Finding Maximal Frequent Itemsets Using Similarity Measure

Developed by:

Fazal Ahmad

378-FBAS/MSCS/F07

Supervisor:

Mr. Muhammad Imran Saeed

Co-Supervisor:

Mr. Saif Ur Rehman

Department of Computer Science & Software Engineering

Faculty of Basic & Applied Sciences

International Islamic University, Islamabad

(2012)

1 - Application Programming

2 - Software Engineering

Department of computer Science & Software Engineering,

International Islamic University Islamabad

FINAL APPROVAL

Dated: 11-06-2012

Certified that we have examined the project report titled ***"Finding Maximal Frequent Itemset Using Similarity Measure"***, submitted by ***Mr. Fazal Ahmad, Registration No: 378-FBAS/MSCS/F07***. In our judgment this research project is of sufficient standard to warrant its acceptance by the International Islamic University, Islamabad for the award of the degree of MS in Computer Science.

Committee:

External Examiner
Dr. Abdus Sattar,
Former D.G.,
Pakistan Computer Bureau,
H-NO: 143, St:60, I-8/3,
Islamabad.

Internal Examiner
Prof. Dr. Muhammad Sher,
Chairperson,
Department of Computer Science,
International Islamic University,
Islamabad.

Supervisor
Muhammad Imran Saeed,
Assistant Professor,
Department of Computer Science & Software Engineering,
International Islamic University,
Islamabad.

Co-Supervisor
Mr. Saif Ur Rehman
Lecturer, Institute of Information Technology (IIT),
Kohat University of Science & Technology (KUST),
Kohat.

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

DEDICATION

Dedicated to my loving parents

&

Honorable teachers.

DECLARATION

I hereby declare that this research work "***Finding Maximal Frequent Itemsets Using Similarity Measure***" neither as a whole nor as a part has been copied from any source. Furthermore it is declared that the research project and the thesis report are entirely developed by my personal efforts under the supervision and sincere guidelines of ***Mr. Muhammad Imran Saeed*** and ***Mr. Saif Ur Rehman.*** In addition no part of this research work has been submitted to this or any other university for the award of degree or qualification.

Fazal Ahmad

378-FBAS/MSCS/F07

ACKNOWLEDGEMENTS

All praises and much gratitude to Almighty Allah, the most merciful and glorious, who granted me the potential to work hard and perseverance to accomplish this research work.

I would like to sprinkle special thanks on my co-supervisor ***Mr. Saif Ur Rehman,*** who always provided me greatest support and help whenever I needed throughout my research work. He was readily available for consultation, shared his knowledge with me as well as motivated me for successful completion of this research work.

I am very thankful to my loving supervisor ***Mr. Muhammad Imran Saeed*** who not only provided me support in this research work but his kind cooperation and encouragement throughout my degree is always memorable.

I cannot forget the successful support of my affectionate parents, who always show desire and pray for my success as well as provided me financial and every other kind of support throughout my life.

I would like to thank my honorable teachers, friends specially ***Mr. Kamran Ullah*** and all those who helped me during this research project.

Fazal Ahmad

378-FBAS/MSCS/F07

PROJECT IN BRIEF

PROJECT TITLE:	***Finding Maximal Frequent Itemsets Using Similarity Measure***
UNIVERSITY:	Department of Computer Science & Software Engineering, International Islamic University, Islamabad.
UNDERTAKEN BY:	Fazal Ahmad (378-FBAS/MSCS/F07)
SUPERVISED BY:	Mr. Muhammad Imran Saeed, Assistant Professor, Department of Computer Science & Software Engineering, International Islamic University, Islamabad.
CO-SUPERVISOR	Mr. Saif Ur Rehman, Lecturer in Institute of Information Technology (IIT), Kohat University of Science & Technology (KUST).
TOOLS USED:	C++ for development of project. ARTool for generating dataset. MS Office Word 2007 for documentation.
OPERATING SYSTEM:	Windows 7 (64-bit).
SYSTEM USED:	DELL STUDIO 1555 Intel Core 2 Due 2.20 GHz 2.20 GHz RAM 3.00 GB.
START DATE:	June 2010.
COMPLETION DATE:	February 2012

ABSTRACT

Maximal Frequent Itemset (MFI's) generation is less computationally intensive than Frequent Itemsets. Once MFI's are determined FIS can be deduced from them easily. Proposed technique presents a novel technique to discover MFI's of specific size very easily than the other established existing techniques. Proposed technique finds similarity among every two items. Then the similarity of the items of combination size is calculated from the two items similarity without generating the intermediate results. The itemset having maximum similarity is the most frequent maximal frequent itemset. The proposed technique will produce results more efficiently than MAFIA.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1
INTRODUCTION

1 INTRODUCTION

From the last few decades there exists huge amount of data in every field of the world. This huge availability of data and rapid increase in data require processing, analysis and storage. There are interesting patterns in the data which are unknown to the user. Extracting these patterns manually from huge amount of data is a difficult task, therefore to discover interesting hidden patterns automatically and to analyze and process huge amount of data is possible by the process of data mining.

Data mining is the combination of two words, "Data" and "Mining".

Facts and figures in the raw form are known as data. Whereas when the data is processed it is converted into meaningful information. For example facts such as cloud, moisture etc results into the weather broadcasting. The historical background of information can be used for future decision making process, which is known as knowledge. Such as collecting information about consumer buying behavior results into making best marketing strategy to increase the sale of a particular product. Another query could be what is the buying habit of consumers? What is the financial position of consumers of the product? Thus on the basis of these information the manufacturer or retailer could determine which product required special offer and display etc.

Different definitions of data mining are as under:

Data mining is the process of extracting knowledge from large amount of data [1]. Data mining digs out valuable, non-trivial information from large multidimensional, apparently unrelated databases (sets) [2].

Data mining is the process to sort large amount of data and get your related information. This information is normally used by business intelligence organizations and financial analysts but increasingly being used in the science to extract information from the enormous datasets generated by modern experimental and observational methods [3].

The process of data mining is possible by the use of sophisticated algorithms to uncover meaningful patterns and correlations that are hidden. For this purpose there exist many algorithms but one will concentrate on those which are more efficient.

1.1 DATA MINING TECHNIQUES

There are various techniques through which the data mining process extracts hidden patterns to discover knowledge. Here we will discuss some of these methodologies.

1.1.1 CLASSIFICATION AND PREDICTION

Classification is a data mining technique in which different classes are modeled and for a particular data object whose class label is unknown, a class is predicted.

In classification different classes are predefined and a data object must belong to a particular class. No new class can be defined for a new data object but the data object can be mapped to the relevant class. Consider a business organization; products can be classified based on per day profit. Profit can be classified as high profit and low profit. A high profit product can be categorized as class "A" product and a low profit product can be categorized as class "B" product. If a new product is introduced it will belong to either class "A" or class "B" based on the profit of the product.

Classification is often referred to as supervised learning because the classes are predetermined before examining the data. Classes are defined based on data attribute values with the help of classification algorithms [1].

In real world the classification problem involves more dimensions and therefore more complex to classify. For example consider the bank loan decision problem. It involves at least two dimensions "age" and "income". If age dimension is youth then loan decision is risky. If income is high then loan decision is safe. If age is middle aged and income is low then loan decision is risky [1]

1.1.1.1 PREPARING THE DATA FOR CLASSIFICATION AND PREDICTION

Before the data can be classified it is required that the data must be prepared for classification. The data can be prepared for classification with the help of following techniques [1].

- **DATA CLEANING**

Data cleaning is required to prepare data for classification and prediction. The noise deviate the general behavior of data therefore it is required to be removed or at least must be reduced. The missing values will be replaced with most probable values. For example consider analyze the AllElectronics sales and customer data. Many tuples may have not recorded values for certain attributes such as customer income. Fill it by means of a constant label such as "unknown" or by average income of customers.

- **RELEVANCE ANALYSIS**

Many attributes in the data may be redundant or irrelevant. These attributes slow down the data mining task. Therefore in classification it is required that if two attributes in data are redundant then one of it will be removed. If an attribute is irrelevant, that attribute will not be considered in classification process. For example consider the resulting classes the attribute birth-place and residence included. Birth-place is removed in the subsequent analysis.

- **DATA TRANSFORMATION AND REDUCTION**

The value for a given attribute will be scaled to a small range. For example consider an attribute say income. The range of income may be from 10000 to 100000 per month. So by the process of normalization it will be scaled down from -1.0 to 1.0 or 0.0 to 1.0.

1.1.2 CLUSTER ANALYSIS

Cluster analysis is different from classification because in cluster analysis classes are not known. Clustering also referred to as unsupervised partitioning because the classes are not known in advance. In cluster analysis data objects are grouped based on their similarities. Data objects in the same group are very similar as compared to data objects of other groups.

