

# Maximal Frequent Itemsets Discovery from Weighted Dataset

T08124



Supervised by:

**Muhammad Nadeem**

Assistant Professor, International Islamic University, Islamabad

Co-Supervisor:

**Saif-ur-Rehman**

Lecturer , Kohat University of Science and Technology, Kohat

Submitted By:

**Kamran Ullah**

374-FBAS/MSCS/F07



**Department of Computer Science  
International Islamic University Islamabad  
(2011)**

**Department of Computer Science  
International Islamic University Islamabad**

Dated: 07/07/2011

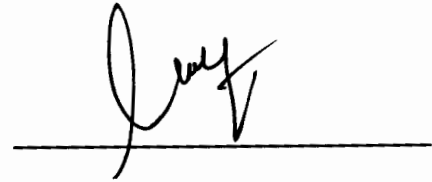
**Final Approval**

It is certified that we have examined the thesis report submitted by **Mr. Kamran Ullah**,  
**Reg No: 374-FBAS/MSCS/F07**, and it is our judgment that this research project is of  
sufficient standard to warrant its acceptance by the International Islamic University,  
Islamabad for the Degree of Master of Science in Computer Science.

**Committee:**

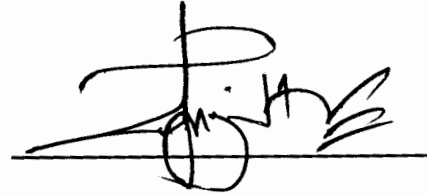
**External Examiner**

**Dr. Hasan Mujtaba**  
Assistant Professor  
Department of Computer Science,  
FAST, Islamabad



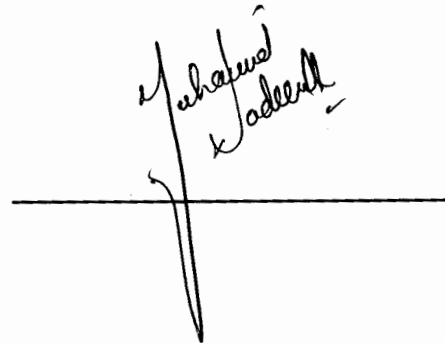
**Internal Examiner**

**Dr. Ayyaz Hussain**  
Assistant Professor  
Department of Computer Science,  
International Islamic University,  
Islamabad



**Supervisor**

**Mr. Muhammad Nadeem**  
Assistant Professor  
Department of Computer Science,  
International Islamic University,  
Islamabad



A dissertation submitted to the  
Department of Computer Science,  
International Islamic University, Islamabad,  
As a partial fulfillment of the requirements  
for the award of the degree of  
MS in Computer Science

## DEDICATION

*This research thesis is dedicated to my beloved parents, respectable teachers, good friends and all those people who have helped me and give me a lot of prayers.*

## **DECLARATION**

I, hereby declare that this research, neither as a whole nor as a part, part has been copied from any source. It is further declare that I have developed this research entirely based on our personal efforts under the able guidance of my supervisor **Mr. Muhammad Nadeem** and co-supervisor **Mr. Saif-ur-Rahman**. If any part of this report is proved to be copied or reported at any stage, I accept the responsibility to face the consequences. No part of this work is inscribed in this report has either been submitted to any other university for the award of degree/qualification.

**Kamran Ullah**  
374-FBAS/MSCS/F07

## **ACKNOWLEDGEMENTS**

All praise to **Allah**, the Almighty, the most merciful and compassionate, Who granted me perseverance to accomplish this research work.

I express my profound gratitude to my kind co-supervisor **Mr. Saif-ur-Rahman**. His motivation guided me to this achievement and kept our morale high with his valuable suggestions and personal encouragement. He was readily available for consultations, without which I never have been able to complete this work.

I cannot ignore the helps and guidance of my supervisor **Mr. Muhammad Nadeem**. He is very kind and helpful. He guided me throughout this research and always checked my work with valuable comments and good suggestions.

I am also thankful to my teacher **Mr. Muhammad Imran Saeed** for his inspiring attitude and untiring help during this project effort. He did a lot of efforts for my success.

I would like also to thank our parents, teachers, friends and all those who helped me and supported us during the project.

**Kamran Ullah**  
374-FBAS/MSCS/F07

## Project in Brief

- Project Title:** *Maximal Frequent Itemset Discovery from Weighted Dataset*
- Objective:** The purpose of this research is to study the Maximal frequent Itemset Discovery in the association rule mining. I have also successes in proving that array based weighted dataset is efficient for search in top down fashion. The experimental results show the effectiveness of this technique.
- Organization:** Department of Computer Science,  
International Islamic University, Islamabad.
- Undertaken By:** Mr. Kamran Ullah (374-FBAS/MSCS/F07)
- Supervised By:** Mr. Muhammad Nadeem  
Assistant Professor, International Islamic University Islamabad
- Co-Supervisor:** Mr. Saif ur Rahman  
Lecturer, Kohat University of Science and Technology, Kohat
- Tool Used:** i) C++ for development of project.  
ii) ARTool used for generating and conversion of synthetic dataset.
- Operating System:** Windows Vista (64-bit)
- System Used:** DELL INSPIRON 1545  
Intel Pentium 2.16 GHz Dual Core
- Date Started:** June, 2009
- Date Completed:** October, 2010

## Abstract

Association Rule mining is one of the methods in the data mining, in which the frequent itemsets are found. The previous work shows great improvement in the area. But they first find the frequent itemsets then find the maximal frequent itemsets. The bottom-up approach was used to mine the maximal frequent itemset. For this purpose, they use incremental order searching which causes a very large number of intermediate results in way to find the maximal frequent itemset. This required large space as well as complex pruning methods.

In the proposed scheme, the maximal frequent itemset will be found from weighted dataset and then the frequent itemsets. The proposed scheme has been implemented in top-down approach. In this way the intermediate results are avoided are not used in the searching of maximal frequent itemset. The weighted dataset has also been used which check the support of an itemset so efficiently. In this scheme the idea has been implemented to search MFI in top-down approach, which first discovers the MFI and then the frequent itemset by using subset methods.

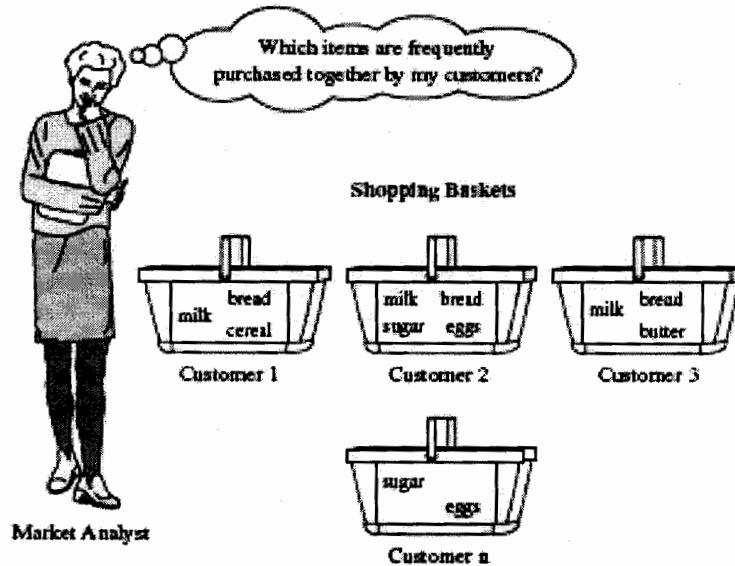


## TABLE OF CONTENTS

Chapter No.	Contents	Page No.
<b>1. INTRODUCTION</b>		
1.1	Introduction .....	1
1.2	Data Mining and DBMS.....	2
1.3	Data Mining Techniques/Methods .....	3
1.3.1	Classification .....	3
1.3.2	Cluster Mining.....	4
1.3.3	Web Mining.....	4
1.3.4	Association Rule Mining.....	4
1.3.5	Process of Association Rule Mining .....	5
1.3.5.1	Frequent Itemset Generation .....	5
1.3.5.2	Association Rule Generation.....	5
1.4	Terminologies Used in Association Rule Mining .....	5
1.4.1	Support .....	5
1.4.2	Confidence.....	6
1.4.3	Frequent Itemset .....	6
1.4.4	Candidate Generation .....	6
1.4.5	Maximal Frequent Itemset.....	7
1.4.6	Closed Frequent Itemset.....	7
1.5	Different Format of Datasets .....	8
1.5.1	DB Format.....	8
1.5.2	ASCII Format.....	8
1.5.3	Bitmap Structure.....	10
1.6	Summary.....	11
<b>2. LITERATURE SURVEY</b>		
2.1	Introduction .....	13
2.2	Literature Survey .....	13
2.3	Some Basic Algorithm of Association Rule.....	13
2.3.1	Apriori Algorithm.....	13
2.3.2	FP-Growth .....	15
2.3.3	Éclat Algorithm .....	15
2.4	Algorithms for Mining Maximal Frequent Itemset .....	16
2.4.1	Pincer-Search Algorithm .....	16
2.4.2	Max-Miner Algorithm .....	17

Chapter No.	Contents	Page No.
	2.4.3 GenMax Algorithm .....	19
	2.4.4 MAFIA .....	20
	2.4.5 PADS.....	21
	2.4.6 MMFI .....	23
	2.4.7 Hybrid Algorithm.....	24
2.5	Analysis of Previous Work.....	25
	2.5.1 Top-Down and Bottom-up Approach.....	25
	2.5.2 Pruning Techniques.....	25
	2.5.3 Scanning the Database/Dataset.....	25
2.6	Problem Statement.....	26
2.7	Summary.....	26
 <b>3. PROPOSED SOLUTION</b>		
3.1	Introduction .....	28
3.2	Problem Domain.....	29
	3.2.1 Multiple Scans of Database/Dataset .....	29
	3.2.2 Large Number of Candidate Generation .....	29
	3.2.3 Finding Support of an Itemset.....	30
	3.2.4 Algorithm Execution Time.....	30
	3.2.5 Large Search Space .....	31
3.3	Proposed Solution.....	31
	3.3.1 Reduced Database Scans .....	32
	3.3.2 Reduce the Candidates .....	33
	3.3.3 Efficient Support Counting Method .....	33
	3.3.4 Reduce Search Space.....	33
	3.3.5 Reduce the Execution Time.....	34
3.4	Scope of Research .....	34
3.5	Summary.....	35
 <b>4. IMPLEMENTATION</b>		
4.1	Introduction .....	37
4.2	Data Structure Used in our Algorithm.....	37
4.3	Tool & Techniques.....	37
	4.3.1 C++ Language .....	38
	4.3.2 ARTool.....	38

Chapter No.	Contents	Page No.
4.4	Software Architecture.....	38
4.4.1	Generating Datasets by ARTool.....	39
4.4.2	Conversion of Dataset.....	40
4.4.3	ASCII Conversion.....	40
4.4.4	Conversion from ASCII to Binary.....	41
4.4.5	Weight Generation.....	42
4.4.6	Creating Combinations.....	43
4.4.7	Pruning Methods.....	44
4.4.8	Support Checking.....	44
4.4.9	MFI Generation.....	45
4.4.10	Termination.....	45
4.5	Summary.....	46
 <b>5. EXPERIMENTAL RESULTS</b>		
5.1	Introduction.....	48
5.2	Generated Datasets for Experiments.....	48
5.3	Experimental Results.....	49
5.3.1	Experimental Results of MFIGen.....	49
5.3.2	Experimental Results of MAFIA.....	51
5.3.3	Experimental Results of PADS.....	53
5.4	Time Analysis in Different Scenario.....	55
5.4.1	Increasing the number of Items.....	55
5.4.2	Increasing the Transactions.....	56
5.4.3	Comparison with MAFIA.....	57
5.5	Summary.....	57
 <b>6. CONCLUSION AND FUTURE WORK</b>		
6.1	Introduction.....	60
6.2	Applications of Association Rule.....	60
6.3	Conclusion.....	61
6.4	Future Work.....	62
6.5	Summary.....	62
 <b>APPENDIX A</b>		
	REFERENCES.....	64



**Figure 1.2: Example of Market Basket Analysis [2]**

Another example is of the bank loan system. If a bank launches a loan scheme, it would like to give the loan to appropriate customers. These customers can be traced by using some criteria which will be based upon the customer balance, transaction frequency in the current time, income level, and age and may be marital status. The analysis should be done to determine that either the customer is capable to return the loan or not. These analysis and criteria can be done by using the Data mining tools. They will classify their customer according to their status.

They will prefer the person with large income, frequent transactions, in the middle age and based on other criteria which are helpful to analyze the customer.

It means that the data mining plays an important role in the different field for analysis such as business. We can easily analyze our database from different perspective using data mining tools. By using these tools, the hidden patterns are extracted from database/dataset which gives us some useful information which is presented in knowledge. The knowledge extracted from datasets/databases is helpful in decision making.

## 1.2 Data Mining and DBMS

Database Management System is used to store the data and SQL is used to query the DBMS for retrieval different data. For example we want to calculate the monthly sale

of a company. We can also group this data by different criteria such salesman or different items.

Data mining tools extract hidden patterns which are used for discovering knowledge. For example we have to classify different itemsets which is frequently purchased together [3]. This discovered knowledge is helpful in decision making.

Here a question arises that why should we not use the Database Management System to extract the knowledge. Actually SQL queries are used retrieve the data from database by which we can compare the data or analyze in a simple way. For example we can compare the student of private schools with the government schools. This can be used for simple analysis or comparison of two types of schools. But if the analysis is based different attributes of the students like income level of parents, motivation from society, location of the schools etc. Sometime it is impossible by SQL queries, or if it is possible, then it would be so hard and complex. We have to right so many queries for that purpose. But it can be done very easily by using Data mining tools. We can easily and quickly extract hidden patterns which is very helpful in knowledge discovery.

### **1.3 Data Mining Techniques/Methods**

As we have mentioned that Mining Process is used in many fields so the data is also analyze by different methods. Therefore different methods are being adopted to mine the data and extract hidden pattern to discover knowledge. The Data mining is divided into different branches. Here we discuss some of them.

#### **1.3.1 Classification**

The classification method is used to divide the data in different classes and then analyze it. In this method some classes are defined and records are read to classify the data. By this way different profiles are created for data and are used for analysis. The classes are defined by using characteristics and the data with similar characteristics are put in the same class. The data mining application is used to model the classes which are used for prediction. For example a bank is issuing a loan scheme and has to analyze the behavior of different type of customers. It classifies the customers and identifies the customer which is to expect to be taking the loan. Then they offer loan to that specific customer.

### 1.3.2 Clustering Mining

In cluster mining the data are grouped on the basis of similarities. For example the customer records are analyzed. So the customers with similar properties must be resides in the same groups and that groups are called clusters. The clusters are used to identify users and its behavior. For example a survey is made and we do not know the customers. So we cannot make the classes. Then the data is to be read and the clusters will be defined for the similar data.

