago, Chile, 1994.
- [3]. Lin D., Z. M. Kedem, "Pincer-Search: An Efficient Algorithm for Discovering the Maximum Frequent Set", *IEEE Tran. Know. and Data Engg.*, Vol. 14, No. 3, May/June 2002, pp. 553-556.
- [4]. J.Han, J.Pei, and Y Yin, "Mining Frequent Patterns Without Candidate Generation", Proc. ACM SIGMOD 2000.
- [5]. C.H. Cai, A.W.C. Fu, C.H. Cheng, and W.W. Kwong, "Mining Association Rules with Weighted Items", Proc. IEEE Int'l Database Engg. And Applications Symp.(IDEAS '98),pp. 68-77, 1998.
- [6]. K.Sun and F.Bai,"Mining Weighted Association Rules Without Preassigned Weights", IEEE Transactions on Knowledge and Data Engineering, Vol. 20, No. 4, April 2008, pages 489-495.
- [7]. Show-Jane Yen, Arbee L. P. Chen: "A Graph-Based Approach for Discovering Various Types of Association Rules". IEEE Trans. Knowl. Data Eng. 13(5): 839-845 (2001).
- [8]. R. Srikant and R. Agrawal In, "Mining Generalized Association Rules", Proc. of the 21st Int'l Conference on Very Large Databases, Zurich, Switzerland, September 1995.
- [9]. K. Wang and M.-Y. Su, "Item Selection by "Hub-Authority" Profit Ranking, Proc. ACM SIGKDD, 2002.
- [10]. Allan Borodin, Gareth O. Roberts, Jeffrey S. Rosenthal, "Link Analysis Ranking : Algorithms, Theory, and Experiments", Panayiotis Tsaparas, ACM Transactions on Internet Technology, Vol. 5, No. 1, February 2005, Pages 231-297.
- [11]. T.Y.Lin, Xiaohua Ho, Eric Louie " A Fast Association Rule Algorithm Based on

Bitmap and Granular Computing", IEEE International Conference on Fuzzy Systems, 2003.

[12].The IBM Synthetic Data Generator, http://www.almaden.ibm.com/software/projects/iis/hdb/Projects/data_mining/datasets/syndata.html.

[13]. Sotiris Kotsiantis, Dimitris Kanellopoulos, "Association Rules Mining: A Recent Overview", GESTS International conference on computer science and engineering, Vol. 32(1) pp- 71-82, 2006.

[14] Cristofor, L., Simovici, D., "Generating an informative cover for association rules", Proc. of the IEEE International Conference on Data Mining, 2002.

[15] Brin, S., Motwani, R. and Silverstein, C., "Beyond Market Baskets: Generalizing Association Rules to Correlations", Proc. ACM SIGMOD Conf., pp. 265-276, May 1997.

[16] Ashrafi, M., Taniar, D. Smith, K., "A New Approach of Eliminating Redundant Association Rules", Volume 3180, 2004, Pages 465 – 474.

[17] Ashrafi, M., Taniar, D., Smith, K., "Redundant Association Rules Reduction Techniques", Volume 3809, 2005, pp. 254 – 263.

[18] Techapichetvanich, K., Datta, A., "Visual Mining of Market Basket Association Rules", Volume 3046, Jan 2004, Pages 479 – 488.

[19] Omiecinski, "Alternative Interest Measures for Mining Associations in Data- bases", IEEE Transactions on Knowledge and Data Engineering, Vol. 15, No. 1, pp. 57-69.

[20] Brin, S., Motwani, R., Ullman, J. D., and Tsur, S., " Dynamic itemset counting and implication rules for market basket data", ACM SIGMOD International Conference on Management of Data, May 13-15, 1997, 255-264.

[21] Liu, B. Hsu, W., Ma, Y., "Mining Association Rules with Multiple Minimum Supports," Proc. Knowledge Discovery and Data Mining Conf., pp. 337-341, Aug. 1999.

[22] Savasere, A., Omiecinski, E., Navathe, S., "Mining for strong negative associations in a large database of customer transactions", Proc. of ICDE. (1998) 494–502.

[23] Wu, X., Zhang, C., Zhang, S., "Efficient Mining of Both Positive and Negative Association Rules", ACM Transactions on Information Systems, Vol. 22, No. 3, July 2004, Pages 381– 405.

[24] Baralis, E., Psaila, G., "Designing templates for mining association rules", Journal of Intelligent Information Systems, 9(1):7-32, July 1997.

PAPER COMMUNICATED

- Garg Deepak, Sharma Hemant, "Comparative Analysis of Various Approaches Used in Frequent Pattern Mining", HiPc 2010 International Conference on High Performance Computing, Dona Paula, Goa, India.