1.1.2.1 CATEGORIZATION OF MAJOR CLUSTERING METHODS

There exist various algorithms for different clustering methods in the literature. Clustering methods are divided into the following broad categories [1]:

- **PARTITIONING METHODS**

Consider a database of n data objects. The partitioning method constructs K partitions of data such that the number of partitions will not exceed the number of data objects.

Initially the partitioning method constructs K partitions then the objects move from one partition to another by a technique called iterative relocation technique, in order to improve the quality of partitioning. Each partition is referred to as a cluster. Data objects in the same cluster are very similar or related as compared to the data objects of other clusters. The two most popular partitioning methods are k-means and k-medoids algorithms [1].

- **HIERARCHICAL METHODS**

The hierarchical clustering creates a tree like structure of clusters. Hierarchical method may either be agglomerative (bottom-up) or divisive (top-down). Agglomerative approach merges the data objects until all the data objects are in one group or until the given conditions are satisfied. In divisive approach the data objects are split until there is one data object in each group [1].

Consider four data objects {w, x, y, z}. In the following diagram we will explain both agglomerative and divisive hierarchical clustering approaches on these four data objects.

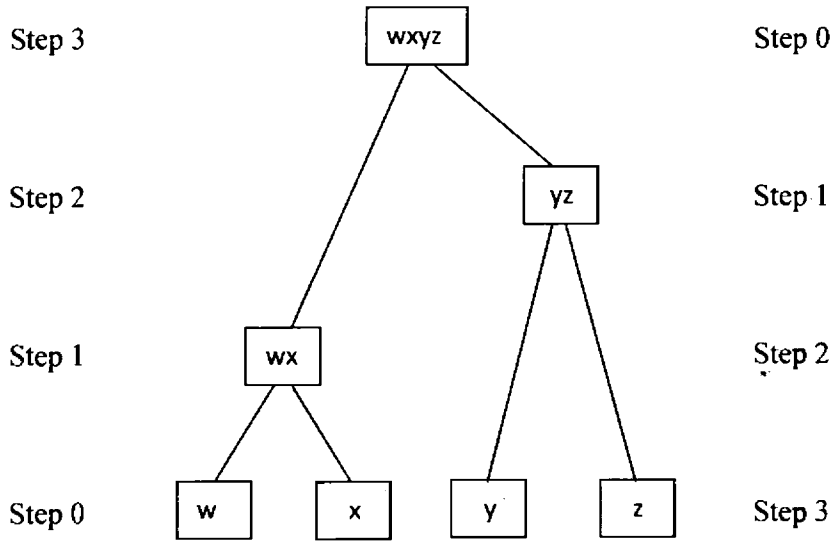

Figure 1.1 Hierarchical Clustering Approaches.

The advantage of hierarchical methods is that it is less expensive but once a step is taken it cannot be roll backed. Therefore once erroneous decisions are taken it cannot be corrected.

- **DENSITY-BASED METHODS**

Density-based clustering methods are developed to discover clusters of arbitrary shapes. In density-based methods the dense region of data objects are separated by sparse region. The sparse region can be considered as noise therefore the density-based clustering can also be used to remove noise or for outlier analysis [1].

Typical examples of density-based clustering methods are DBSCAN and OPTICS.

- **GRID-BASED METHODS**

In grid-based clustering methods the data objects are quantized into a finite number of cells in order to form a structure called grid structure. Then all clustering operations are performed on this structure. One of the important advantages of this approach is that its processing time is fast [1].

Examples of this approach are STING, WaveCluster and CLIQUE etc.

STING explores statistical information stored in the grid cells. WaveCluster clusters objects using a wavelet transform method. CLIQUE represents a grid and density-based approach for clustering in high dimensional data space.

- **MODEL-BASED METHODS**

In this approach a model is hypothesized for each of the clusters. Then the given data is fit into the model which is more relevant.

EM, COMWEB, and SOM are algorithms for this approach.

1.1.3 OUTLIER ANALYSIS

Some data objects in the database deviate from general behavior of data. These data objects are called outliers. In data mining process these outliers are considered as noise or exceptions and can be thrown away, however in some applications these are very interesting patterns can be used for important purposes such as fraud detection etc.

Consider students admission section of a university. If a candidate's qualification is less than the required qualification for a particular program, it can be recorded as outlier.

There are four methods for outlier detection:

The statistical approach, the distance-based approach, the density-based local outlier approach, and the deviation based approach [1].

Here we will not go into the details of these approaches.

1.1.4 ASSOCIATION RULE MINING

Association means relationship. In data mining association means hidden relationship among data items in data sets. Thus association rules mining uncover data items that are statistically related in the underlying data.

Data items that occur more frequently together in a given data set are called frequent itemset. Associations among items in large data set (transactional or relational) are discovered by means of frequent itemset mining. Many industries collect and store

heavy amount of data continuously therefore they are becoming interested to mine such type of patterns from their databases.

Market basket analysis is a typical and most interesting example of frequent itemset mining. Different customers visit to the supermarket and each has its own needs and desires as well as financial status. This process analyzes buying habits of customers by finding associations between different items that customer put in their shopping baskets. The discovery of such type of associations can provide help to retailers in developing marketing strategies by gaining insight that which items are purchased together frequently by customers. Consider association between pepsi and chips. If customers are buying pepsi, how likely are they to also buy chips (and what kind of chips) on the same trip to the supermarket?

{Pepsi} ⟶ {Chips}

This shows a strong association between pepsi and chips. If a customer visits to the supermarket to buy pepsi there are more chances that he will also buy chips on the same trip. Such information increases sales of retailers by helping them to do selective marketing and planning their shelf space [1].

Market basket analysis helps retailers in designing different store layouts. One strategy is that items which are frequently purchased together can be placed closed so that the sale of such items can be further increased. If customer who purchases pepsi tends to also buy chips at the same time, then placing pepsi display close to the chips display help in increasing the sale of both the items. Similarly consider association between milk and bread. If customer who purchases milk tends to also buy bread at the same time, then placing the milk display close to the bread display. Alternative strategy is that placing pepsi display opposite to the chips display or placing milk display opposite to the bread display of the store motivates the customers who purchase one item to pick up the second item along the way [1].

Association rule mining is a two step process given bellow.

1.1.4.1 FREQUENT ITEMSET GENERATION

When two or more items occurring more frequently in a given data are called frequent itemset. An itemset is frequent if it satisfies a minimum support threshold. Support of an itemset means that how many transactions in the database contain the itemset.

Support can be defined as:

"The percentage of transactions from a transaction database that the given rule satisfies" [1].

If we have two items A and B in an itemset then their support is given as;

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

1.1.4.2 ASSOCIATION RULE GENERATION

When the frequent itemset is generated, it is quite easy to generate association rule. Association rule can be generated by using confidence measure. A rule is strong association if it satisfies minimum support threshold and minimum confidence threshold.

Confidence can be defined as:

"The conditional probability P (Y|X), that is, the probability that a transaction containing X also contains Y" [1].

Consider two items A and B, the rule A=>B has confidence "c" and can be calculated as;

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

Once frequent itemset generated, it is straight forward to construct association rule from it. Therefore our in-depth concentration is on frequent itemset generation.

1.2 EXISTING TECHNIQUES

Many algorithms exist to find Frequent Itemset and Maximal frequent itemset. Each algorithm has its own pros and cons. Here we will discuss some of these techniques.

1.2.1 APRIORI ALGORITHM

Apriori algorithm is proposed by R. Agrawal and R. Srikant in 1993 [4]. Apriori is one of the basic algorithms to find the frequent itemset in a simple way. Apriori is a breadth first search algorithm that is used to prune the itemsets which are not frequent. This algorithm scans the data base many times.

Apriori Algorithm suffers from the following shortcomings:

- It may need to generate a huge number of candidate itemsets which results in storage wastage.
- Due to many database scans and candidate generation it results in low efficiency.
- It is computationally unfeasible.
- It is I/O Intensive.

1.2.2 MAX-MINER ALGORITHM

Max-Miner Algorithm was proposed by Bayadro (1998) also searches for MFIS [5]. Max-Miner uses Rymon's set enumeration framework to order the search space as a tree [6]. Max-Miner algorithm employs a breadth first traversal of the search space. It reduces the database scanning by employing a look ahead pruning strategy based on superset frequency.

Max-Miner has the following drawbacks:

- As Max-Miner requires only one transaction at a time in memory, therefore it is I/O Intensive.
- Scans the Database more than two times.
- It uses clever lower bound techniques to determine whether an itemset is frequent without accessing the database and actually counting its support. Hence, in this case too the method is not able to exactly determine the support of all frequent itemsets [7].