### 1.3.3 Web Mining

Web mining is used to extract information from World Wide Web by using traditional data mining tools and techniques. These tools gather and integrate useful information from web, in which the hidden pattern is extracted.

It has three types. Web Usage Mining uses to find pattern on customer visits to a site. Web Content Mining, the content of a site is used to discover pattern. It is often called Text Mining. Web Structure Mining, this kind of web mining is used to analyze nodes and connections by using graph theory.

### 1.3.4 Association Rule Mining

Association Rule Mining is the branch of data mining in which we associate items or itemsets with each others. This is used to find the itemsets which is frequently occurs in the database/dataset together. As we have given the example of milk and diaper in the earlier section. The association rule is discussed in the next section in detail.

Association rule mining is used to associate itemset with each other which are frequently purchased together. In other words, the frequent items/data in the dataset can be associated with each other by association rule.

The association rule was proposed by R. Agrawal in 1993 in the article. In this article he had also proposed an algorithm which is called Apriori Algorithm [4]. This was the first algorithm used for association rule mining. Now a number of algorithms was proposed and researched for the association rule. The efficiency of these algorithms is increasing day by day. We have mentioned some basic algorithm in this chapter and have also discussed some other algorithms in my literature survey which are about the Maximal Frequent Itemset and appropriate to my research topic.

### 1.3.5 Process of Association Rule Mining:

The association rule mining is divided into main steps.

#### 1.3.1 Frequent Itemset Generation:

Frequent Itemset consist of those itemsets which frequently occurs in the given datasets or itemsets which are found in the dataset up to provided *minimum support* thresholds.

*Support* is the frequency of the itemsets that how many transactions in the database have itemsets. For example if we have item A and item B in an itemset. So the support is as under:

#### 1.3.2 Association Rule Generation:

Association Rule Generation is the main step in association rule mining. This can be done after finding frequent itemsets [1]. Once frequent Itemset have been found then Association Rule can be found using Support and Confidence.

*Confidence* is defined as transactions which has both the itemsets for which we are finding the association rule. It can be calculated as the conditional probability of the items. For two items A and B, we have the following equation to find the confidence [2].

$$\text{Confidence } (A \rightarrow B) = P (A/B) \quad (1.1)$$

The support and confidence is calculated in probability but the value is always represented from 0% to 100%. If the Support and Confidence is equaled to or greater than minimum given threshold, so the Association Rule is called Strong Association Rule.

## 1.4 Terminologies Used in Association Rule Mining

Association rule process used some terminologies. These are as under:

### 1.4.1 Support

We have already explained the term support. Support is basically the frequency of the item/itemset in the dataset. It is calculated as percentage of the transactions having the item/itemset with the total number of transactions.

For example, we have 1000 transaction in our dataset. Suppose itemset {A} occurs 250 times in the dataset. So the support of the {A} is as under:

$$\text{Support}(A) = 250/1000 = 0.25 = 25\% \quad (1.2)$$

### 1.4.2 Confidence

The term confidence has also been discussed in the previous sections. The confidence is used for the itemset for which the association rule is to be fined. In this the conditional probability is calculated for the itemsets. It means that the transactions will be calculated for the confidence in which the both itemsets are occurs in the same transactions.

For example, we want to find the association rule from itemset {A} to itemset {B}. So if both of the itemsets are occurred together in the 200 transactions of a database having 1000 transactions. So the confidence of the {A} and {B} is as under:

$$\text{Confidence}(A \rightarrow B) = P(A/B) = 200/1000 = 0.2 = 20\% \quad (1.3)$$

### 1.4.3 Frequent Itemset

Itemset is the set of items in the dataset/database. It may contain item(s) from 1 to  $n$ , where  $n$  is the total number of items in the dataset/database. We often call it  $k$ -itemset where  $k$  is the number of items the itemset.

The itemset is to be called Frequent Itemset if and if it satisfies the minimum threshold of support. For example we have a transactional database has 1000 transactions and the given minimum support is 2%. So the itemsets which have occurs in 20 or greater transactions. In the above example of section 2.3.1, the itemset {A} is frequent itemset, because it has support greater than minimum support given.

### 1.4.4 Candidate Generation

Candidate generation is used in the algorithms for finding the frequent itemsets. When the  $k$ -itemset has some frequent itemsets then the next candidate can be generated by checking the frequent with  $(k+1)$ -itemsets. This process is repeated until the new frequent itemset is discovered.



The example of Candidate Generation is such as if we have five items in a transactional dataset i.e.  $I = \{A, B, C, D, E\}$ . If we want check frequent itemsets having size from 1 to 5. So at first check, the 1-itemsets has  $\{A\}, \{B\}, \{D\}$  as frequent. Then the 2-itemsets have to be checked and these will be generated by taking the combinations of the frequent itemset having size 1. So the candidates for 2-itemset are  $\{A, B\}, \{A, D\}, \{B, D\}$ .

#### 1.4.5 Maximal Frequent Itemset

Maximal Frequent Itemset (shortly called MFI or MFIS) is the that Frequent Itemset which has no superset as Frequent Itemset or the Frequent Itemset is called MFIS which is not the subset of any of the Frequent Itemset.

As we discussed that Frequent Itemset has size from 1 to n, so a dataset may have a large number of frequent itemsets of different size. So itemsets with large size have subsets as frequent itemset. So the itemset which is the superset of different itemsets and having support equal to or greater than the support threshold are said to be Maximal Frequent Itemsets (MFIS).

For example if we have five items in a transactional database A, B, C, D and E. Suppose the Frequent Itemsets are  $\{A\}, \{B\}, \{C\}, \{E\}, \{A, B\}, \{A, C\}, \{A, B, C\}, \{B, E\}$ . So the MFIS are  $\{A, B, C\}$  and  $\{B, E\}$ .

Now a day the Maximal Frequent Itemset discovery is an emerging area of research and many researchers have made algorithms for it like GenMax, MaxMiner etc. These Algorithms along with some other algorithms are mentioned in chapter as my literature survey. My own research is also based on discovery of Maximal Frequent Itemset.

#### 1.4.6 Closed Frequent Itemset

Closed Frequent Itemset is another type of Frequent Itemset. The itemset is said to be Closed Frequent Itemset if it frequent itemset and there is no superset of it with the same support count.

For example we have itemset  $\{A, B, C\}$  having support 3,  $\{B, C\}$  having support 4,  $\{A, B\}$  having support 3 and  $\{B, E\}$  having support 2. If the minimum support is 2

thus all the itemset are frequent, but  $\{A, B, C\}$ ,  $\{B, C\}$  and  $\{B, E\}$  are closed frequent itemsets.  $\{A, B\}$  is same support as  $\{A, B, C\}$  so it is not closed frequent itemset.

## 1.5 Different formats of Datasets

The dataset to be mined by association rule can be representing in different formats. It is according to the requirements. We have use three basics formats in our research project implementation.

### 1.5.1 DB Format

DB format (\*.db) is that format which can be generated synthetically by using ARTool Software. The ARTool [27] software can generate random dataset in this form which can be used to check the algorithm for mining association rules.

This form of dataset can be read by ARTool and applicable in the C++. But I am using C++ for the implementation of our research project. So that this form is converted to ASCII format, which is discussed next and ARTool is also able to do so.

### 1.5.2 ASCII Format

ASCII format (\*.asc) is special standard format of the dataset used to check the mining algorithms. As we discussed that ARTool is used to convert to the DB format to ASCII format. The transactions are arranged in proper order in the ASCII format. Unlike DB format, the transaction of ASCII format is understandable to human. This format is supported by C++. We use a C++ program to read the dataset in ASCII format.

The ASCII format divides the transactional datasets into two parts. The first part has the items descriptions along with an identifier, incremental number, which represent the item in the transactions. Each line is start with a number and then the description of the items separated by a blank space. The next item is represented in the line. The example is shown in figure 1.3.

The second part includes the transitions of the datasets. The first line of this part is "BEGIN\_DATA" to indicate that the transactions are started. The transactions are started from next line. Each line represents a transaction having the identifiers of the item which is given in the first part. The items' identifiers are separated by a blank

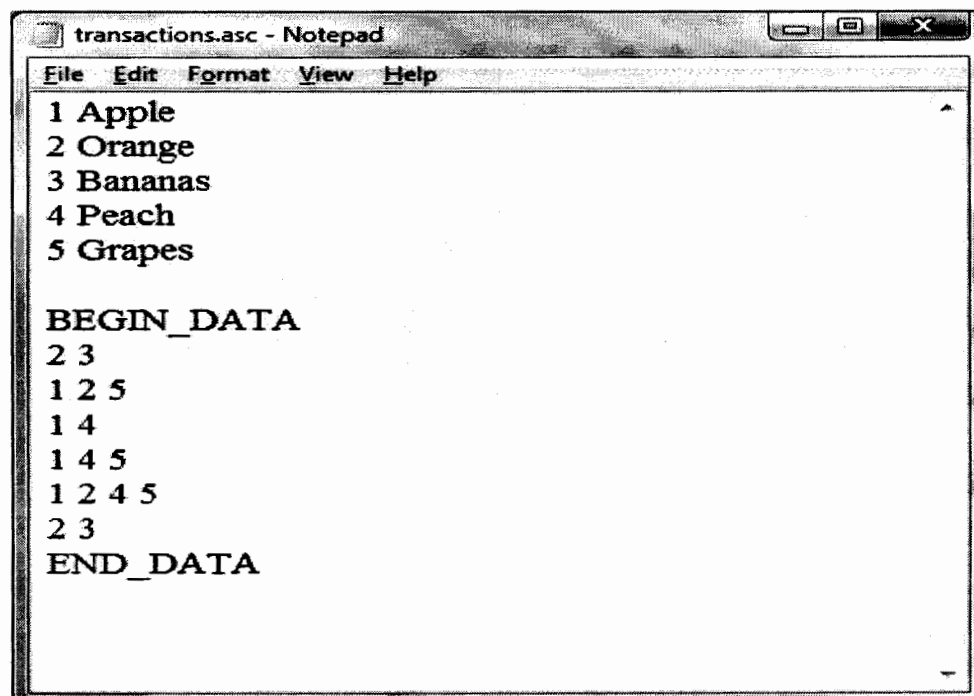
space. At the end of the transaction, the line is broken and next transaction is started from the next line. In the last, the word “END\_DATA” in a line is used to end the transactional dataset. The figure 1.3 shows the example.

The advantage of this format is simplicity and ease of access in the programming tools such as C++. Note that the transaction ID is not included as it has no matter in the association rule mining. The table 1.1 shows an example of transactional dataset having five items and six transactions. The ASCII format of this dataset/database is shown in the figure 1.3.

Transaction ID	Purchased Items
100	Orange, Bananas
200	Apple, Orange, Grapes
300	Apple, Peach
400	Orange, Peach, Grapes
500	Apple, Orange, Peach, Grapes
600	Orange, Bananas

**Table 1.1: An example of transactional dataset**

The above transactional dataset can be represented in the ASCII format in the following manner.



```

1 Apple
2 Orange
3 Bananas
4 Peach
5 Grapes

BEGIN_DATA
2 3
1 2 5
1 4
1 4 5
1 2 4 5
2 3
END_DATA

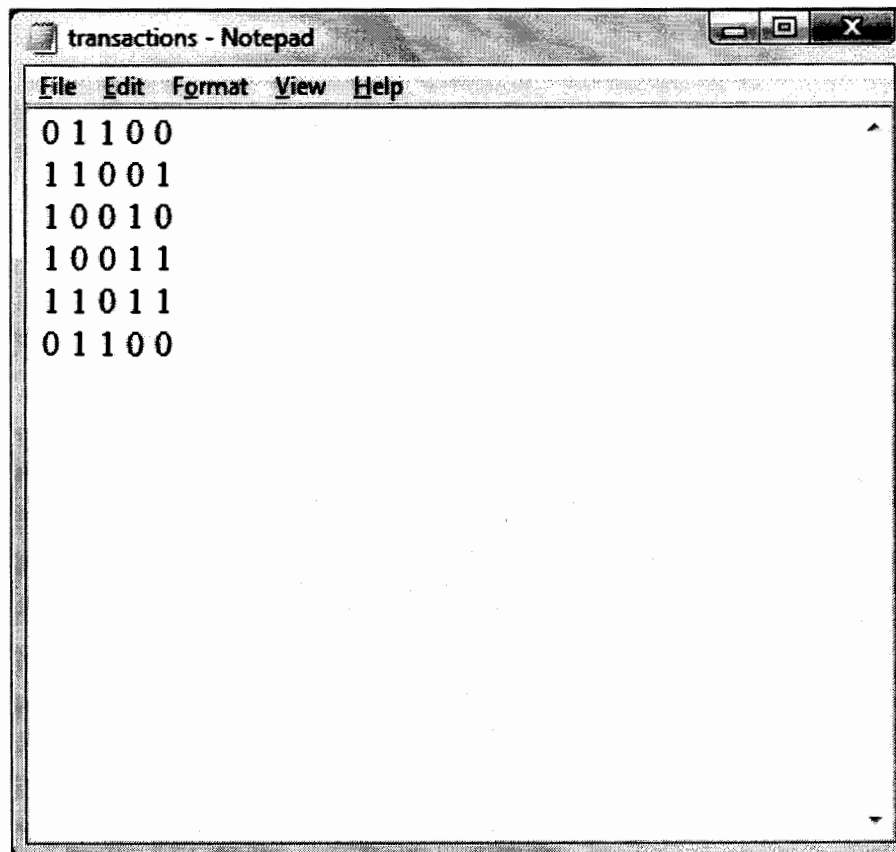
```

**Figure 1.3: The ASCII Format of Transactional Database**

### 1.5.3 Bitmap Structure:

Bitmap Structure is another way to represent the dataset in a different format which is very unique in style [11]. This is often called binary format (\*.bin). The file of this format is created from ASCII format by using any programming tool. I have used C++ program to generate the binary format from the ASCII format of the dataset.

In this format the items in a transaction is represented by 1 if it is purchased or 0 otherwise. Every item has its own place in the transaction. So if the item is there in a transaction, 1 is inserted and if it does not in the transaction the 0 is inserted. Like ASCII format, it has also a line for a transaction and every item (or bit, as it is either 0 for absent or 1 for present) is separated by a blank space. But it has only one portion just for transactions. The binary format for the above given example is shown in figure 1.4.



```
File Edit Format View Help
0 1 1 0 0
1 1 0 0 1
1 0 0 1 0
1 0 0 1 1
1 1 0 1 1
0 1 1 0 0
```

Figure 1.4: Binary format of a dataset

## 1.6 Summary

In this chapter we discussed the data mining introduction, its area and different techniques used for different purposes. As our research area is association rule mining, so we have extends it and described some terminologies used in it. We have also discussed different formats of datasets used in data mining. In the next chapter the related work is discussed which is used to the discovery of maximal frequent itemset.

**Chapter 2**

**LITERATURE  
SURVEY**

## 2.1 Introduction

The literature review is a foundation to the research idea. In this the previous work related to research is critically reviewed and analyzed some problems found in it. Some of the previously published algorithms were reviewed and are going to discuss in this chapter. These algorithms are all related to the discovery of maximal frequent itemsets. The algorithm is in the order of published dates. The reviews lead to formulate the problem statement given in the last section.

## 2.2 Literature Survey

Our research is based on the area of association rule mining and the discovery of Maximal Frequent Itemset. The research and problem so far in association rule mining is to find frequent itemsets in the transactional database. We have reviewed some already proposed algorithm for the discovery of Maximal Frequent Itemset. Besides these we have also reviewed some basic algorithm used in association rule mining. These algorithms are discussed in the next sections.

## 2.3 Some Basic Algorithm of Association Rule

Association rule is one of the important and interesting mining methods for pattern discovery. We have discussed that association rule consists of two steps, i.e. finding frequent itemsets and association rule generation. But the complicated step is the frequent itemset discovery. The researchers have proposed a number of algorithms to find association rule. We have discussed some basic algorithm here.

### 2.3.1 Apriori Algorithm

Apriori Algorithm is proposed by R. Agrawal and R. Srikant in 1993[4]. This is one of basic algorithms to find the frequent itemset in a simple way. Apriori algorithm finds all sets of itemsets that have support no less than minimum support. Itemsets that satisfy minimum support constraint are called frequent itemsets. Apriori is characterized as a level-wise complete search (breadth first search) algorithm to prune the itemset which are not frequent: "If an itemset is not frequent, any of its superset is never frequent" [4].

The working of Apriori algorithm is as it scans all the itemsets from 1-itemset to (k-1) itemset for k-itemset to generate candidate itemset. First, it scans for 1-itemset and prune the items which does not support the *min\_sup*. Then generate the candidate itemset for 2-itemset and repeat until no frequent itemset is generated.

Apriori property is used to improve the generation of frequent itemsets in different level. This property made the searching efficient because the pruning is on the basis of non-frequent itemset. The Apriori property is given below.

**Apriori property: *All nonempty subsets of a frequent itemset must also be frequent [2].***

This property tells that if an itemset is frequent then the subset must also be frequent because for counting the frequency, all the items in the itemset must be included. So this applied that all the items in the item set must also as frequent as the superset is, or it may be more frequent.

If the dataset with greater number of items and large number of frequent patterns then the efficiency of Apriori must be suffered of candidate generation. This leads to scan the database in a very large number. Apriori algorithm access the database based on the itemset size and the database must be scanned  $k$  times for the itemsets having  $k$  number of items. So the large number of frequent and large size frequent itemset is really a big problem for Apriori.

For example, we have 10 items. Then in the 5-itemset, the number of candidates will be following.

$${}^nC_r = \frac{n!}{r!(n-r)!} = \frac{10!}{5!(10-5)!} = 252$$

Since based on the above discussion, Apriori Algorithm has the following short comings:

- It is I/O Intensive.
- It is computationally unfeasible.



### 2.3.2 FP-Growth

It was proposed by Jiawei Han, Jian Pei, and Yiwen Yin [5]. This algorithm uses support-based measures to find maximal frequent item sets using I-projected databases [5]. This algorithm compresses a large database into a compact, FP tree structure.

FP-Growth algorithm requires only two database scans for mining frequent /large item sets:

- In the first scan, it finds all frequent item sets.
- In second scan, it constructed 1st FP-tree that contains all frequent information of the given dataset.

In this algorithm, two techniques partitioning and divide and conquer are used for searching, while Apriori uses level wise candidate generation for searching the frequent itemset. The partitioned based technique and divide and conquer improved the searching by reducing the conditional patterns for generation the candidates on the subsequent level. The FP-Tree [6] is also used to reduce the searching of frequent itemset.

### 2.3.3 Éclat Algorithm

This algorithm is same like FP-Growth algorithm. It uses support based measures to frequent item sets. Difference is that it uses different data structures, it uses vertical layout and uses the intersection based approach to compute the support of an item set [6]. The example of Vertical Layout and Horizontal Layout is shown as under:

TID	Items
1	A,B
2	B,C,D
3	A,C,D
4	A,B,D
5	A,C,D

(a) Vertical

A	B	C	D
1	1	2	2
3	2	3	3
4	4	4	4
5			5

(b) Horizontal Layout

Figure 2.1: Vertical and Horizontal Layouts of a transactional database

Éclat is different from Apriori algorithm in pruning step. All the items in the database are recorded in ascending order with respect to the support count [6]. Éclat counts the support of all item sets more efficiently than Apriori algorithm.

Disadvantage of this algorithm is that it generates too many candidate sets to derive frequent item sets at each iteration of algorithm. Another problem is intermediate tid-lists may become too large for memory.

## 2.4 Algorithms for Mining Maximal Frequent Itemsets

Maximal Frequent Itemset discovery is the main functionality of our research problem. So we have studied some research papers and already proposed algorithms for the mining of Maximal Frequent Itemsets. Some of important algorithms are given in the following section.

### 2.4.1 Pincer-Search Algorithm:

Pincer-Search [7] was presented by Dao-I Lin and Zvi M. Kedem in 1997. In this algorithm, the maximal frequent itemsets is found by combining both top-down and bottom-up approaches. The bottom-up approach is as Apriori uses, while the top-down approach is finding the frequent itemset by using MFCS (Maximum Frequent Candidate Generation) and starts the search from  $n$ -itemset followed by  $(n-1)$ -itemset.

In Pincer-Search, the discovery of maximal frequent itemset has a unique method. In this search, top-down approach was introduced and the combined with bottom-up approach. The candidates are generated and are pruned by bottom-up approach as Apriori does. It is also helpful in the candidate generation for Top-Down approach by pruning the infrequent items or itemset. For example a dataset having six items i.e.  $I = \{1, 2, 3, 4, 5, 6\}$  and 1 and 6 are not frequent, it would be pruned from searching MFIS by using top-down approach.

In the top-down approach, the searching for Maximal Frequent Itemset start from  $n$ -itemset and goes by decreasing the itemset size. The candidates are generated and are named as MFCS. In this are itemsets which are frequent is marked as MFIS and infrequent itemsets are pruned. The discovered maximal frequent itemset are useful in bottom-up searching. Because all the subsets of maximal frequent itemset are marked as

frequent itemset as we that all the subsets of MFI are frequent. For example we have an itemset  $T = \{1, 2, 5\}$  as MFI. So  $\{1\}$ ,  $\{2\}$ ,  $\{5\}$ ,  $\{1, 2\}$ ,  $\{1, 5\}$ ,  $\{2, 5\}$  must be frequent. As reversing the Apriori property that if an itemset is not frequent, its superset is also not frequent. By marking the subsets as frequent itemsets, the search space for the bottom-up must be reduced.

Let's take an example of the transactional dataset having six items  $I = \{1, 2, 3, 4, 5, 6\}$ . The figure 2.1 shows the pincer-search for discovering maximal frequent itemset. The Maximal Frequent Itemsets are encircled.

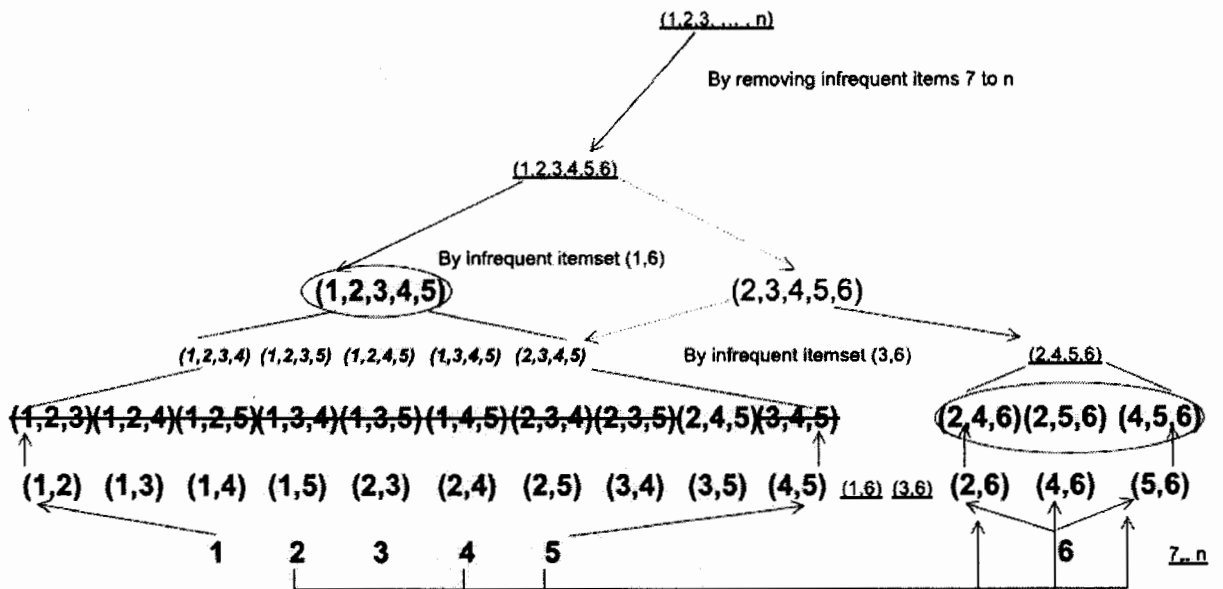


Figure 2.2: Pincer-Search Algorithm for Dataset of six items [7].

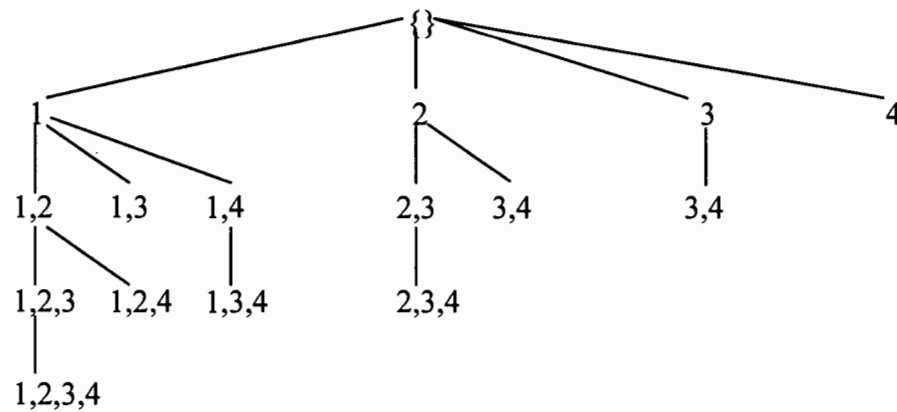
This algorithm has showed great improvement in the search of Maximal Frequent Itemset by adding the top-down approach for the search. But it uses the complex strategy as it has to combine the two approaches. It requires a large search space. It uses many passes to the dataset/database. Although it uses top-down approach and bottom-up approach but uses the candidate generation method for both approaches, so it may be computationally infeasible as it will generate a lot of candidate to be searched for Maximal Frequent Itemset and to be pruned if it is not.

### 2.4.2 Max-Miner Algorithm:

MaxMiner Algorithm was proposed by Bayadro in 1998 also searches for MFIS [8]. MaxMiner uses Rymon's set enumeration framework to order the search space as a tree

[9]. MaxMiner algorithm employs a breath first traversal of the search space. It reduces the database scanning by employing a look ahead pruning strategy based superset frequency.

In Max-Miner, the set enumeration tree was used. In this tree the search space is reduced because of the pruning mechanism. In this the pruning is in such a way that when an itemset is infrequent, then all the sub branches will be pruned. A set enumeration tree for four items  $I = \{1, 2, 3, 4\}$  is shown in the figure 2.4.



**Figure 2.2: Complete Enumeration Tree for Database having four items**

In Max-Miner, each node represent by *candidate group*. Each has two itemsets. For example we have a candidate group  $g$ . The first itemset is called head and denoted by  $h(g)$  which has the items enumerated by node and the second itemset is the tail and denoted by  $t(g)$ . Consider the above example of the dataset having four items. The candidate group  $g$  for itemset  $\{1\}$  is as follows:

$$h(g) = \{1\}$$

$$t(g) = \{2, 3, 4\}$$

The support counting for the candidate  $g$  is the support of itemset  $h(g)$ ,  $h(g) \cup t(g)$  and  $h(g) \cup \{i\}$  for all  $i \in t(g)$ . In this  $h(g) \cup t(g)$  and  $h(g) \cup \{i\}$  are used for pruning. If  $h(g) \cup t(g)$  is frequent, so all the sub-nodes must also be frequent but maximal frequent itemset as the superset is frequent. And if  $h(g) \cup \{i\}$  is infrequent then the sub-node containing  $i$  will also infrequent.

In the worst case MaxMiner does the same number of passes over a database as Apriori does. As uses the bottom-up approach and pruning strategy is based on infrequent itemset, so if the itemsets having less size are frequent then the search space will not be reduced.

### 2.4.3 GenMax Algorithm

GenMax is used to find the Maximal Frequent Itemset [10]. It is *backtrack search* based algorithm. It is an optimized by using new techniques.

First, *progressive focusing* technique is used to eliminate the non-maximal Frequent Itemset. The second technique is the use of *diffset* algorithm for fast frequency checking [8]. The main idea of the diffset is to avoid the storing the entire transaction IDs of each element in the combine set. Instead it stores only the IDs of the itemset which is combined (i.e.  $I_i \cup \{x\}$ ).

Backtrack searching is used for efficiency of finding MFIs. In backtrack search tree, items are selected from *k-itemset* where *k* starts from 1 to *n*. Items other than *k-itemset* put in the candidates sets. In this the combinations will be generated as possible candidates. Here the itemset which is not frequent will be pruned. If possible itemsets are empty then the search will stopped. This will repeat for all items or if the entire candidates are pruned.

Let's take an example of a dataset in table 2.1. The dataset consist of five itemsets i.e.  $I = \{A, C, D, T, W\}$  and six transactions.

TID	Items
1	A, C, T, W
2	C, D, W
3	A, C, T, W
4	A, C, D, W
5	A, C, D, T, W
6	C, D, T

**Table 2.1: An example of a five item dataset**

If the above transactional dataset is mined through GenMax with minimum support 2, it will give the following search tree shown in figure 2.3.