1.2.3 GENMAX ALGORITHM

GenMax is used to mine the Maximal Frequent Itemset [8]. It is backtrack search based algorithm. It is 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. $I \cup \{x\}$).

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

1.2.4 SB-MINER

SB-Miner is an approach to mine the frequent itemsets. This technique utilizes the set enumeration tree and inclusion exclusion principle to correctly discover the frequent item sets [9]. This technique works on apriori property and performs single scan of the data sets. It suffers from a few drawbacks given below.

1) Node size of the set enumeration tree is large because every node is storing tid's lists.

2) Time to find FI's is quite large because of operation performed on the long tid's list is computationally intensive.

1.2.5 HBMFI

Hash based Maximal Frequent Itemsets was proposed by A.M.J Zubair Rahman and P. Balasubramanie in 2008 [10]. This algorithm uses vertical data format for storing the transactions in the database and uses hash data structure to represent this data format [10].

This algorithm is very efficient because:

- Pruning does not require after finding FI completely, but can be done while finding MFIs.
- At each level, after computation of FI, we are computing MFI also. So, the time taken to compute MFI is negligible [10].

Besides its advantages HBMFI is computationally intensive.

1.2.6 BOMO ALGORITHM

BOMO algorithm is a frequent pattern-growth (FP-growth) based approach and known as the currently best algorithm in mining N-most interesting itemsets category [11]. BOMO uses a compact frequent pattern-tree (FP-tree) to store compressed information about frequent itemsets. FP-growth is a depth first search based approach, and does not rely on candidate generation-and-test mechanism and achieves impressive results in frequent itemset mining problems.

1.2.7 N-MOSTMINER AND TOP-K-MINER ALGORITHM

In this paper they present novel efficient algorithms (N-MostMiner and Top-K-Miner) [12] using Bit-vector dataset representation approach. The major advantage of using Bit-vector dataset representation approach in our algorithms is that, it optimizes the itemset frequency counting cost. This paper also presents a novel bit-vector projection technique which we named as projected-bit-regions (PBR). The main advantage of using PBR in N-MostMiner and Top-K-Miner is that, it consumes a very small processing cost and memory space for projection.

1.3 PROBLEMS FACED BY FIM ALGORITHMS

All the frequent itemset mining algorithms share the following problems.

1.3.1 USER THRESHOLD SUPPORT VALUE

In order to find out the required frequent patterns users run the algorithm with different support threshold. Hence hit and trial methods are used to discover interesting frequent patterns. Incorrect settings may cause an algorithm to fail in finding the true patterns [13]. Perhaps more insidious problem is that we may find patterns that do not exist [14].

1.3.2 INCREMENTAL IN NATURE

All frequent itemsets mining algorithms are incremental in nature. In order to produce frequent itemsets of larger size these algorithms use information from immediate previous computational step.

1.3.3 COMPUTATIONAL TIME

Non expert users required ample amount of time to find the intended patterns and results in too many patterns with the FIM algorithm, because users are not data mining experts or domain expert. Mining generally becomes inefficient or, often, simply unfeasible [15]. Expert users require little amount of computation time to find the interesting patterns as they are domain experts.

1.4 SCOPE OF THE PROJECT

Association rule mining is a very effective and useful research area. Association rule mining consists of the following two steps:

- Finding frequent itemsets.
- Association rules generation from frequent itemsets.

Once frequent itemset generated, it is straight forward to construct association rule from it. Later on the paradigm has been shifted from FIS to MFIS. Maximal Frequent Itemset generation is less computational intensive than FIS. Therefore our in-depth concentration is on maximal frequent itemset generation. All the MFIS generation techniques use support measure. In order to compute MFIS these algorithms need user provided support threshold as primary parameter among other parameters. Now if the user sets high support value lesser number of MFIS is discovered. On the other hand if user provides low support threshold value large number of MFIS is computed. In case of lesser MFIS it is quite possible that user's desired pattern is dropped because of very high support threshold value. To our knowledge no such technique exists so far which does not requires any threshold value from the data miner.

CHAPTER 2
LITERATURE SURVEY

2 LITERATURE SURVEY

Frequent itemsets generation can be handy in consumer market basket analysis, network intrusion detection and analysis of web page access logs and in many other data mining areas. Apriori and all post Apriori solutions will eventuate in a large number of rules. But now the recent paradigm has shifted towards techniques which can produce fewer FI's by having enough information encoded into them to produce association rules. These variants of FIS are called Maximal frequent itemsets (**Maximal** FIS) [16], [17]; maximum length frequent itemsets (**Maximum** LFIS) [18] and constraint based methods [19], [20], [21], [22], [23], [24], [25]. We will discuss the following techniques to find FIS in our literature survey.

2.1 PAPERS STUDIED AND THEIR LIMITATIONS

We have studied many papers in our literature survey. In the following section we discuss only those which are more relevant to our research work and their limitations.

2.1.1 MINING ASSOCIATION RULES BETWEEN SETS OF ITEMS IN LARGE DATABASES

This research paper represents apriori algorithm proposed by R. Agrawal [4] in 1993. Author represents that algorithm incorporate buffer management and pruning techniques. This algorithm finds frequent itemsets by using support and confidence measure. It scans the database many times and follows breadth first searching technique.

The data mining problem is decomposed into two sub problems. Combinations of items that are above minimum support threshold are called large itemsets and that are below minimum support threshold are called small itemsets.

If memory is low during a pass this algorithm puts some data on disk in order to utilize memory. The data on the disk will be considered in the next pass in order to avoid missing important data.

This algorithm follows iterative approach and candidate k itemsets is used to find candidate k+1 itemsets. To explain how apriori algorithm works consider transactional database:

- First of all scan the database to find candidate 1-itemset denoted by C1. Then find support count for each item of the candidate itemset by calculating that how many times each item exists. Now compare the support count of each item of the candidate 1-itemset (C1) with minimum support threshold let minimum support threshold is 3. If the calculated support of an item is less than the minimum support threshold then the item will be pruned away else add the item to the list of frequent 1-itemset denoted by L1. In order to understand easily and systematically store the items in L1 in ascending order.

- In order to find frequent 2-itemset, first find candidate 2-itemset denoted C2 by joining L1 with itself in such a way that each item will be joined with each and every item that comes next it. Calculate the number of occurrences of each itemset and compare it with minimum support threshold. If calculated support is less than the minimum support threshold prune it else add to frequent 2-itemset denoted by L2.

- Similarly find C3 by joining L2 with itself in the same way as discussed above. For efficiency purposes and removing headache of long processing the algorithm use apriori property [1]:

  "If any of the subset of an itemset is not frequent the itemset will be pruned away".

- now calculate support count for remaining items of candidate group, compare it with minimum support count, prune away items that have support less than minimum support threshold and add remaining items to frequent 3-itemset L3.

- Repeat the process until candidate itemset results in empty itemset.

- The above whole process of apriori algorithm is given in the table given below.

Table 2.1 Apriori Mining Process

(a) Original Database

TID	List of Item_IDs
T100	I1, I2, I5
T200	I2, I4
T300	I2, I3
T400	I1, I2, I4
T500	I1, I3
T600	I2, I5
T700	I1, I3
T800	I1, I2, I3, I5
T900	I1, I2, I3
T1000	I1, I2, I5, I6

(b) C1 (Candidate 1-Itemset)

Itemset	Calculated Support
[I1]	7
[I2]	8
[I3]	5
[I4]	2
[I5]	4
[I6]	1

(c) L1 (1-FIS)

Large 1-Itemsets
[I1]
[I2]
[I3]
[I5]

(d) C2 (Candidate 2-Itemset)

Itemset	Calculated Support
[I1,I2]	5
[I1,I3]	4
[I1,I5]	3
[I2,I3]	3
[I2,I5]	4
[I3,I5]	1

(e) L2 (2-FIS)

Large 2-Itemsets
[I1,I2]
[I1,I3]
[I1,I5]
[I2,I3]
[I2,I5]

(f) C3 (Candidate 3-Itemset)

Itemset	Calculated Support
[I1,I2,I3]	2
[I1,I2,I5]	3

(g) L3 (3-FIS)

Large 3-Itemsets
[I1,I2,I5]

SHORTCOMINGS OF APRIORI ALGORITHM

Apriori Algorithm suffers from the following shortcomings:

- It may need to generate a huge number of candidate itemsets which results in storage wastage.
- Due to many database scans and candidate generation it results in low efficiency.
- It is computationally unfeasible.
- It is I/O Intensive.

2.1.2 NEW ALGORITHMS FOR FAST DISCOVERY OF ASSOCIATION RULES

This research paper is presented by M. J. Zaki, S. Parthasarathy, M. Ogihara and W. Li [26] in 1997. The algorithms of this research paper use vertical database layout and scan the database only once. In order to mine maximal frequent itemset, the techniques used in this research paper first approximate the maximal frequent itemsets. For rough approximation two clustering approaches are used i, e, Equivalence Class Clustering and Maximal Hyper graph Clique Clustering. The latter approach is more precise than the previous one. Bottom-up and hybrid traversal approaches are used to discover true maximal frequent itemsets from these potential maximal frequent itemsets. The hybrid approach provides more accurate results than the bottom-up approach.

SHORTCOMINGS

The available algorithm is quite simple and provides an order of magnitude performance improvement over the previous techniques but it is quite expensive because

of finding tid lists by means of intersection process. Also an extra amount of storage is required.

2.1.3 EFFICIENTLY MINING LONG PATTERNS FROM DATABASES

This research paper explains Max-Miner algorithm proposed by Roberto J. Bayardo Jr [5] in 1998. This algorithm is developed to overcome the short comings of Apriori like algorithms for the discovery of maximal frequent itemsets. An itemset is maximal frequent if it is frequent and has no frequent super set. For mining long patterns Max-Miner provides a minimum two order of performance improvement over Apriori algorithm. Max-Miner follows bottom-up traversal and using look ahead pruning strategy. Max-Miner can be implemented by using set enumeration tree and follows breadth first search.

Unlike Apriori, Max-Miner algorithm uses superset frequency pruning in addition to subset infrequency pruning. If any of the subset of an itemset is infrequent the itemset will also be infrequent and can be pruned away as Apriori does. If a superset of an itemset is frequent the itemset will be discarded to reduce the search space because it will not be maximal. Max-Miner uses item reordering in order to increase the effectiveness of superset frequency pruning. Support lower bounding play an important role in increasing efficiency of superset frequency pruning.

SHORTCOMINGS OF MAX-MINER

Max-Miner has the following drawbacks:

- As Max-Miner requires only one transaction at a time in memory, therefore it is I/O Intensive.
- Scans the Database more than two times.
- It uses clever lower bound techniques to determine whether an itemset is frequent without accessing the database and actually counting its support. Hence, in this case too the method is not able to exactly determine the support of all frequent itemsets [7].

2.1.4 MEMORY ISSUES IN FREQUENT ITEMSET MINING

This research paper was proposed by Bart Goethals [27] in 2004. It represents Medic, an efficient algorithm for frequent itemsets mining with low memory requirements. Medic is proposed to remove the drawbacks of FP-growth and Eclat algorithm. Both FP-growth and Eclat require the entire database to be in main memory. Sometime there is not enough memory to store all the transactions of database therefore affect the efficiency and effectiveness of the algorithm. Like FP-growth and Eclat, Medic also uses depth first search. Medic sorts the items of the database in ascending order of support count. The items that are not satisfying minimum support threshold are deleted from the search space. Medic store the cover of only those items that occurring in current read transactions. The items that are not required long will be removed from memory.

SHORTCOMINGS

The techniques work well for sparse databases however not effective for dense databases.

2.1.5 MAFIA: A MAXIMAL FREQUENT ITEMSET ALGORITHM

MAFIA algorithm is developed by Doug Burdick, Manuel Calimlim, Jason Flannick, and Johannes Gehrke [28]. They first create algorithm with depth first traversal without pruning mechanism but there was no improvement over breadth first traversal in regard of search space, therefore they adopt various pruning strategies. In parent equivalence pruning (PEP) they compare the head and tail items. In FHUT they adopt superset pruning mechanism. As the head and tail union of a node is the largest itemset at that node, if it is frequent the subtree initiated at that node will be pruned away. An enhanced version of FHUT is HUTMFI, unlike FHUT there is no need to explore leftmost branch of subtree. HUTMFI is more advantageous than FHUT in regard of memory requirement because HUTMFI calculate lesser number of itemsets as compared to FHUT.

The effectiveness of pruning mechanism can be increased by storing the items in increasing order of support count instead of lexicographic ordering. This may result in sufficient utilization of the storage at the left side of the tree.

MAFIA algorithm can also find frequent itemsets and frequent closed itemsets by slightly changing in pruning mechanism but we will not go in details because this algorithm is basically designed for maximal frequent itemsets.

MAFIA algorithm uses vertical bitmaps for database representation. If an itemset is present in a transaction the bit value for that itemset will be set to one in that transaction. If the itemset is not present, the bit value is set to zero.

SHORTCOMINGS OF MAFIA ALGORITHM

One problem is that if there are more zeros in a transaction, useless operations will be performed.

For performance enhancement MAFIA uses compression techniques. The zero will be ignored in subsequent transactions.

2.1.6 GENMAX: AN EFFICIENT ALGORITHM FOR MINING MAXIMAL FREQUENT ITEMSES

GenMax algorithm is proposed by K. Gouda and Muhammad J. Zaki [8] in 2005. In this algorithm backtrack search technique is used to find maximal frequent itemsets. It follows depth first traversal and performs pruning at the time of mining exact maximal frequent itemset instead of performing pre-pruning and post pruning steps. If a superset of a frequent itemset is found to be frequent, the existing itemset will be deleted from the search space. For performance improvement the itemsets are reordered in increasing order of support count.

For maximality checking GenMax uses progressive focusing technique. In progressive focusing technique a list of local maximal frequent itemsets is generated which limit the search of maximal supersets to the list of local maximal frequent itemsets. This technique is very helpful for dense datasets.

So for different algorithms use either horizontal or vertical data format for frequency testing. Vertical data format has some advantages over horizontal data format.

Vertical format is simple and faster in computing the support of itemsets. Instead vertical format is more versatile than horizontal format. However the intersection time of vertical data format is very large. Besides the intermediate results for finding frequent patterns is large to fit in memory. GenMax uses a new format called diffset to remove shortcomings of vertical data format. Diffset stores the differences of transaction ID's instead of storing the entire Tidset.

Consider two TID's A and B its difference is given as;

$d(AB) = t(A) - t(B)$ and

Support of (AB) is given as;

$ó(AB) = ó(A) - |d(AB)|$.

SHORTCOMINGS OF GENMAX

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

2.1.7 SIMILARITY BASED MINING FOR FINDING FREQUENT ITEMSETS

This research paper is presented by Sikandar Hayat Khiyal, Saif Ur Rahman, Dawlat khan and Abdus Salam [9] in 2007. In this research paper they proposed an algorithm called SB-Miner algorithm. SB-Miner is an approach to mine the frequent itemsets. This technique utilizes the set enumeration tree and inclusion exclusion principle to correctly discover the frequent item sets [9]. This technique works on apriori property and performs single scan of the data sets. The itemsets in set enumeration tree is arranged in such a way that every child node is superset of its parent node. In this research paper they make use of vertical layout for clustering related data and mine frequent itemsets on the basis of similarity measure.

As the set enumeration tree nodes store level wise frequent itemsets, in level 2 only those itemsets of level 1 can be linked whose similarity satisfies minimum similarity threshold. Similarly in level 3 only those itemsets of level 2 can be linked and will be

frequent that satisfies minimum similarity threshold and so forth. Similarity can be calculated with the help of union and intersection operations.

DRAWBACKS OF SB MINER

SB-Miner algorithm suffers from a few drawbacks given below:

1) Node size of the set enumeration tree is large because every node is storing tid's lists.

2) Time to find FI's is quite large because of operation performed on the long tid's list is computationally intensive.

2.1.8 AN EFFICIENT ALGORITHM FOR MINING MAXIMAL FREQUENT ITEM SETS

Hash based Maximal Frequent Itemsets was proposed by A.M.J. Md Zubair Rahman and P. Balasubramanie [10] in 2008. Their main objective is to mine maximal frequent itemsets efficiently. They use a special kind of data structure called Hash data structure to represent data in vertical format. Initially two hashes are maintained one for itemset and another for tidset. The support threshold is specified and now in the second level the permutation of only those itemsets will be considered and put in hash data structure for itemsets that satisfy minimum support threshold i.e. frequent. Now the intersection of tidset of each permutation is taken and only those permutations will be considered for next level that are frequent taking into consideration maximality checking and so forth. The time to find maximal frequent itemsets is inversely proportional to the support threshold.

At the end they compare their algorithm with MAFIA and experimental result showed that HBMFI is 2 to 3 times faster than MAFIA. The reason for efficiency of this algorithm is that:

- Pruning does not require after finding FI completely, but can be done while finding MFIs.
- At each level, after computation of FI, computing MFI also. So, the time taken to compute MFI is negligible [10].