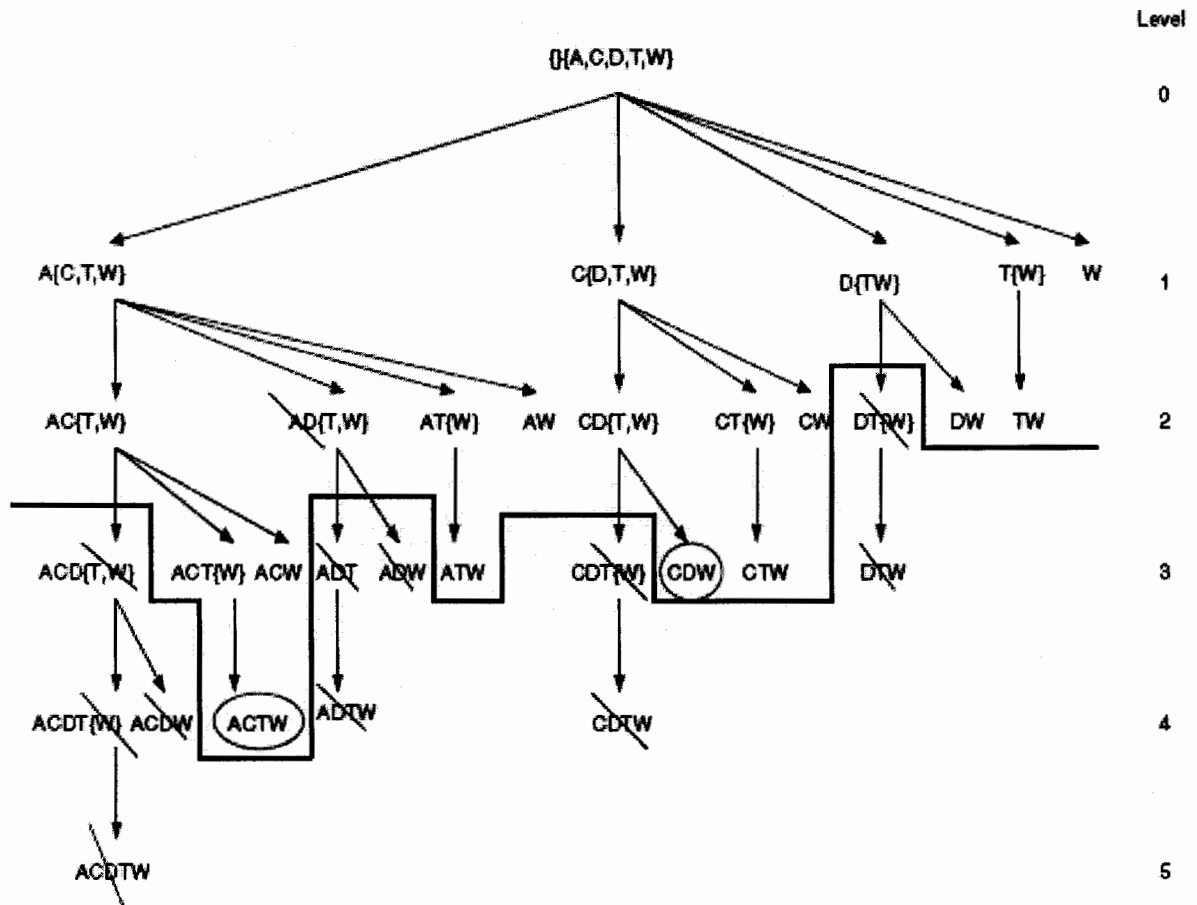


Figure 2.3: Backtrack Search Tree Used for Database of Table 2.1[10]

Although GenMax is optimized by using new techniques but it has also some drawbacks. It follows the incremental approach and hence generates too many Frequent Itemsets on the way to Maximal Frequent Itemset.

#### 2.4.4 MAFIA

MAFIA (Maximal Frequent Itemset Algorithm) is a maximal frequent itemsets mining algorithm [11]. It has been proposed in 2005. MAFIA integrates a depth-first traversal of the itemsets lattice with effective pruning mechanisms. Mafia uses bitmap structure. Bitmap structure is ideal for both candidate itemset generation and support counting.

Some efficient pruning mechanism was used in MAFIA. One of them was named as PEP (Parent Equivalence Pruning). This method is based on the relationship of parent/child nodes. In this method, if the support of the head of node is equal to the union of head and any element in the tail of that node, then the element should be added with its head and the element should be removed from the tail. For example the head of node  $C$  is  $X$  and  $\{y\}$  belong to the tail of the node. So if  $t(X) = t(X \cup \{y\})$  then the head should be  $XU\{y\}$  and  $\{y\}$  should be removed from the tail of the node  $C$ .

FHUT (Frequent Head Union Tail) is another method used in MAFIA. In this if the superset of itemset is frequent, so it would not checked for MFI as well as its sub tree is not checked. Another technique is used which is called HUTMFI. It combines simple method with HUT. HUTMFI does not expand the search tree unlike the FHUT is which explore the sub-node. The superset checking is used here. If superset is not frequent then HUT will be used to check MFI.

The drawback of Mafia is that it mines a superset of the MFI, and requires a post-pruning to eliminate non-maximal patterns. It would be more time for searching.

### 2.4.5 PADS Algorithm

PADS (Pattern-Aware Dynamic Search) was presented by Xinghuo Zeng in 2008 [12]. It is another algorithm used to mine the Maximal Frequent Itemset which uses pattern-aware technique for efficiency. In pattern aware, many search spaces are pruned by future scheduling. It uses those patterns for futures which have already been searched. It is dynamically scheduled and uses to prune the branches which are not frequent.

PADS also avoid the lattice tree of the items like Apriori uses which uses all the frequent items in next level for maximal frequent itemset. PADS uses set enumeration tree like MaxMiner [10]. In which the search space is reduced by creating head and tail of the nodes.

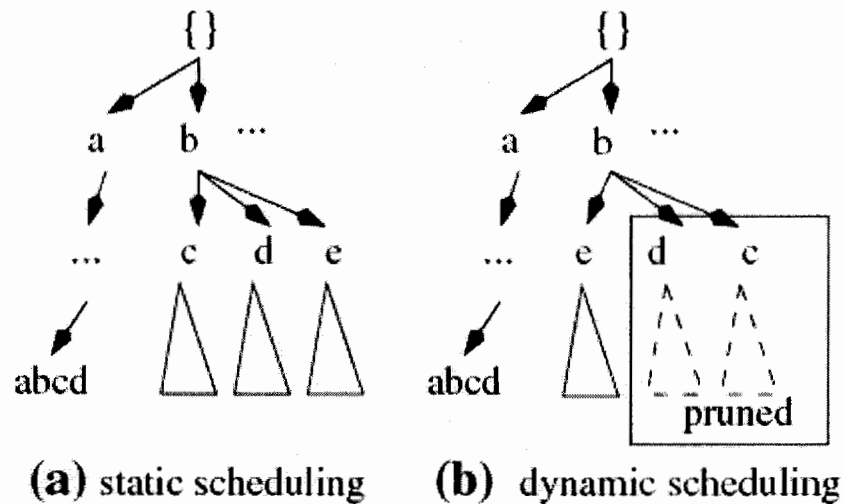
In PADS the set enumeration tree is used more efficiently. In this paper the statistical and dynamic trees was compared. It reduces the search space by eliminating the already found itemset from the tail of the other nodes. Let's see an example of a transactional

database having six items  $I = \{a, b, c, d, e, f\}$  shown in table 2.2 and the comparison of two types of trees are shown in the figure 2.5.

Transaction ID	Items
10	b, c, d, e
20	a, b, c, d
30	a, b, c, d, f
40	a, b, c, d, e
50	d, e, f

**Table 2.2: A transactional Database**

As we see if the  $min\_supp=2$ , then all the items are frequent. So the search tree will be used for all the items in the next level for simple methods used before. For example, if the itemset  $\{a, b, c, d\}$  is frequent. Then using the static tree for searching for  $b$ , all the children  $c, d, e$  will be searched but in the dynamic tree only  $\{b, e\}$  will not be searched because  $\{b, c\}$  and  $\{b, d\}$  are already found as frequent itemsets. The two types of search trees *static scheduling tree* and *dynamic scheduling tree* are shown in the figure 2.5. PADS uses the novel techniques to prune the infrequent branches, but it also uses the candidate generation approach to find the maximal frequent itemset, which requires a large space and high computation.



**Figure 2.5: Comparison of static and dynamic Trees [12]**



PADS algorithm is enhanced with dynamic search but again it depends on the pruning of itemsets. If the items are not pruned then it should search the items in the next steps and there is no advantage of dynamic searching.

#### 2.4.6 MMFI Algorithm

MMFI (Mining Maximal Frequent Itemset) is another algorithm to find MFIS. This was presented by Shiguang Ju and Chen Chen in 2008 [13]. In this algorithm Node by Node (NBN) technique was used by adapted the FP-Tree data structure. There is no superset checking as every itemset is supposed to be checked for Maximal Frequent Itemset. Secondly, there is no need to construction of conditional frequent pattern tree because of recursive NBN traverse. So that it reduced the running time. In MMFI, the FP-Tree method was also used with enhancement. FP-Tree was improved and named as ordered FP-Tree [6].

NBN technique was used to order the node layer by layer and is arranged them in bottom up fashion. Head Table was used to link the layers and nodes belong to same item. In the NBN method the ordered FP-Tree was enhanced by adding to additional field for other information. The *tag* field was added to check whether the itemset is maximal frequent itemset or not and *count* field was used to store the support count.

The FP-Tree and ordered FP-Tree are almost similar. The ordered FP-Tree has four fields. The *item-name* field represents the name of the item. It is often denoted by a number or alphabet. Another field is *ahead*. This field is used to point the next sibling of the tree. Two other fields *tag* and *count* field was discussed above.

The ordered FP-Tree can be created by scanning the database two times. In first step, the database is scanned to generate all frequent itemsets and insert to head table. The head table will store these itemset in descending order [21]. In the second step, the database is scanned and inserts the transactions of frequent items to head table. In this step, the pointer *ahead* is also set to the first child of the nodes and it also set the pointer *next* to next sibling. After inserting the new node, it will be insert in proper order in its respective position.

If we have four frequent items  $I = \{a, b, c, d\}$  in the database. The ordered FP-Tree is shown in figure 2.5. Note that the ordered FP-Tree stores only the frequent items which is generated at first scans as discussed above.

The avoiding of superset checking and construction of conditional pattern tree, MMFI reduced the running time. But, to suppose every itemsets as Maximal Frequent Itemset require intensive computations. This is also required many scans of databases which also complex the algorithm.

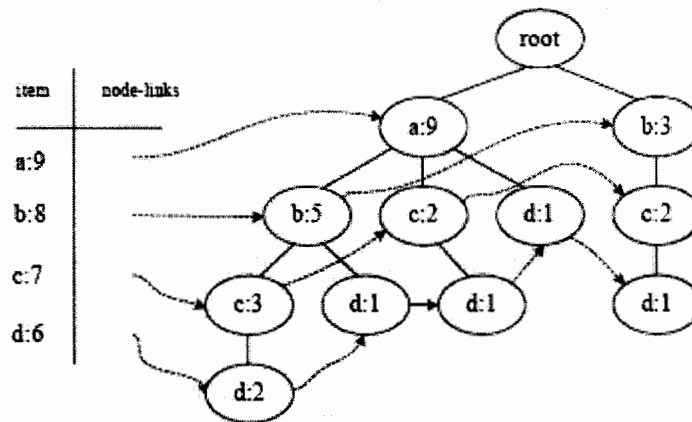


Figure 2.6: The ordered FP-Tree with its support and links [13].

### 2.4.7 HybridMiner Algorithm

HybridMiner is also a recent algorithm presented in 2009 by Shariq Bashir and Rauf Baig [14]. This algorithm is called Hybrid because the mining is done by both Array-Based and Vertical Bitmap layout. Hybrid Database Representation (HDR) represented the items vertically in a header and transactions horizontally. In this algorithm the Maximal Frequent Item is checked by using Header link for every item to connect with other items. This approach optimizes the previous work. Firstly it is a scalable approach for sparse and real dataset, and does not require any extra memory from projection [13].

Although Hybrid is efficient in search space, but it is complex for databases having huge transactions because it combine both approaches by Hybrid Database Representation. It also uses many scans of the database which also indicates that it should be I/O intensive for the database having large items.

## 2.5 Analysis of Previous Work

We have reviewed some algorithm which has been used to search maximal frequent itemset. These are discussed in the previous section 2.3. The previous algorithms are based on bottom-up approach except Pincer-Search [5] which combines both bottom-up and top-down. So they are incremental in nature and uses intermediate results. They use many scans of the databases.

### 2.5.1 Top-Down and Bottom-Up Approach

All the reviewed algorithms are using bottom-up approach for MFI discovery. Pincer- Search [5] combines the top-down and bottom up which also create complexity. All the others use the bottom-up approach and generate intermediates results.

### 2.5.2 Pruning Techniques

Pruning is a very important step in the searching for MFI's because it reduces the candidates and search space for methods. Sometime algorithm based on efficient pruning methods. Apriori [4] uses very simple technique. It says that when an itemset is infrequent then its superset must be infrequent. Pincer Search [5] also follows the same technique for bottom-up and introduce new technique for top-down that when an itemset is frequent, the subset of its must not MFI's which is also the one the property of MFI. GenMax [7] uses progressive focusing technique for pruning the itemsets which are not frequent. It is also very efficient and effective. GenMax uses the depth first search tree and the itemsets which is not frequent, the parent node is also pruned. MAFIA [8] and PADS [9] uses set enumeration tree. MAFIA prunes the itemset from the tree which are infrequent. PADS algorithm improves with dynamic scheduling. But in worst case, its searching is same as Apriori.

### 2.5.3 Scanning the Database/Datasets

Apriori, Pincer Search scans the database as many time as the iterations does not stop. Therefore it needs to scans the datasets so many times. The new techniques GenMax, MaxMiner reduced the number of scanning the database by introduce new methods like frequency focusing technique and diffset methods. MAFIA and PADS reduced even

more by using the using the search enumeration tree and dynamic scheduling. But still it can be improved to scan the just one time to be more efficient.

## 2.6 Problem Statement

Frequent Itemset generation is computationally intensive in nature. The Frequent Itemset production algorithms mostly available are incremental in nature i.e. in order to generate Maximal Frequent Itemset, they generates large intermediate results.

Therefore these algorithms have large size of search space. There is a dire need of such technique which can produce Frequent Itemset and not follow incremental approach. This can be done if we classify the given dataset into number of pattern and then detecting the Maximal Frequent Itemset (Pattern) in top-down fashion which satisfy minimal provided user support threshold. The Frequent Itemsets are basically subsets of Maximal Frequent Itemset. Hence Frequent Itemset can be generated once Maximal Frequent Itemsets are found.

## 2.7 Summary

In this chapter we have analyzed the previous work related to the area. The algorithms used for MFI discovery has been discussed. In these algorithms some problems and drawbacks are discussed. In the last section the problem statement is given which is has been formulated from the reviewing the previous work. In the next chapter the proposed solution will be discussed that what do be done in the research and how can we achieve our objectives.

**Chapter 3**

**PROPOSED  
SOLUTION**

### 3.1 Introduction

After discussing the previous work and formulating the research problem, it is necessary to propose the problem solution. It is also worthy to first identify the issues and problem domain that what to be done. Then solve the issues in a way that how it to done.