SHORTCOMINGS

Besides its advantages, the HBMFI is computationally intensive.

2.1.9 PADS: A SIMPLE YET EFFECTIVE PATTERN AWARE DYNAMIC SEARCH METHOD FOR FAST MAXIMAL FREQUENT PATTERN MINING

PADS is another efficient algorithm for mining maximal frequent patterns by Xinghou Zeng, Jian Pei, Ke Wang, Jinyan Li [29] in 2008. In this algorithm they modify depth first search by incorporating latest approach called pattern-aware dynamic search. Most of the search space is pruned away by having knowledge about prior known maximal frequent patterns. If a pattern is found to be maximal the remaining sub tree will be arranged in such a way that the patterns intended to be maximal will be scheduled to be find earlier and the subsets of already known maximal patterns will be arranged at the last to be pruned away.

2.1.10 MINING TOP-K FREQUENT PATTERNS WITHOUT MINUMUM SUPPORT THRESHOLD

This research paper is written by A. Salam, M. Sikandar Hayat Khayal [30] in 2011. In this research paper they represent an efficient method for mining a little amount of top maximal patterns. Most of the algorithms for mining frequent patterns use minimum support threshold as a primary parameter and scans the database many times which results in performance degradation. The technique used in this research paper is support free and scans the database only once. Also the algorithm used in this research paper uses top-down approach. In support based technique if minimum support value is incorrect may not produce required patterns and erroneous patterns may be reported therefore the author is motivated of using support free technique.

2.2 PROBLEM STATEMENT

All the MFIS generation technique use support measure. In order to compute MFIS these algorithms need users provided support threshold as primary parameter

among other parameters. Now if the user sets high support value lesser number of MFIS are discovered. In case of lesser MFIS it is quite possible that user's desired patterns are dropped because of high support threshold value. To our knowledge no such technique exists so far which does not requires any threshold value from the data miner and which uses similarity measure to compute MFIS.

CHAPTER 3
PROPOSED SOLUTION AND ARCHITECTURE

3 PROPOSED SOLUTION AND ARCHITECTURE

In this chapter we will discuss the architectural design of the software to show how the proposed technique will work. The proposed technique will work in the following three phases.

3.1 INITIALIZATION PHASE

In this phase the data set is scanned only once. During this phase similarity of every item with other item of the data set is created. This similarity of itemsets along the corresponding items is stored in two dimensionally array.

Algorithm for initialization phase is shown in figure 3.1 below;

Algorithm for Module 1

Input

D: *database of transactions*

Output

F^2: *Frequent itemset of size 2*

1) *S=Readfile_Makelist (D)*
2) F^2*=find2FIS (S)*
3) *EXIT*

Figure 3.1: 2FIS Generator

The algorithm takes the data set "D" as input parameter. The output of the algorithm is frequent itemset of size 2. Algorithm will work in three steps. Step 1 of the algorithm creates a linked list. The linked list has nodes linked by pointers. Each node has a tag field and an info field. The tag field contains the actual item which is to be compared with other items for similarity. The info field contains the transaction id's list of those transactions in which the item in tag field is present. The pointer links the nodes

of the linked list and points to the next node. A model linked list created by 1st step of the algorithm is given in the following diagram.

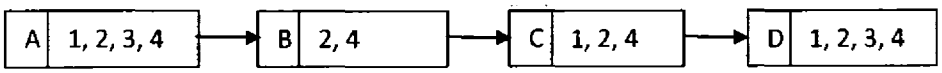

Figure 3.2: linked List Representation

The 2nd step of the algorithm will generate frequent 2 itemset of the items of the linked list by finding similarity of the itemsets and put it in 2 dimensional arrays.

The 3rd step performs termination of the algorithm.

EXAMPLE 3.1:

Consider a data set of four items A, B, C and D with transaction ID's 1, 2, 3, 4.

Table 3.1 Dataset of four items for initialization phase

TID	A	B	C	D
1	1	0	1	1
2	1	1	1	1
3	1	0	0	1
4	1	1	1	1

A = {1, 2, 3, 4}, B = {2, 4},

C = {1, 2, 4}, D = {1, 2, 3, 4}

A∩B = {2. 4}, AUB = {1, 2, 3, 4}

Sim $_{AB}$ = |A∩B| / |AUB| = 2/4 = 0.5

A∩C = {1, 2, 4}, AUC = {1, 2, 3, 4}

Sim $_{AC}$ = |A∩C| / |AUC| = 3/4 = 0.75

A∩D = {1, 2, 3, 4}, AUD = {1, 2, 3, 4}

Sim $_{AD}$ = |A∩D| / |AUD| = 4/4 = 1

B∩C = {2, 4}, BUC = {1, 2, 4}

Sim $_{BC}$ = |B∩C| / |BUC| = 2/3 = 0.67

B∩D = {2, 4}, BUD = {1, 2, 3, 4}

Sim $_{BD}$ = |B∩D| / |BUD| = 2/4 = 0.5

C∩D = {1, 2, 4}, CUD = {1, 2, 3, 4}

Sim $_{CD}$ = |C∩D| / |CUD| = 3/4 = 0.75

COMBINATION NO	1ST ITEM	2ND ITEM	SIMILARITY
1	A	B	0.5
2	A	C	0.75
3	A	D	1
4	B	C	0.67
5	B	D	0.5
6	C	D	0.75

Figure 3.3: Two Dimensional Arrays

3.2 2 FREQUENT ITEMSET SORTING USING QUICK SORT

This is second phase in proposed technique. In this phase every frequent-2-itemset (calculated in previous step) is sorted in descending order according to their similarity by means of quick sort.

The required algorithm for the second phase is given below;

Algorithm for second phase

Input

A: *Two dimensional arrays.*

Output

A_s: *sorted arrays.*

1) *Select top-most similarity.element as pivot. Define two variables m, n to scan the array use variable m as top pointer, and n as down pointer to scan the array*
2) *Scan array from down-to-upward, by decreasing n, until a key greater than or equal to pivot is encountered*
3) *Scan array from top-to-downward, by increasing m, until key smaller than or equal to pivot is encountered*
4) *Swap the keys identified by m and n, swap 2nd column.item, swap 3rd column.item*
5) *Move pointers to next cells (increase m by 1, and decrease n by 1)*
6) *Repeat step# 2 through step# 5 until m<n*
7) *Partition array at m=n*
8) *Repeat step# 1 through step#7 until each partition results into single row*

Figure 3.4: Quick sort algorithm

Consider 2 dimensional array of figure 3.3 of example 3.1. The sorting will be performed on the basis of similarity.

The top most key 0.5 is chosen as pivot. The variables "m" and "n" are used to scan the array. The top pointer m is positioned at cell 1 and pointer n is positioned at cell 6 of the array. Scanning is started from down to up. Since key 0.75 is larger than the pivot 0.5, the up scanning is stopped. The scanning from top to down is started. Since the key in cell 1 is equal to the pivot, the top scanning is stopped. The keys 0.5 and 0.75 in cell 1 and 6 respectively are swapped. Similarly the items of 2nd and 3rd columns are also swapped. The top pointer is moved to next down cell 2, and the down pointer is moved to next up cell 5. The down to up scanning is started. Since key 0.5 is equal to the pivot the down to top scanning is stopped. The top to down scanning is started. Since the key 0.75 at cell 2 is greater than pivot 0.5, the top pointer is moved to the next down cell 3. The key at cell 3 is 1 which is greater than pivot; therefore the top pointer is moved to next down cell 4. The key at cell 4 is 0.67 which is greater than pivot; therefore the top pointer "m" is moved to next down cell 5. Now the pointers "m" and "n" point to the same cell, the partitioning procedure is terminated and results into the following two partitions.

1	C	D	0.75
2	A	C	0.75
3	A	D	1
4	B	C	0.67
5	B	D	0.5

Figure 3.5: Top partition

6	A	B	0.5

Figure 3.6: Down partition

Now the down partition is a single row therefore the same procedure will be applied only on top partition recursively.

3.3 MAXIMAL FREQUENT ITEMSET GENERATION

In this phase maximal frequent itemset is created from the already calculated frequent-2-itemsets. This module takes total number of items incurred in the dataset and combination size as input parameter to produce maximal frequent itemsets.