Data analysis is the process of looking at and summarizing data with the intent to extract useful information and develop conclusions [14]. Data analysis and Data Mining are almost same things but data mining is done on data set having large number of data. Secondly, data mining does not inference the data and uses as it is originally collected for different purposes. In statistical applications, data analysis divided into two parts, exploratory data analysis and confirmatory data analysis. In exploratory data analysis, the new feathers are discovered in the data and confirmatory data analysis is used to verify some theory or hypothesis.

*Data Mining* is the process of analyzing data and discovers hidden patterns in the large amount of data. Business organizations and financial analyst uses it, but now it is used everywhere. Mostly, scientists use it in their experiments and observational methods. They use data mining for extract enormous patterns in the large data sets.

The data and patterns which are unknown which is useful can be extracted and described by using the data mining tools. The potential information is often helpful in the science. Without using a data mining, the extraction of these pattern is very hard when the size and volume of data is very large.

Data mining method make it easier for statistical and logical analysis which is helpful in decision making.

In data mining Association Rule is a powerful method for extract hidden patterns. In this first we find the frequent itemsets, the itemset which is occurred more frequently in the transactional database [18]. Then the itemset which is frequents can be associated with each other by combining it and find confidence. Confidence is discussed below.

Association rules are required to satisfy a user-specified minimum support and a user-specified minimum confidence at the same time. Support and Confidence are discussed in chapter 1.

## 3.2 Problem Domain:

Our research is to find the Maximal frequent Itemset. Most of existing researches in the maximal frequent itemset is following Bottom-Up approach except Pincer-Search, which uses both Bottom-Up and Top-Down approaches simultaneously. In these techniques the research work shows great efficiency in the finding of Maximal Frequent Itemset from datasets. Most of the researcher is focused on the reducing search space by introducing better pruning techniques [23]. But they are faced some problems in the discovery of Maximal Frequent Itemset because there are large number of intermediate results in the way to find it. In this way, their algorithms required more search space and more computational power. Moreover we meet the following challenges of mining Maximal Frequent Itemset.

- Multiple Scans of database/dataset
- Large number of candidate generation
- Find a support of an Itemset of large number of transactions
- Algorithm Execution Time
- Large Search Space

### 3.2.1 Multiple Scans of the Database/Dataset

The major problem of Association rule mining is the multiple database scans, since first frequent itemsets are searched in the database & then 2-itemset and k-itemsets are created to find similarity between the two large itemsets which requires consulting the database again and again [23]. Moreover, since the frequent itemset generation is also performed on huge databases & large data warehouses, there is chance of multiple Disk I/Os which are the main obstacle in efficiency of database & association rule mining algorithms. Therefore, the primary goal in association rule mining should be to reduce database scans & the disk I/Os.

### 3.2.2 Large Number of Candidate Generation

Creation of candidate itemsets resembles to a chain process i.e. 1-itemsets are used to create 2-itemsets & 2-itemsets are used to create k-itemsets & so on. Hence, the more the size of the candidate itemset [12] [19], the more complicated will the rules be & the more time would it take to execute the algorithm for generating frequent itemsets. Also, since the number of rules will be numerous then, finding interesting rules will be time

TH 8/24

consuming & hard. Moreover, the size & quantity of itemsets also leads to disk & I/O overhead. The number of database scans required by Apriori-based algorithms depends on the size of the largest large itemsets.

### 3.2.3 Finding Support of an Itemset

Finding support for an itemset in a large dataset is also a tedious work. We have to search an itemset in a whole database [3]. Then it would be retrieved the support of the given itemset. As we know that every candidate itemset should be checked that either it is frequent itemset or not. So the frequency is depends upon its support. So the support of every candidate should be calculated by scanning the whole transactions in the dataset/database [24]. We have also discussed that toward the finding of maximal frequent itemset, there are huge number of candidates found in the intermediate results. So we have to find the support of these candidates, which requires a large search space and intensive computation power. So the finding the support of an itemset is also a problem in the finding of the maximal frequent itemsets.

### 3.2.4 Algorithm Execution Time

Another challenge for Association Rule algorithm is the execution time. We discussed the above difficulties and challenges for the discovery of mining the maximal frequent itemset [11].

The development of fast efficient algorithm that can handle the huge volume of data is a challenge for data mining. The good computation performance needs and the performance is almost measure in the terms of time. Time management is the key factor in any algorithm for the fast retrieval of results of the queries especially in today's world where there is huge amount of data & shortage of time [6]. The faster the frequent itemsets are generated, the faster would the process of sales & promotion of products is, since the consequent rules would be generated & reviewed faster. For example, suppose If a clothing store records the purchases of customers, a data-mining system could identify those customers who favor silk shirts over cotton ones. The faster this system would identify such customers before the respective season comes, the more increase will be observed in the retail sales & the process of importing the demanded cloth would be faster & easier.



### 3.2.5 Large Search Space

Another identified problem in the existing algorithm is having large search space for the candidate generation and support count. As we have discussed that mining maximal frequent itemset, the bottom-up approach is used in many of the existing techniques. They have to scan the itemsets which has been generated as candidates [23]. These candidates are often large in number. So to handle it, a huge space is required. Moreover when the size of the transactional database is large, then we have also to compute the support of the candidates and store it the memory.

### 3.3 Proposed Solution

Our proposed scheme is based on “*Maximal frequent itemsets discovery from weighted dataset*”. In order to discover maximal frequent itemsets from datasets number of techniques been stated in literature survey. Now we introduce a novel technique which is unique in the sense that it uses specific form of database and does not utilized uses specific format of data structure and no candidate generation with very lesser memory requirement. The software architecture involves the following steps shown in the following.

#### 1. Preprocessing module:

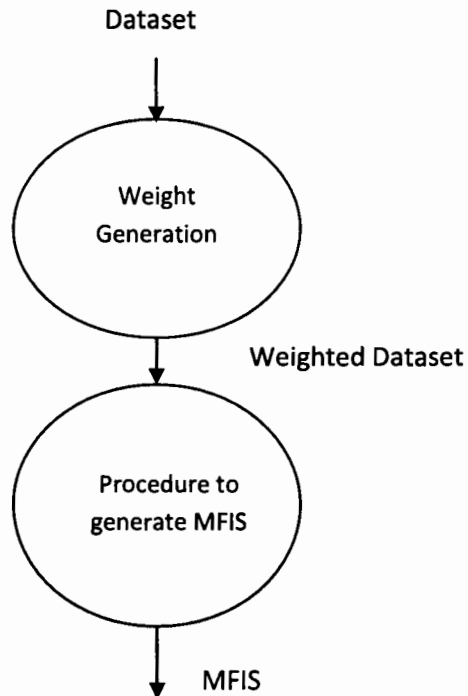
The preprocessing module is the first step module. It takes the original database or dataset as input and uses the WeightGenerator. The WeightGenerator actually transforms the given data set into Weighted Dataset.

#### 2. Frequent Itemset Generation:

This is the major step which generates frequent itemsets. We use a technique which generates frequent itemsets by using top down approach. In top down approach we follow will generate maximal frequent itemsets. Once maximal frequent itemsets are determined it is easy to determine rest of the frequent itemsets. The following figure shows our how these steps can be done.

In our proposed module, we achieve different problems faced in the algorithm of association rule mining for the discovery of maximal frequent itemset. To achieve good runtime performance & efficient running of association rule mining algorithm, the

above mentioned issues should be considered & solutions to these problems must be found to prevent performance degradation. For instance, lessen the number of database scans & reduce huge number of candidate itemsets.



**Figure 3.1: Proposed Framework to Generate Frequent Itemsets**

### 3.3.1 Reduced Database Scans

Multiple database scans is a problem in the algorithms of association rule mining. This creates an overhead on the disk I/O but is also time consuming [26]. Multiple scans are needed in order to create 1-itemset, 2-itemset & k-itemset as well as to keep track of support counts. However, our proposed scheme WeightGenerator performs only a single scan over the whole database while reading transactions list from the binary file (\*.bin) and then stores the count of the transaction in the weighted dataset. Once the count of each transaction is maintained in the weighted dataset, we use a method to count the support of an itemset [23].

Moreover, among various layouts of the database, horizontal & vertical layouts are the most common. Horizontal layout consists of the list of transactions. Each transaction has an identifier followed by list of items. The vertical layout however consists of list of items. Each item has a transaction IDs list- the list of all transactions containing the

item. Therefore, our algorithm uses the vertical layout of the database since this format performs only a single scan while reading the transaction row by row & storing its support count meanwhile in weighted dataset.

### 3.3.2 Reduce the Candidates

Large number of candidate generation is another issue the association algorithms. To find the maximal frequent itemset, a large number of candidates are generated throughout the whole process [20]. These candidates are used to be checked for the MFIS, so the support count is another problem for them.

In our proposed scheme, we follow top-down approach [7], in which the MFIS are found first. In this the number of candidates is numerously reduced by the pruning method. As the candidates are generated, it is pruned by checking the subset of already found MFIS. And know by definition that there is no frequent itemset as superset of an MFIS. So it would be pruned before the checking of support count.

### 3.3.3 Efficient Support Counting Method

In the earlier algorithm or schemes, the support counting for an itemset is also a problematic issue. When a candidate is to be checked whether it is MFI or not, first its support is to be checked and the whole dataset is to be searched for support counting.

We have used an efficient method for support counting in our algorithm. As we discussed that the support of all the combination of the whole dataset is stored in the weighted dataset. So when the support count for an itemset is to be checked, so we check the weighted dataset directly for the given itemset. The support of the supersets of given itemset is also be calculated and added to it, because the itemset is also included in that itemset as well. So by this way the support count for an itemset is calculated very efficiently and in faster way.

### 3.3.4 Reduced Search Space

Search space is another issue in the MFIS discovery in the association rule mining. In the previous algorithms, the bottom-up approach is followed. In this way to find the maximal frequent itemset, a large number of intermediate results are found as frequent itemset [22], these are the candidates for the maximal frequent itemset. So these

itemsets with their relevant information have to be stored. This requires a large search space for the algorithms.

In our algorithm, we use top-down approach. In this way we find the maximal frequent itemset first. So all the subsets are pruned and cannot be a part of candidate itemset. So only the found MFIS are to be stored. By this way, the search space is reduced very efficiently.

### 3.3.5 Execution Time

In today's era, fast and efficient algorithms are demanded as time has become a key element in our lives. Also, since such algorithms are required to handle large amounts of data in data warehouses & perform numerous computations. The faster the algorithm executes, the faster the frequent itemsets are generated. Therefore, the weight generator creates the frequent itemsets faster. Also, the single scan of the database performed by this algorithm contributes to the speed of generation of FIS. The top-down approach is used in our scheme. As the number of candidates reduced, the iterations become less. Therefore the algorithm takes very less time. The efficient method for support count is also helpful in reducing the execution time.

Although the proposed scheme has problem in time complexity but the idea can be implemented further and time may be reduced by new approaches in this idea.

### 3.4 Scope of the Research

Our research idea is in the area of Association Rule and used to find Maximal Frequent Itemset using Top-Down Approach.

The earlier algorithms are using bottom-up approach. These were first finding Frequent Itemset and finally find Maximal Frequent Itemset by the help of that Frequent Itemsets.

In our research project we will use a new technique. First the database/dataset will be converted to weighted dataset and the Maximal Frequent Itemset will be found first and then the Frequent Itemsets will be found from it as all the Frequent Itemsets are the subsets of the Maximal Frequent Itemset.

### 3.5 Summary

This chapter is related to proposed solution to research problem. First we identify some problems regarding the searching of the maximal frequent itemset discovery. These problems are related to execution time, search space, pruning technique. Then we proposed some ideas to overcome the problems in it. In the next chapter we are going to discuss the implementation of proposed idea. In the implementation we will discuss that how the proposed idea is worked in different steps.

## **Chapter 4**

# **IMPLEMENTATION**

## 4.1 Introduction

In this chapter we have discussed the implementation of our proposed scheme. We have discussed the data structure we have used. The tool and techniques are also discussed in the next section. Before going to our algorithm, we have discussed the architecture of our software and also have discussed the different phases and stages of our software in detail.

## 4.2 Data Structure used in our Algorithm

We use top-down approach toward the discovery of Maximal frequent itemset. The searching starts from  $n-1$  to  $1$  itemset, where ' $n$ ' is the total number of items in transactional database. The candidates are generated and these candidates are used for the searching of MFIS. After generating the candidates and checking for the MFI, first it the combinations must be checked for pruning. If all candidates are pruned, the algorithm would stop searching and would be terminated. Otherwise the level is decremented after the MFI checking. The combination for a five items in a dataset is shown in figure 4.1.

											<b>Level 5</b>
			{1,2,3,4,5}								
	{1,2,3,4}	{1,2,3,5}	{1,2,4,5}	{1,3,4,5}	{2,3,4,5}						<b>Level 4</b>
{1,2,3}	{1,2,4}	{1,2,5}	{1,3,4}	{1,3,5}	{1,4,5}	{2,3,4}	{2,3,5}	{2,4,5}	{3,4,5}		<b>Level 3</b>
{1,2}	{1,3}	{1,4}	{1,5}	{2,3}	{2,4}	{2,5}	{3,4}	{3,5}	{4,5}		<b>Level 2</b>
{1}	{2}	{3}	{4}	{5}							<b>Level 1</b>

**Figure 4.1: Itemset Combinations for a five items dataset.**

## 4.3 Tool and Techniques

The research idea has been implemented to check the correctness and efficiency. We have used the following tools to implement our research idea.

### 4.3.1 C++ Language

C++ is used for implementation of the research project. Synthetic dataset is generated from the ARTool [27]. The implementation results are taken from the synthetic dataset. C++ is an object oriented programming language. It has all the functionality used in programming.

### 4.3.2 ARTool

ARTool [27] is java based application which is used for association rule mining. In this tool some basics algorithms and tool are presented like Apriori [4] and FP-Growth [5] which is technique for finding the frequent itemset and then make the associations of the frequent items. ARTool is also used for the generating the synthetic dataset which can be used in experiments of association rule mining.

## 4.4 Software Architecture:

We have designed an algorithm for the implementation of our proposed idea. We worked hard and change it again and again for efficiency. The pseudo code of our main algorithm is shown in figure 4.2.

Input:	Dataset in Binary Format (*.bin)
Output:	Maximal Frequent Itemsets (MFI)
	<ol style="list-style-type: none"> <li>1. Begin</li> <li>2. C= No. of items in the dataset</li> <li>3. R=C-1</li> <li>4. min_supp= minimum support</li> <li>5. MFI= <math>\emptyset</math></li> <li>6. Repeat step 7 to 13 until CMFI = <math>\emptyset</math> or R=1</li> <li>7. CMFI = getComb(R--)</li> <li>8. <math>\xi</math> = pruneCandidate(MFI, CMFI)</li> <li>9. If(<math>\xi = \emptyset</math>)</li> <li>10.       Break;               //Break the loop. All MFI are found.</li> <li>11. For i=0 to length(<math>\xi</math>)</li> <li>12. If(support(equivalent Set(<math>\xi_i</math>) U alternativeSet(<math>\xi_i</math>) &gt;= min_supp)</li> <li>13. MFI = MFI U <math>\xi_i</math></li> </ol>

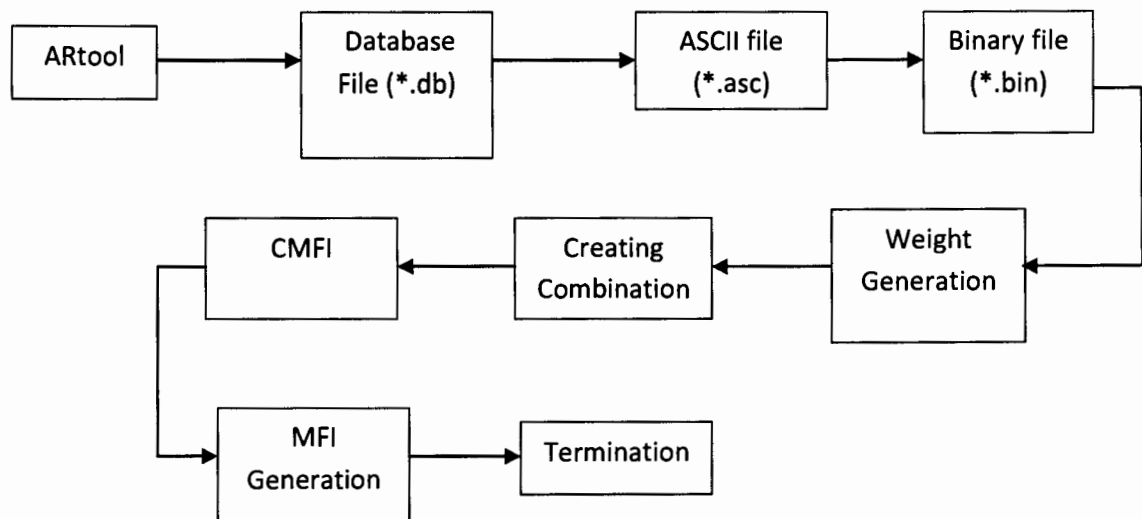
**Figure 4.2: Pseudo Code of the algorithm for MFI from weighted dataset**



The proposed scheme is divided into number phases through which the idea has been implemented. These phases work integrated, some phases are inter related to each other. The following phases have been identified in this research project.

- Generating Dataset by ARTool
- Conversion the dataset
- Weight Generator
- Creating Combination
- Pruning Candidates
- Calculating Alternative Sets
- Checking Support
- Generating MFI's set
- Termination

The following diagram shown in figure 4.1 illustrates the scheme of our project. All the phases are shown in it and the arrow shows the next phase of it. The detail of explanation is given below.



**Figure 4.3: Phases of the Project**

#### 4.4.1 Generating Dataset by ARTool

We have used ARTool [27] utility for generating synthetic dataset for experiments. ARTool is software developed in JAVA. It gives us some facilities. We can generate a synthetic dataset to use in our experiments for verification and validation. In the

ARTool the dataset can be generated with different options. In this we can select the number of items, transaction size. Besides, we can also choose the average size of the transaction, number of patterns and average size of patterns. Two other options correlation and corruption are also given.

ARTool gives dataset in DB format. It cannot be read without ARTool. But we need to read a dataset in C++ because our implementation is in it, which can understand the text format only. So that the DB format must be convert to understandable format for C++. The conversion steps are shown in the following section.

#### 4.4.2 Conversion of Dataset

As we discussed that we are using C++ for the implementation of our research project. We need to read the dataset in a C++ program. For this purpose we need to convert the dataset in ASCII format and then the ASCII format will be convert to binary format. The conversion from DB to ASCII format is done by the ARTool utility, while for conversion the ASCII format to Binary, we have developed a C++ program. The pseudo code of that program is given in the figure 4.4.

#### 4.4.3 ASCII Conversion

After the generating of synthetic database by ARTool, first of all we need to convert it to ASCII format. ARTool provides a utility to convert the DB format to ASCII and vice versa.

For the conversion of DB format to ASCII format, we need the following command using JAVA command line execution.

```
C:\jdk1.3\bin> java db2asc db_file_name asc_file_name
```

This command is used to generate the ASCII file as output which is converted from the DB file. The format of the ASCII is given in the chapter 1. It has a specific format, in which first the ID of the items followed its name. The IDs are the incremental numbers. Then all the transactions are entered in the form of item IDs. These are enclosed in two lines. Before the transaction begins, the line *BEGIN\_DATA* is inserted to indicate that the transactions are started. After all the transactions inserted, *END\_DATA* is inserted to indicate the finishing of transactions.

For example we have a five items dataset having six transactions. It would be displayed as follows in the ASCII file format.

```
1 MAGZINES
2 BOOKS
3 CDs
4 DVDs
5 REGISTERS
BEGIN_DATA
1 2 3
3 4
2 3 4
3 4
1 5
1 3 4 5
END_DATA
```

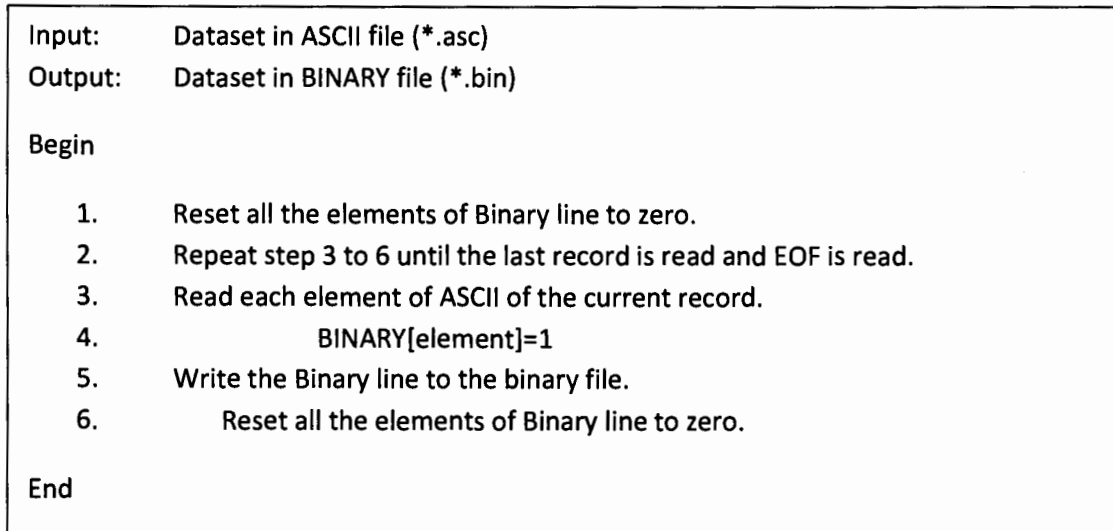
We can also convert the ASCII file back DB format. This can be done the following command.

```
C:\jdk1.3\bin> java asc2db asc_file_name db_file_name
```

The above command uses the ARTool utility *ASC2DB*, takes ASCII file format and converts it to DB format, which is understandable to ARTool. After converting the DB format to ASCII format, C++ program can read it.

#### 4.4.4 Conversion from ASCII to BINARY

The next step in the conversion phase is to convert the ASCII file to BINARY format. The BINARY format is called the bitmap structure of the dataset. In this every transaction is represented in a line of bits. Every line has number of bits equals to the number of items. The transaction is recorded in such a form that the present item is represented by '1' in its own place and absent item is represented by '0'. For this step we have developed a program in C++. The pseudo code is given the figure 4.4.



**Figure 4.4: Pseudo Code for conversion the ASCII dataset to BINARY format**

The program which is shown in the above figure reads the ASCII file and converts it to BINARY format. First it counts the number of items and reserves an array having size as the number of items. Then every line is read by it and the present items are stored in an array. Note that all the elements of the array are initialized to zero. After reading the whole line, it is written to the BINARY file. By this way, all the transactions are read from ASCII file and written to BINARY file. The binary file for the above example is as follows:

```

1 1 1 0 0
0 0 1 1 0
0 1 1 1 0
0 0 1 1 0
1 0 0 0 1
1 0 1 1 1

```

The generated BINARY file is also called bitmap structure of the dataset. It can be used easily by a C++ program to implement an algorithm. We have also used this bitmap structure in our implementation.

#### 4.4.5 Weight Generation

Weight Generation is an important step in our software. In this step all the transactions from binary file is read and recorded to the weighted dataset. Weighted Dataset is the place where the supports of all combinations are recorded with their respective places. Actually it is an array of size  $2^{(\text{No. of Items})}$  for recording the support count for every

combination. For example we have a dataset of 5 items. The size of array for weighted dataset is  $2^5 = 32$  because there are 32 possible combinations in the dataset.

Weighted Dataset is created during the scanning of the database. The transactions are read from binary form of dataset then the combination is converted to decimal and recorded to weighted dataset of that index.

For example we have a dataset of five items  $I = \{1, 2, 3, 4, 5\}$  and a transaction  $T = \{1, 3, 4\}$  so the binary format of  $T$  is  $T_b = \{1\ 0\ 1\ 1\ 0\}$  and this binary combination is equal to 22 in decimals. When this is read, so the entry will be recorded in the weighted dataset as index  $[22-1] = [21]$  will be incremented by 1. By this way all the transactions are scanned and are recorded in the weighted dataset. If the example in bitmap structure of the above section is stored in the weighted dataset. First the size of the array is 32 as  $2^5$ . The following index positions are set and all the rest are zero.

```
weightedArray[5]=2
weightedArray[13]=1
weightedArray[16]=1
weightedArray[22]=1
weightedArray[27]=1
```

Now all the transactions are recorded in the weighted dataset and this array can be used for the counting of support for the itemsets. By using this array the support of an itemset can be calculated very easily and effectively. This mechanism has been discussed in the later section.

#### 4.4.6 Creating Combinations

Creating combinations is another step for creating the candidates of maximal frequent itemset. After reading the dataset from binary file and storing it to the dataset. The next step is to discover the maximal frequent itemsets. For this purpose we use a novel approach and starting the search of maximal frequent itemset in top-down fashion. So the candidates can be generated by using the combinations of the items starting from  $n-1$  and decrementing until the size of itemset reaches to 1.

These combinations are candidates and have to be checked for the maximal frequent itemsets. Note that before the support checking for the candidates, first it is checked that

either it is a subset of the MFIs already found. If so, the candidates will be pruned. This pruning strategy is discussed in the next section. The figure 4.X shows the combinations of a dataset having five items.

#### 4.4.7 Pruning Method

Pruning means to avoid the checking of an itemset for frequent itemset or maximal frequent itemset. This is an important step in the discovery of MFIS. Sometime an algorithm gets efficiency by a rich pruning method. We have also introduced a new technique for pruning the non-maximal frequent itemset. It is simple but very efficient. In our algorithm, the candidates are pruned after the creating of the combinations of the itemsets [26]. Before the checking of support, first the combinations are checked for subset of already found maximal frequent itemset. As we know that an itemset called to be maximal if and only if it has the following two properties:

- The itemset should be frequent, as the support of the itemset is equals to greater than the minimum support threshold.
- There is no superset of the maximal frequent itemset is frequent.

If an itemset is a subset of any MFI already found, so the itemset should be pruned because it is not a maximal frequent itemset [2]. So the combinations are first checked for subset and pruned if a combination is found the subset of an MFI. The MFIs are checked after the pruning steps by checking the support of the itemsets which are not pruned this step. Note that if all the candidates are pruned at some level then the searching for MFI is to be stopped. At that level, all the itemsets of lower size must be the subsets of the already found MFI and cannot be a candidate for the MFI searching. It is discussed in section 4.4.10.

#### 4.4.8 Support Checking

As we have defined that support is the frequency of an itemset. It is calculated for checking that either the itemset is frequent or not [1]. If the support of an itemset is greater than equal to the minimum support threshold, then the itemset is called frequent itemset otherwise the itemset is infrequent. Support checking is also a complex and time consuming method during the algorithms for finding frequent itemsets because the whole dataset is to be searched.

We have used an efficient technique for support counting [18]. Weighted dataset stores all the transactions in an array with the indexes of the combinations of itemset. If we want to check support for a given itemset, we have to calculate the decimal value of that itemset in bitmap structure. The value at the index of the decimal value minus one is actually the frequency of that itemset. But the frequency of supersets of the itemset must also be included to calculate the support. We give specific name alternative sets to that sets in our algorithm. The support of the itemset is the frequency of itemset and alternative sets.

For example we have itemset  $\{3, 4\}$  of the above example of a dataset of five items in earlier in this section. The alternative sets are  $\{1, 3, 4\}$ ,  $\{2, 3, 4\}$ ,  $\{3, 4, 5\}$ ,  $\{1, 2, 3, 4\}$ ,  $\{1, 3, 4, 5\}$ ,  $\{2, 3, 4, 5\}$ ,  $\{1, 2, 3, 4, 5\}$ . The support of the itemset  $\{3, 4\}$  is summation of the frequencies of all these sets.

#### 4.4.9 MFI Generation

The last stage of our algorithm is MFI Generation. This is the stage for which all the previous phases done their work as this our ultimate objective of our algorithm. MFI checking is also a simple work in our algorithm [25]. As we have discussed in the above section that we can count the support of an itemset in the dataset. So if the support is greater or equal to the minimum support threshold then the itemset is called the frequent itemset. We are using top-down approach and pruning the subsets of already found MFI before the support checking [17]. So that the itemset would not only a frequent itemset but it would also a Maximal Frequent Itemset.

As we know that the MFIs can be more than one in a dataset. So we have used a set of MFIs. If an MFI is generated, it would be added to that set.

#### 4.4.10 Termination of Searching

The termination is very important factor because it stops the searching for MFIs and ends the algorithm. Our algorithm ends when either of the two conditions occurs. The conditions are given below.

1. The algorithm stops searching for MFI when all the combinations are pruned by subset checking of MFI. If all the combinations at certain level are the subsets of the

already found MFIs, then no need to search the lower levels. Because all the lower levels combination must be subsets of the MFIs.

2. The second condition is based on size of itemset. The size of itemset cannot be reduced to zero, the zero size means empty set and the empty set cannot be an itemset. If the itemset size become one (if  $R=1$  in the algorithm given in figure 4.2) then the algorithm must be terminated.

## 4.5 Summary

In this chapter we discussed the tool and techniques used in the implementation. The proposed idea is implemented in some steps which work together in a module. We have used combination method for generating the candidates, and then pruned it by subset method. An efficient method used to count the support quickly to find the MFI. After implementation it is necessary to check the results. In the next chapter experimental results are discussed.