The algorithm for finding frequent itemsets is given below;

Algorithm for Module 2

Input

i: *Totalitem=4*

ii: *CSIZE=3*

Output

All MFI's of size having C size

```
1) #Combinations = NoofCombination(totalitem, CSIZE);
2) Combinations = New [#Combinations]
3) for(int k=0; k<#combinations; k++)
3.1) for(int i=0; i<CSIZE-1; i++)
3.2) Combination[k].run+=FindPositionNumber(i, j)
```

Figure 3.7: All MFI's of Combination Size

LS88-HL

First step of the algorithm takes both input parameters and returns the number of combinations. Second step creates combinations. Third step creates required maximal frequent itemsets by finding similarity of the combinations.

We will explain the working of the algorithm by considering dataset of example 3.1.

A = {1, 2, 3, 4}, B = {2, 4},

C = {1, 2, 4}, D = {1, 2, 3, 4}

Sim $_{AB}$ = 0.5, **Sim** $_{AC}$ = 0.75, **Sim** $_{AD}$ = 1,

Sim $_{BC}$ = 0.67, **Sim** $_{BD}$ = 0.5, **Sim** $_{CD}$ = 0.75

As total number of items is 4 and combination size is 3 therefore number of combinations will be;

n!/(r!(n-r)!) = 4!/(3!(4-3)!) = 4

The following combinations will be formed;

ABC, ABD, ACD, BCD

The similarity of each combination is equivalent to the sum of similarities of all subsets of the combination that is;

Sim $_{ABC}$ = **Sim** $_{AB}$ + **Sim** $_{AC}$ + **Sim** $_{BC}$ = 0.5 + 0.75 + 0.67 = 1.92

Sim $_{ABD}$ = **Sim** $_{AB}$ + **Sim** $_{AD}$ + **Sim** $_{BD}$ = 0.5 + 1 + 0.5 = 2

Sim $_{ACD}$ = **Sim** $_{AC}$ + **Sim** $_{AD}$ + **Sim** $_{CD}$ = 0.75 + 1+ 0.75 = 2.5

Sim $_{BCD}$ = **Sim** $_{BC}$ + **Sim** $_{BD}$ + **Sim** $_{CD}$ = 0.67 + 0.5 + 0.75= 1.92

The itemset having maximum similarity is the most maximal frequent itemset so on so far.

CHAPTER 4
IMPLEMENTATION

4 IMPLEMENTATION

This section of research thesis covers the implementation details of the algorithm. Here we will discuss the data structure used and also the tools and techniques used in the proposed technique.

4.1 DATA STRUCTURE USED IN THE ALOGORITHM

Our algorithm uses direct approach instead of iterative approach. It finds the similarity between every two items of the linked list. Then the similarity of items will be calculated equivalent to the number of combination size without generating the intermediate results. That is if the combination size is 5, the similarity of every 5 items will be calculated directly from similarity of every two items contained the items in 5 itemset without calculating similarity of 3 items, then 4 items.

4.2 ARTOOL

ARtool is a java based application tool. In our research work it is used to create synthetic database with .db extension. This database is then converted into ASCII format with .asc extension. Synthetic database can be created by opening the graphical user interface of ARtool, then clicking on the tools menu -> generate synthetic database, a new window will be opened. Relevant name will be given to the database with .db extension. The database will also have some characteristics i.e. number of items, number of transactions, average size of transactions, number of patterns and average size of patterns, correlation and corruption. Proper values will be assigned to each of these characteristics keeping in view importance and level of each one.

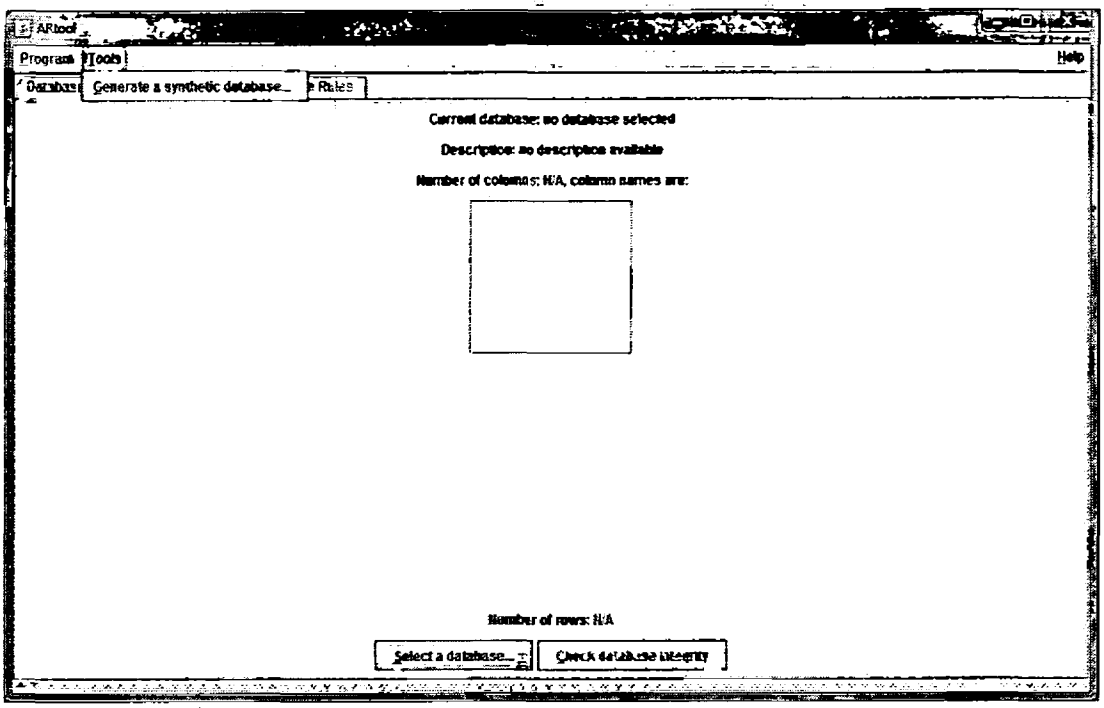

Figure 4.1: User Interface of ARtool

4.3 C++ LANGUAGE

C++ is middle level object oriented programming language. Our proposed idea is implemented in C++. The motivation of using C++ programming model by most of the programmers is that they know exactly what they want to do and how to use the language constructs to accomplish that goals [31]. As the data in the binary form can be easily understand and implemented in C++, therefore the database created in .db format by ARtool can first be converted into ASCII format then from ASCII to binary format.

4.4 DB TO ASCII CONVERSION

The dataset created by ARtool with .db extension can be converted into ASCII format by means of ARtool utility. For this conversion we will have to open command prompt. Then the following line of code is used for conversion.

C:\Program Files\Java\jdk1.6.0_07\bin>java db2asc db_filename asc_filename

If we do not specify ASCII filename then the default name for new file in ASCII format will be the name of file in db format with only change in extension of .asc instead of .db.

4.5 ASCII TO BINARY CONVERSION

The dataset in the ASCII format will be converted into binary format. In binary format the transaction is represented in the form of 0's and 1's. 1 represents presence of item in the transaction whereas 0 represent absence of item in the transaction. In binary format the dataset will be stored with .txt extension. We have converted the dataset from ASCII to binary format by means of C++ programming language.

4.6 MAXIMAL FREQUENT ITEMSET GENERATION

The proposed algorithm takes the dataset in binary format. The similarity of every two items is created and stored in two dimensional arrays. The combination size is given as input parameter on the basis of which combinations are created. Then the similarity of every combination is calculated from already being calculated 2 frequent itemsets. The above whole idea is implemented by means of C++ code. The pseudo code is given in two modules. Module 1 creates similarity of every two items of the data set where as the module 2 finds the maximal frequent itemsets of size "k" from already calculated two items similarity. The pseudo code for module 1 is available at page no 24 figure 3.1 and for module 2 available at page no 29 figure 3.7.

CHAPTER 5
EXPERIMENTAL RESULTS AND COMPARISON

5 EXPERIMENTAL RESULTS

For experimental results we will test proposed algorithm with different datasets. The same datasets will also be checked with some other algorithms in order to analyze the accuracy and correctness of our algorithm and to compare that how efficiently it generates results.

5.1 DATASETS GENERATED

For our experiments we will generate datasets by means of ARtool. We have generated three datasets given below in table 5.1;

Table 5.1 Synthetic Datasets for Experiments

DATASET	T	AT	I	P	AP
T1000_AT5_I7_P4_AP4.db	1000	5	7	4	4
T1500_AT7_I10_P40_AP5.db	1500	7	10	40	5
T2000_AT10_I13_P50_AP8.db	2000	10	13	50	8

First column of the table shows the name of the dataset used. The remaining columns represent properties of the concerned dataset i.e.

T -> Represents the number of transactions in the dataset.

AT -> Represents the average size of transactions.

I -> Represents the number of items incorporated in the dataset.

P -> Represents the number of patterns in the dataset.

AP -> Represents the average size of patterns in the dataset.

5.2 EXPERIMENTAL RESULTS OF PROPOSED ALGORITHM

The datasets of table 5.1 is executed by proposed algorithm in order to generate maximal frequent itemsets of "k" size given in appendix A.

5.3 EXPERIMENTAL RESEULTS OF MAFIA

The datasets of table 5.1 is executed with MAFIA [28] for maximal frequent itemsets generation and the results given in appendix "B" were recorded. MAFIA can run at any support threshold value, in our experiment we have taken support threshold of "0.02" for MAFIA

It is clear that MAFIA provide the experimental results of proposed algorithm which is the evidence of the correctness of proposed algorithm.

5.4 PERFORMANCE IN VARIOUS SCENARIOS

Performance can be evaluated by analyzing that how long the proposed system will take to produce the required results. For performance evaluation, the proposed algorithm is run with different datasets by changing one of the parameters that is either changing the number of items and keeping number of transactions fixed or changing the number of transactions and keeping the number of items fixed.

5.4.1 DATASETS WITH VARIABLE NUMBER OF ITEMS

For checking the performance of proposed algorithm, we will keep the number of transactions fixed and will change the number of items for each dataset given in table 5.2. The datasets will be run at both the proposed algorithm (SIMMFI) and MAFIA. We have executed the MAFIA algorithm with support threshold value of "0.02"

Table 5.2 Datasets with Constant Transactions and Variable number of Items

DATASET	T	AT	I	P	AP	TIME (MS)	
						SIMMFI	MAFIA
T5000_AT5_I6_P30_AP4.db	5000	5	6	30	4	16	110
T5000_AT6_I8_P30_AP4.db	5000	6	8	30	4	47	125
T5000_AT7_I10_P40_AP5.db	5000	7	10	40	5	110	328
T5000_AT10_I12_P40_AP6.db	5000	10	12	40	6	156	484
T5000_AT13_I15_P50_AP8.db	5000	13	15	50	8	297	703

Figure 5.1 shows the performance measure of the datasets of table 5.2. The graph shows that the execution time changes with increase in the number of items.

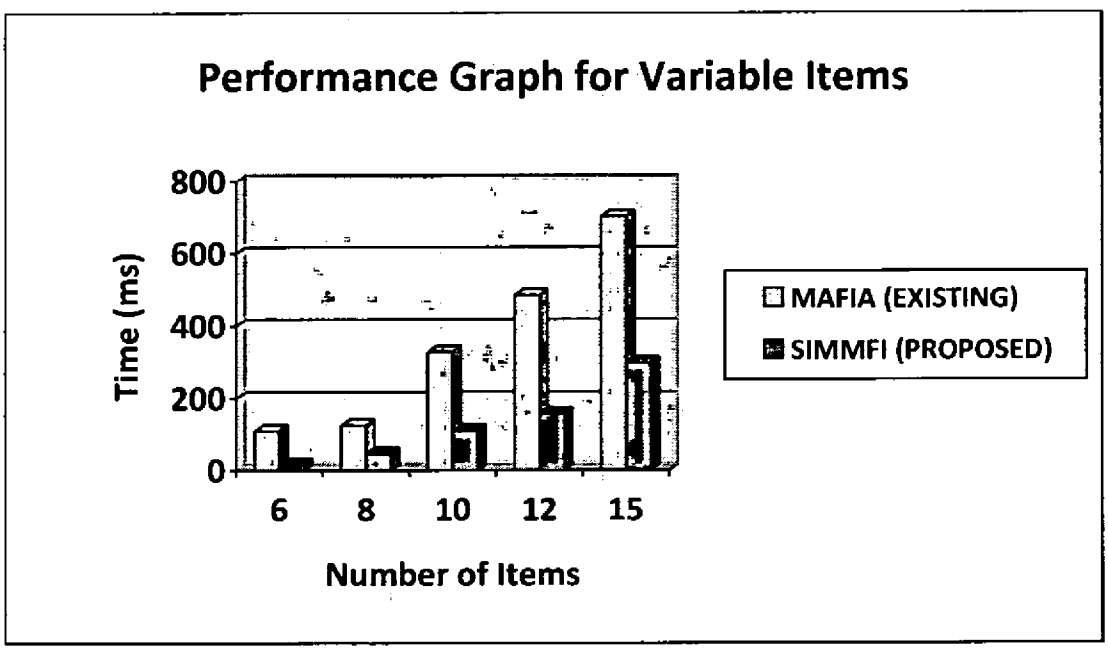

Figure 5.1: Performance of the Algorithm with Variable number of Items

5.4.2 DATASETS WITH VARIABLE TRANSACTIONS

Performance of the proposed algorithm can also be evaluated by changing the number of transactions in the datasets and keeping the number of items fixed given in table 5.3. The datasets have run with both the proposed algorithm (SIMMFI) and MAFIA. We have run MAFIA with support threshold value of "0.02".

Table 5.3 Datasets with Variable Transactions and Constant number of Items

DATASET	T	AT	I	P	AP	TIME (MS)	
						SIMMFI	MAFIA
T3000_AT9_I11_P40_AP5.db	3000	9	11	40	5	109	141
T4000_AT9_I11_P50_AP6.db	4000	9	11	50	6	109	141
T5000_AT9_I11_P50_AP6.db	5000	9	11	50	6	125	156
T6000_AT9_I11_P50_AP6.db	6000	9	11	50	6	125	141
T7000_AT9_I11_P50_AP6.db	7000	9	11	50	6	110	172

Figure 5.2 represents performance measure of the datasets of table 5.3. It may be noted that the execution time varies with increasing the number of transactions.

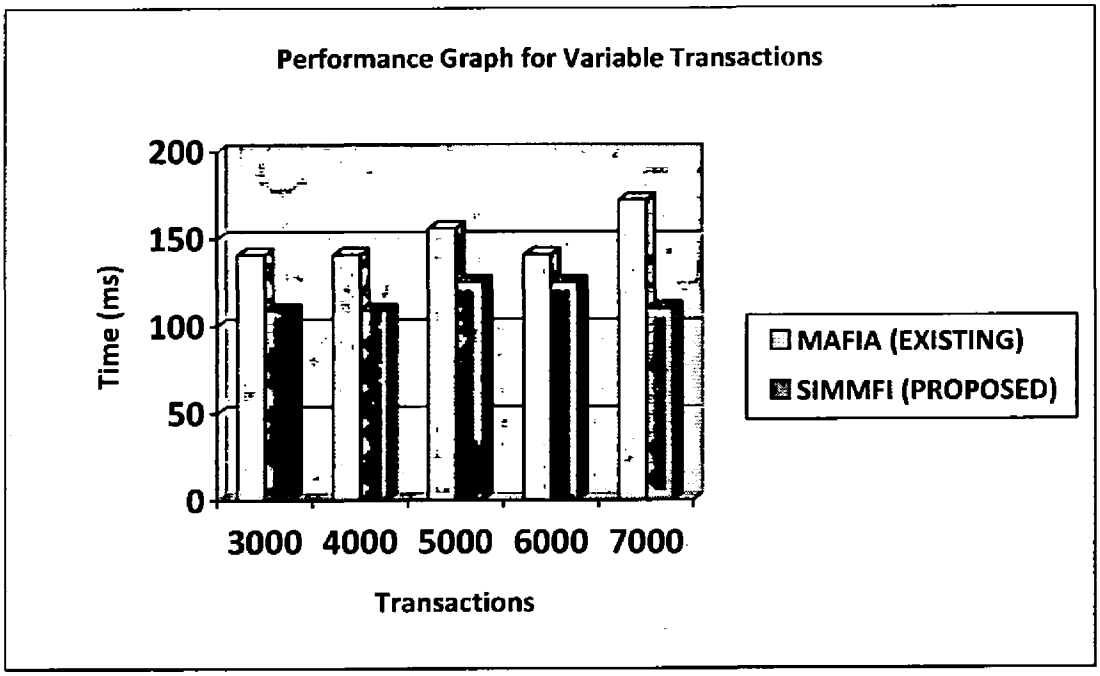

Figure 5.2: Performance of the Algorithm with Variable number of Transactions

CHAPTER 6
CONCLUSIONS

6 CONCLUSIONS

We have proposed a novel technique that discovers MFIS of specific size very easily without any support threshold and without intermediate results than the other establishing techniques available.