**Chapter 5**

**EXPERIMENTAL  
RESULTS**

## 5.1 Introduction

After the completion of implementation, it is important to judge the research work. In this chapter we want to judge our by experiments. There are two sections in the chapter. First we check the correctness of the research work by matching the results with two other algorithms MAFIA [11] and PADS [12]. Then the compare the efficiency of research work with the MAFIA in the time complexity.

## 5.2 Generated Datasets for Experiments

We need datasets with different properties for our experiments. We have used ARTool to generate datasets. Synthetic dataset can be generated with different options provided by ARTool. We have generated five datasets with different options. The datasets with their options are given in the table 5.1.

Dataset	T	AT	I	P	AP
T1000_AT5_I8_P50_AP5.db	1000	5	8	50	5
T2000_AT4_I8_P40_AP5.db	2000	4	8	40	5
T1000_AT5_I10_P60_AP4.db	1000	5	10	60	4
T2000_AT6_I10_P40_AP4.db	2000	6	10	40	4
T1000_AT10_I12_P50_AP5.db	1000	10	12	50	5

**Table 5.1: Synthetic Datasets used in experiments**

The table shows the properties of the dataset. In the first column, the name of the dataset is shown. Here the name is self explaining itself. This is the beauty of ARTool that its name the dataset in such a form that it shows its property. Every dataset can be created with the following properties.

1. ‘**T**’ shows the numbers of transactions in a dataset. For example the first dataset has 1000 transactions, so that the value of T is 1000. We use different size of transactions in different datasets.
2. ‘**AT**’ stands for the average transactions and it shows the average transaction size. We have used different size of transaction in the dataset for our experiment.
3. ‘**I**’ shows the number of items in the datasets.
4. ‘**P**’ shows the number of patterns in the dataset.
5. ‘**AP**’ shows the size of items in the patterns of dataset.

### 5.3 Experimental Results

We have executed our implemented algorithm with the above five datasets. Then we have compared it with two already proposed algorithms MAFIA [11] and PADS [12]. The results are shown in the following sections.

#### 5.3.1 Experimental Results of MFIGen

The above mentioned datasets in table 5.1 is executed for MFI's by our algorithm MFIGen and the following outcomes are recorded.

Dataset	T	AT	I	P	AP	Minimum Support	Found MFI's with Support
T1000_AT5_I8_P50_AP5.db	1000	5	8	50	5	0.3	1 2 4 5 6 7 (320) 1 2 4 5 7 8 (333) 1 2 4 6 8 (303) 1 4 6 7 8 (302) 1 2 3 4 (325) 1 2 3 6 (305) 1 2 3 8 (302) 1 3 4 6 (344) 1 3 4 8 (311) 1 5 6 8 (303) 2 3 4 6 (305) 4 5 6 8 (301) 5 6 7 8 (303) 1 3 7 (307) 3 6 7 (307) 3 6 8 (319) 3 5 (313)
T2000_AT4_I8_P40_AP5.db	2000	4	8	40	5	0.2	1 2 5 6 7 (631) 1 2 5 6 8 (431) 1 2 5 7 8 (434) 1 2 6 7 8 (494) 1 5 6 7 8 (550) 2 5 6 7 8 (422) 1 3 7 (435) 3 5 (432)
T1000_AT5_I10_P60_AP4.db	1000	5	10	60	4	0.2	1 2 3 5 (243) 1 2 4 5 (319) 1 2 4 9 (225) 1 2 4 10 (284) 1 2 5 9 (207) 1 2 5 10 (297) 1 2 9 10 (207) 1 4 5 10 (230) 2 4 5 10 (217) 2 5 9 10 (215) 1 2 7 (239)

							1 4 7 (224) 1 5 7 (246) 2 5 7 (210) 3 4 (203) 7 10 (204)
T2000_AT6_I10_P40_AP4.db	2000	6	10	40	4	0.3	2 4 5 6 7 (616) 2 4 5 6 8 (747) 2 4 5 7 8 (602) 1 2 4 5 (676) 1 4 5 6 (659) 1 4 5 8 (620) 4 6 7 8 (617) 5 6 7 8 (653) 1 2 8 (614) 1 5 7 (663) 1 6 8 (617) 4 9 (601) 5 9 (601) 10 (642)
T1000_AT10_I12_P50_AP5.db	1000	10	12	50	5	0.4	2 7 8 9 10 11 12 (533) 1 8 9 10 11 12 (402) 1 7 9 10 11 (404) 1 7 9 10 12 (405) 1 7 9 11 12 (404) 1 7 10 11 12 (414) 4 9 10 11 12 (421) 1 7 8 10 (404) 1 7 8 11 (402) 1 7 8 12 (400) 3 9 10 11 (410) 3 10 11 12 (415) 4 7 9 11 (401) 4 7 10 11 (413) 4 7 11 12 (400) 1 2 10 (401)

**Table 5.2: Experimental Results of MFIGen**

The table 5.2 shows the results of the proposed algorithm MFIGen. The first six columns shows the properties of the datasets as shown in the tables 5.1 and the next column shows the minimum support to find the MFI's. At last the MFI sets are shown which the ultimate result of the algorithm. The MFI is shown in a line and its support is shown in the bracket. All the five datasets chosen for experiments are executed by MFIGen and the MFI's for each dataset are shown.

### 5.3.2 Experimental Results of MAFIA:

MAFIA [11] is one of the popular algorithms that are used to generate MFI's. We have executed the dataset given in the table 5.1 on the MAFIA. We have found the following results given in table 5.3.

Dataset	T	AT	I	P	AP	Minimum Support	Found MFI's with Support
T1000_AT5_I8_P50_AP5.db	1000	5	8	50	5	0.3	8 6 7 4 1 (302) 3 2 4 1 (325) 3 7 1 (307) 3 6 4 1 (344) 3 6 2 4 (305) 3 6 7 (307) 3 6 2 1 (305) 3 8 2 1 (302) 3 8 4 1 (311) 3 8 6 (319) 8 6 2 4 1 (303) 3 5 (313) 8 5 7 2 4 1 (333) 8 5 6 1 (303) 8 5 6 7 (303) 8 5 6 4 (301) 5 6 7 2 4 1 (320)
T2000_AT4_I8_P40_AP5.db	2000	4	8	40	5	0.2	3 5 (432) 3 1 7 (435) 8 5 6 1 7 (550) 8 2 6 1 7 (494) 8 2 5 1 7 (434) 8 2 5 6 1 (431) 8 2 5 6 7 (422) 2 5 6 1 7 (631)
T1000_AT5_I10_P60_AP4.db	1000	5	10	60	4	0.2	4 5 1 2 (319) 3 5 1 2 (243) 3 4 (203) 9 5 1 2 (207) 7 1 2 (239) 7 5 2 (210) 7 5 1 (246) 7 4 1 (224) 9 4 1 2 (225) 9 10 1 2 (207) 7 10 (204) 4 10 1 2 (284) 9 10 5 2 (215) 4 10 5 2 (217) 4 10 5 1 (230) 10 5 1 2 (297)

T2000_AT6_I10_P40_AP4.db	2000	6	10	40	4	0.3	9 4 (601) 9 5 (601) 10 (642) 1 2 4 5 (676) 1 6 4 5 (659) 7 6 2 4 5 (616) 1 7 5 (663) 1 8 2 (614) 1 8 4 5 (620) 7 8 2 4 5 (602) 7 8 6 4 (617) 7 8 6 5 (653) 1 8 6 (617) 8 6 2 4 5 (747)
T1000_AT10_I12_P50_AP5.db	1000	10	12	50	5	0.4	3 9 11 10 (410) 3 12 11 10 (415) 4 9 12 11 10 (421) 1 7 12 11 10 (414) 1 8 7 10 (404) 1 8 7 11 (402) 1 8 7 12 (400) 1 7 9 12 10 (405) 1 7 9 11 10 (404) 1 7 9 12 11 (404) 1 8 9 12 11 10 (402) 4 7 11 10 (413) 4 7 9 11 (401) 4 7 12 11 (400) 1 2 10 (401) 2 8 7 9 12 11 10 (533)

**Table 5.3: Experimental Results of MAFIA**

The structure of the table 5.3 is same as table 5.2 and it shows the results of experiment for MAFIA [11]. The datasets in the table 5.1 are executed through MAFIA to test the correctness of the proposed algorithm MFIGen. The generated MFI's for each dataset are the same as generated by proposed algorithm. Note that sequence of the items in the sets is different but the items and support values are same.

### 5.3.3 Experimental Results of PADS:

We have compared the results with another algorithm PADS [12], which has been proposed in 2009. As the comparison of results is based on the datasets given in the table 5.1, we have executed the PADS algorithm on that dataset and found the results as the MFI's. These results are shown in the table 5.4.

Dataset	T	AT	I	P	AP	Minimum Support	Found MFI's with Support
T1000_AT5_I8_P50_AP5.db	1000	5	8	50	5	0.3	3 6 8 (319) 3 5 (313) 3 7 1 (307) 3 7 6 (307) 3 2 1 4 (325) 3 2 8 1 (302) 3 2 6 1 (305) 3 2 6 4 (305) 3 4 8 1 (311) 3 4 6 1 (344) 8 7 6 5 (303) 8 2 6 1 4 (303) 8 2 7 1 4 5 (333) 8 4 7 6 1 (302) 8 4 6 5 (301) 8 1 5 6 (303) 5 1 4 2 7 6 (320)
T2000_AT4_I8_P40_AP5.db	2000	4	8	40	5	0.2	3 5 (432) 3 7 1 (435) 8 1 6 5 2 (431) 8 7 2 1 5 (434) 8 7 2 6 5 (422) 8 7 2 6 1 (494) 8 7 5 6 1 (550) 2 7 1 6 5 (631)
T1000_AT5_I10_P60_AP4.db	1000	5	10	60	4	0.2	3 4 (203) 3 5 1 2 (243) 7 1 4 (224) 7 10 (204) 7 2 5 (210) 7 2 1 (239) 7 5 1 (246) 9 2 1 4 (225) 9 5 2 10 (215) 9 5 1 2 (207) 9 10 2 1 (207) 4 1 5 10 (230) 4 2 10 5 (217) 4 2 10 1 (284) 4 2 5 1 (319) 10 2 1 5 (297)

T2000_AT6_I10_P40_AP4.db	2000	6	10	40	4	0.3	9 4 (601) 9 5 (601) 10 (642) 1 5 7 (663) 1 8 6 (617) 1 8 2 (614) 1 8 4 5 (620) 1 6 5 4 (659) 1 2 5 4 (676) 7 4 6 8 (617) 7 2 4 8 5 (602) 7 2 6 4 5 (616) 7 5 6 8 (653) 8 5 4 2 6 (747)
T1000_AT10_I12_P50_AP5.db	1000	10	12	50	5	0.4	3 10 11 9 (410) 3 12 11 10 (415) 1 10 2 (401) 1 8 11 7 (402) 1 8 9 11 10 12 (402) 1 8 12 7 (400) 1 8 10 7 (404) 1 7 12 11 9 (404) 1 7 10 9 12 (405) 1 7 10 9 11 (404) 1 7 10 12 11 (414) 4 11 9 7 (401) 4 12 11 9 10 (421) 4 12 7 11 (400) 4 10 11 7 (413) 2 10 11 12 9 7 8 (533)

**Table 5.4: Experimental Results of PADS**

The datasets are also executed on another algorithm PADS [12]. The results have shown in the table 5.4 in the same manner as earlier two tables. It also shows the MFI sets and support in the last columns. The item's sequence is also not in order as MAFIA generated but the MFI sets are same as generated by MFIGen and MAFIA.

We have used the same datasets and execute our algorithm on datasets as well as two other algorithms MAFIA and PADS. We have used different values for minimum support which is same for every dataset to each algorithm. The output is shown in the above tables for each algorithm. The found MFI's of our algorithm are identical with the results of both MAFIA and PADS. It implies that our proposed algorithm is correct and working properly.



## 5.4 Time Analysis in Different Scenario

To check the efficiency and behavior of the proposed scheme in different scenario and comparison with other algorithms, the proposed scheme has been executed and checked on different dataset.

First in section 5.3.1, we have fixed the number of items and change the transaction size then in section 5.3.2, we fixed the transactions and change the number of items. In section 5.3.3 we have generate some other datasets and execute the proposed scheme and MAFIA [11] to compare in time domain.

### 5.4.1 Increasing the Number of Items:

To check the efficiency in the term of timing, the proposed scheme has been executed on a datasets given in the table 5.5 with 10000 transactions and different number of items (i.e. 10, 20, 30 and 40).

Dataset	T	AT	I	P	AP
T10000_AT5_I10_P50_AP5.db	10000	5	10	50	5
T10000_AT4_I10_P40_AP5.db	10000	12	20	40	10
T10000_AT5_I10_P60_AP4.db	10000	20	30	60	15
T10000_AT6_I10_P40_AP4.db	10000	25	40	40	20

Table 5.5: Synthetic Datasets with fixed transaction and increasing items

The graph in figure 5.1 shows the execution time of above datasets. The graph shows the change in time increase.

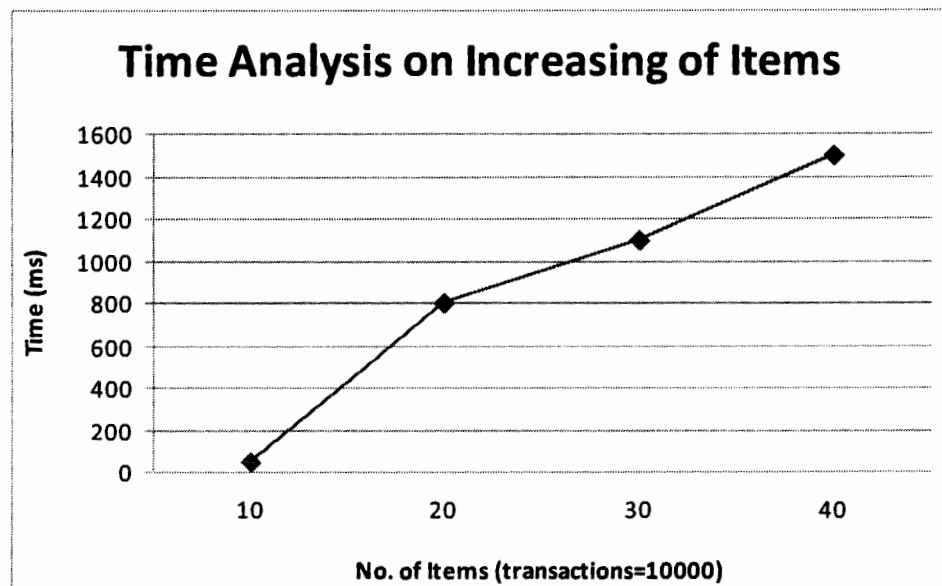


Figure 5.1: Execution time of dataset with increasing of items

In the above the graph has shown for the results of MFIGen on the execution of datasets given table 5.5. In these experiments the proposed algorithm has been checked on the datasets by increasing the number of items. On the increases of items, the running time increases much. It indicates that the algorithm is not feasible for the datasets having large number of items.

### 5.4.2 Increasing the Transactions:

The proposed scheme is executed again on different datasets. This time the number of transactions has been increased from 2000 to 10000.

Dataset	T	AT	I	P	AP
T2000_AT5_I10_P50_AP5.db	2000	5	10	50	5
T4000_AT4_I10_P40_AP5.db	4000	4	10	40	5
T6000_AT5_I10_P60_AP4.db	6000	5	10	60	4
T8000_AT6_I10_P40_AP4.db	8000	6	10	40	4
T10000_AT6_I10_P50_AP5.db	10000	6	10	50	5

Table 5.6: Synthetic Datasets with fixed items and increasing transactions

The below graph in figure 5.2 shows that the number does not effect on running so much because of converting to weighted dataset. So after reading the transactions, all the reading is recorded into weighted dataset and requires no time to calculate support. That's why it do not effect much on the increasing the number of transactions.

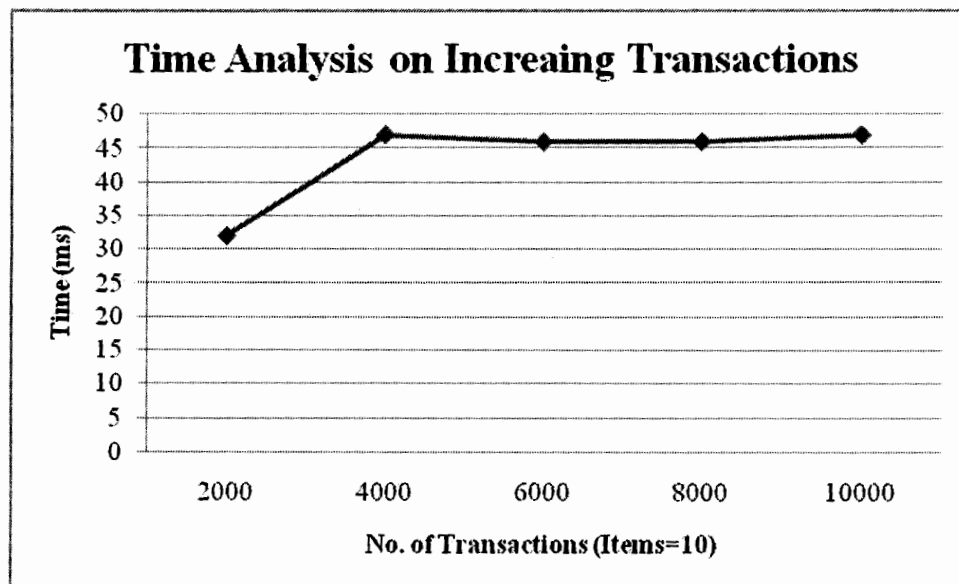


Figure 5.2: Execution time of dataset with increasing the number of transactions

The MFIGen is also checked on the other datasets given in the table 5.6 and behavior of the algorithm is shown in the figure 5.2. In these datasets we increase the number of transactions and checked that on increasing the transactions the algorithm running time increases very slightly as shown by graph in the figure 5.2. The logic behind is the conversion of datasets into weighted dataset which reduces the number of scans and efficient support count.

### 5.4.3 Comparison with MAFIA

The MFIGen (proposed scheme) and MAFIA have been compared by executed on four datasets shown in the table 5.5. The results are found correct and accurate but the execution time taken by the MFIGen is higher than MAFIA as shown in graph below.

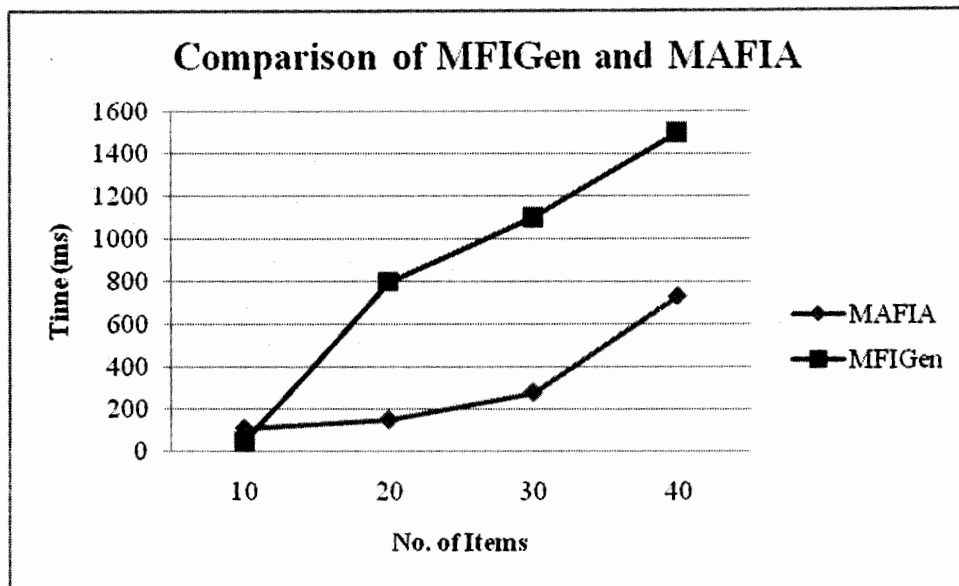


Figure 5.3: Comparison of MFIGen with MAFIA

In the figure 5.3 the MFIGen is compared with MAFIA [11] in the terms of time. The graph shows the behavior of the proposed algorithm MFIGen and MAFIA for execution on the datasets given in the table 5.5. It shows that the proposed algorithm does gives efficient results as MAFIA gives. This will be discussed in the next chapter.

## 5.5 Summary

In this chapter we have discussed about the experiments we have done for the proposed algorithm. First we check the correctness of the algorithm and execute the proposed

algorithm MFIGen on different datasets and record the results. Then the results have been checked on other two algorithms MAFIA and PADS which has been implemented earlier. As the results are same so that the proposed algorithm is correct. Then the behavior of proposed algorithm is also checked by increasing the number of items and transaction. In the last we have compared the proposed algorithm with MAFIA in terms of execution time. After experiment the turn of conclusion is here. In the next chapter we conclude our research work and its future works.

## **Chapter 6**

# **CONCLUSION & FUTURE WORK**

## 6.1 Introduction

After experiments given in the previous chapter to analyze the proposed idea, in this chapter we discuss the summary and conclusion of the research work and suggest some future works to implement the idea in other ways with modification. This chapter also includes the implementation of association rule in the real life.

## 6.2 Application of Association Rule

Frequent Patterns mining has many applications in real life. Today the era of information technology is implemented everywhere and people take benefits of it.

The association rule mining mostly implemented in the market basket analysis as we have described in chapter 1. We know that competition in trades is very much in the modern world. Every organization wants to analyze the customer behavior and makes policies and decisions in the lights of that analysis. Association rule mining is one of great analysis tool for market basket analysis which identifies the frequent itemset and association with other itemsets.

Another application of the association rule is in the field of scientific research and experiments. Scientific experiments are usually based on large amount of data. Data mining and specially association rule is useful in the analysis of the data and finding frequent patterns.

Medical research and laboratory tests also use the data mining technique in different ways. The frequent patterns play an important role in medical field. Many diseases can be diagnosed through these patterns. Association rule uses to identify frequent patterns. These pattern can also be compared with other parameters like age, location etc for analysis. These parameters can be associated with the help of association rule mining.

Data mining techniques are also applied in the image processing. Association rule mining is useful in the image segmentation. In image segmentation, different contents of image are identified. These contents can be analyzed through association rule for finding hidden patterns in the images. As we know that image processing has a great application in the medical fields like MRI. So data mining has its own techniques which are helpful in the extracting of hidden patterns in diseases.

### 6.3 Conclusion

The objective of this research work to implement a novel approach for finding the Maximal Frequent Itemset. A new approach is used for the discovery of MFI's which is used in top-down fashion. The purpose of this approach is to avoid the generation of intermediate results toward the discovery of MFI's.

In this research work, we have successfully implemented our new idea. The idea has been implemented in an algorithm. To check the correctness of the new algorithm, we have compared it with two other algorithms MAFIA [11] and PADS [12]. We have executed the algorithms on different dataset. The result of these algorithms is found identical. It shows the correctness of our algorithm.

Besides the correctness, we have also compared it with MAFIA and the running time is shown which has been discussed in the section 5.3.3 and has been shown in figure 5.3. As it has been discussed that the proposed idea is not efficient in time domain and the execution time of the algorithm is more than the previous algorithm execution time. But keeping in view the time limit and scope of the research, it not possible to work again and implement the in another way which may be better than the previous algorithm like MAFIA [11] and PADS [12].

The proposed idea is based on top-down approach, in which the itemset is to be checked for MFI first and then the frequent itemsets to be find by subset methods. The searching of MFI starts from larger itemsets towards the smaller itemsets. First the candidates are to be checked for pruning. The candidate is pruned if it is the subset of already found MFI. The pruning of candidates avoids the support checking for all the pruned candidates.

In the pruning step, if all the candidates are pruned at some level then the algorithm should be stopped. Because when all the candidates are the subset at a level then all the candidates in the lower level must be the subset of the found MFI's.

In this way, the algorithm is really good for the datasets which are dense in nature and having large size MFI's. So that at a very higher level, the algorithm will be stopped

and searching for MFI's would be stopped. Secondly it would be good in the discovery for the top-k frequent itemset [21], because it is known that up to which level the algorithm should run and find the MFI's.

## 6.4 Future Work

The proposed technique has correct but it does not give efficient results. Therefore we suggest that this technique can be implemented with modification to achieve efficient results. This technique can be implemented for MFI discovery without using user given support threshold [16].

As we have discussed in section 6.1 that the proposed idea is good enough to find the MFI's in dense datasets and will give efficient results. So that this idea must be keep in view for future implementation of the proposed idea. We have also discussed in the previous section that the top-k frequent itemsets [21] have a fixed number of iteration and are at top level searching, so that the proposed idea can also be implemented for discovery of top-k frequent itemsets.

## 6.5 Summary

In this chapter we conclude the research work by describing the implementation of the research in the real life and giving suggestion for the future work to be done in this area of research. It has been noticed that the proposed idea does not gives efficient results in some scenario so that we suggest the this idea can be implemented in different ways like similarities based itemset mining of frequent itemset [16]. It can also be implemented for the searching of largest maximal frequent itemsets and top-k itemsets.



**Appendix A**

**REFERENCES**

## References

- [1] D. Hand, H. Mannila and P. Smyth, “Principles of Data Mining”, MIT Press 2001
- [2] J. Han and M. Kamber, “Data Mining Concepts and Techniques”, Morgan Kaufmann Publishers, 2<sup>nd</sup> Edition, 2006
- [3] J. Han, P.S. Yu, “Data Mining: an overview from Database perspective”, IEEE Transactions on Knowledge and Data Engineering, 1996.
- [4] R. Agrawal, T. Imielinski, and A. Swami, “Mining Association Rules between Sets of Items in Large Databases,” ACM SIGMOD Conference Management of Data, 1993
- [5] J. Han, J. Pei and Y. Yin, “Mining Frequent Patterns without Candidate Generation,” Proc. ACM-SIGMOD Int'l Conf. Management of Data, 2000
- [6] J. Hipp, U. Guntzer and Gholamreza “Algorithms for Association Rule Mining – A General Survey and Comparison”, SIGKDD Explorations, 2000
- [7] Dao-I Lin and Zvi M. Kedem, “Pincer-Search: A New Algorithm for Discovering the Maximum Frequent Set”, Springer conference on Advances in Database Technologies-EDBT, 1998
- [8] R. Bayadro, “Efficiently mining long patterns from databases”, ACM-SIGMOD Conference on management of Data, 1998
- [9] S. Kotsiantis and D Kanellopoulos “Association Rules Mining: A Recent Overview”, GESTS International Transactions on Computer Science and Engineering, 2006
- [10] K. Gouda and J. Zaki, “GenMax: An Efficient Algorithm for Mining Maximal Frequent Itemsets”, Data Mining and Knowledge Discovery, 2005
- [11] D. Burdick and M. Calimlim, “MAFIA: A Maximal Frequent Itemset Algorithm”, IEEE Transactions on Knowledge and Data Engineering, 2005
- [12] X. Zeng, “PADS: a simple yet effective pattern-aware dynamic search method for fast maximal frequent pattern mining”, Springer conference on Knowledge Information System, 2008
- [13] S. Ju and C. Chen, “MMFI: an Effective Algorithm for Mining Maximal Frequent Itemsets”, International Symposiums on Information Processing, 2008
- [14] S. Bashir and R. Baig, “HybridMiner: Mining Maximal Frequent Itemsets Using Hybrid Database Representation Approach”, 10<sup>th</sup> IEEE-INMIC conference, 2009
- [15] [http://en.wikipedia.org/wiki/Web\\_mining](http://en.wikipedia.org/wiki/Web_mining)
- [16] Saif-ur-Rahman and Dr. Sikandar Hayat, “Similarity Based Mining for Finding Frequent Itemsets”, 8<sup>th</sup> International Conference on Computers, Communications and Systems 2007

- [17] Haiwei Jin “A Counting Mining Algorithm of Maximum Frequent Itemset based Matrix”, 7<sup>th</sup> International Conference on Fuzzy Systems and Knowledge Discovery, 2010
- [18] Guizhen Yang, “The Complexity of Mining Maximal Frequent Itemsets and Maximal Frequent Patterns”, KDD, 2004
- [19] Yuan Li and Jun Li, “Mining Algorithm of Maximal Frequent Itemsets Based on Position Lattice”, IEEE International Conference on Granular Computing, 2010
- [20] Z. Zheng and R. Kohavi, “Real World Performance of Association Rule Algorithms”, KDD, 2010
- [21] Hua-Fu Li, “Mining top-k maximal reference sequences from streaming web click-sequences with a damped sliding window”, Expert Systems with Applications, 2009
- [22] Yue Chen and Jiankui Guo, “Incremental Mining of Sequential Patterns Using prefix tree”, Hi-tech research and development program of China, 2007
- [23] Mi S. Gu and Jeong H. Hwang “Weighted-FP-Tree Based XML Query Pattern Mining”, the Regional Core Research Program by National Research Foundation of Korea, 2010
- [24] R.C.Agrwal and C.C.Aggarwal, V.V.V.Prasad, “Depth First Generation of long Patterns”, In Proceedings of the ACM SIGKDD Conference, 2000
- [25] Tianming Hu and Sam Y. Sung “Discovery of maximum length frequent itemsets”, Journal of Information Sciences by Science Direct, 2008
- [26] Geetha M. and R.J.D Souza, “A Dynamic Approach for Discovering Maximal Frequent itemsets”, International Conference on Computer Engineering and Technology, 2009
- [27] <http://www.cs.umb.edu/~laur/ARtool>