6.1 ACHIEVEMENT

The main objective of our research work is to find Maximal Frequent Itemsets in efficient way. For the said purpose we select similarity measure. Our technique finds MFIS by top down approach. Some of the previous algorithms found other itemsets on the way to find MFIS which results in performance degradation and also consume a lot of memory. Our techniques find limited number of itemsets of size K according to the requirements. The already existing algorithms were required user specified support threshold as a parameter with other parameters. If the support value is high a very limited number of itemsets are generated so there is chance that some of the important itemsets may be dropped. Our technique does not require any user provided support threshold so the MFIs are generated most frequently as compared to other techniques. In the previous chapter we have compared our algorithm (SIMMFI) with MAFIA which shows that our algorithm is efficient in comparison to MAFIA on different synthetic data sets.

6.2 FUTURE WORK

We applied similarity measure to compute MFI's of size to transactional database. The transactional databases were synthetic in nature. In future we intend to apply this technique to mine textual data from textual databases because of the in built capability of the algorithm to mine the frequent words of specific length with required changes.

REFERENCES

7 REFERENCES

[1] J. Han, and M. Kamber, "Data Mining Concepts and Techniques," 2nd ED, 2006.

[2] Prof Dr. Ahsan Abdullah, "A Brief Introduction to Data Mining," 2008.

[3] J. Wiley, "Data Mining: Concepts, Models, Methods and Algorithms," 2003.

[4] R. Agrawal, T. Imielinski, and A.N. Swami, "Mining association rules between sets of items in large databases," In Proc. of the 1993 ACM SIGMOD Int'l Conf. on Management of Data, ACM Press, 1993, pp. 207-216.

[5] R.J. Bayardo Jr. "Efficiently Mining Long Patterns from Databases," In Proc. of the ACM SIGMOD Int'l. Conf. on Management of Data, Seattle, Washington, USA, 1998, pp. 85-93.

[6] R. Rymon, "search through systematic set enumeration," In Proc. 3rd Int'l Conf. on principles of knowledge Representation and reasoning, 1992, pp. 539-550.

[7] S. Orlando, P. Palmerini, and R. Perego, *"The DCP algorithm for Frequent Set Counting,"* Technical Report CS-2001-7, Dip. di Informatica, Universita di Venezia, [Online],Available: http://www.dsi.unive.it/~orlando/TR01-7.pdf, 2001, pp. 1-16.

[8] K. Gouda, and J. Zaki, "GenMax: An Efficient Algorithm for Mining Maximal Frequent Itemsets," Data Mining and Knowledge Discovery, 2005, pp. 223-242.

[9] S.H. Khiyal, S. Rehman, and A. Salam, "Similarity BASED Mining for Finding Frequent Itemsets," Int'l Conf. ICCCS Daegu University South Korea, Nov 2007, 198-202.

[10] A.M.J Md. Zubair Rahman, and P. Balasubramanie, "An Efficient Algorithm for Mining Maximal Frequent Item Sets," Journal of Computer Science 4 (8): 2008, pp. 638-645.

[11] Y. L. Chequng, A.W.Fu, "An Fp-tree Approach for Mining N-most Interesting Itemsets," *In Proc. of the SPIE Conf. on Data Mining,* 2002, pp. 312-318.

[12] S.Bashir, Z.Jan, and A.R.Baig, "Fastv Algorithms for Mining Interesting Frequent Itemsets without Minimum Support," CoRR abs/0904.3319: 2009, pp. 1-25.

[13] E. Keogh, S. Lonardi, and C. Ratanamahatana, "Towards parameter-free data mining," In: Proc. ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining, Seattle (2004), pp. 206-215.

[14] E. Keogh, J. Lin, and W. Truppel, "Clustering of Time Series Subsequences is Meaningless: Implications for Past and Future Research," In proc. of the 3rd IEEE ICDM, 2003. Melbourne, FL.Nov 19-22, 2003, pp. 115-122.

[15] S. Bistarelli, F. Bonchi, "Interestingness is not a dichotomy: Introducing softness in constrained pattern mining," In: Jorge, A.M., Torgo, L., Brazdil, P.B., Camacho, R., Gama, J. (eds.) PKDD 2005. LNCS (LNAI), vol. 3721, Springer, Heidelberg (2005), pp. 22–33.

[16] J.H. Chang, "Mining Weighted Sequential Patterns in a Sequence Database with a Time-Interval Weight," ELSEVIER Journal of Knowledge Based Syestems, 2011, pp.1-9.

[17] S. Ju, C. Chen, "MMFI: An Effective Algorithm for mining Maximal Frequent Itemsets," Int'l Symposiums on Information Processing, IEEE, 2008, pp. 144-148.

[18] Tianming Hu, Sam Yuan Sung, Hui Xiong, Qian Fu, "Discovery of maximum length frequent itemsets," Information Sciences: an Int'l Journal, v.178 n.1, January, 2008, pp. 69-87.

[19] R. Ng, L. Lakshmanan, J. Han, and A. Pang, "Exploratory Mining and Pruning Optimizations of Constrained Association Rules," In Proc. of the ACM SIGMOD Conf. on Management of Data, 1998, pp. 13-24.

[20] J. Pei, and J. Han, "Can we push more constraints into frequent pattern mining?" In Proc of the 2000 ACM SIGKDD Int'l Conf. on Knowledge Discovery in Databases, KDD'00, Boston, Massachusetts, 2000, pp. 350-354.

[21] J. Pei , J. Han, Laks V.S. Lakshmanan, "Mining Frequent Item Sets with Convertible Constraints," Proc of the 17th Int'l Conf. on Data Engineering, April 02-06, 2001, pp. 433-442.

[22] R.J. Bayardo, "The hows, whys, and whens of constraints in itemset and rule discovery," In: Boulicaut, J.-F., De Raedt, L., Mannila, H. (eds.) Constraint-Based Mining and Inductive Databases. LNCS (LNAI), vol. 3848, Springer, Heidelberg 2006, pp. 1–13.

[23] S. Bistarelli, F. Bonchi," Interestingness is not a dichotomy: Introducing softness in constrained pattern mining," In: Jorge, A.M., Torgo, L., Brazdil, P.B., Camacho, R., Gama, J. (eds.) PKDD 2005. LNCS (LNAI), vol. 3721, Springer, Heidelberg, 2005, pp. 22–33.

[24] J.F. Boulicaut, and B. Jeudy, "Using constraints during set mining: Should we prune or not?" In Actes des Seizime Journes Bases de Donnes Avances BDA'00, Blois (F), 2000, pp. 221–237.

[25] R. Pears, and S. Kutty, "FGC: An efficient Constraint Based Frequent Set Miner," Proc. of the 2007 ACS/IEEE Int'l Conf. on Computer Systems and Applications, AICCSA, 2007, pp. 424-431.

[26] M.J. Zaki, S. Parthasarathy, M. Ogihara, and W. Li, "New algorithms for fast discovery of association rules," In proc. of the 3rd Int'l conf. on KDD and Data Mining (KDD '97), Newport Beach, California, August 1997, pp. 283-286.

[27] Bart Goethals SAC, "Memory issues in Frequent Itemset Mining," Nicosia, Cyprus, March 14-17, 2004, pp. 530-534.

[28] D. Burdick, and M. Calimlim, "MAFIA: A Maximal Frequent Itemset Algorithm," IEEE Transactions on Knowledge and Data Engineering, 2005, pp. 1490-1504.

[29] X. Zeng, "PADS: a simple yet effective pattern-aware dynamic search method for fast maximal frequent pattern mining," conf. on Knowledge Information System, Springer, 2008, pp. 375-391.

[30] A. Salam, and M.S.H. Khayal, "Mining top-k frequent patterns without minimum support threshold," Journal of Knowledge of Information System, Springer, Feb 2011, pp.57-86.

[31] http://www.hotscripts.com/listing/basic-features-of-c-language/.

APPENDIX A

Table A.1 Experimental Results of Proposed Algorithm

DATASET	T	AT	I	P	AP	MFI SIZE	FOUND MFI'S
T1000_AT5_I7_P4_AP4.db	1000	5	7	4	4	4	1 3 5 7 (6000)
							1 3 4 5 (5091)
							1 3 4 7 (5091)
							1 4 5 7 (5091)
							3 4 5 7 (5091)
							1 3 5 6 (4215)
							1 3 6 7 (4215)
							1 5 6 7 (4215)
							3 5 6 7 (4215)
							1 2 3 5 (3822)
							1 2 3 7 (3822)
							1 2 5 7 (3822)
							2 3 5 7 (3822)
							1 3 4 6 (3785)
							1 4 5 6 (3785)
							1 4 6 7 (3785)
							3 4 5 6 (3785)
							3 4 6 7 (3785)
							4 5 6 7 (3785)
							1 2 3 4 (3335)
							1 2 4 5 (3335)
							1 2 4 7 (3335)
							2 3 4 5 (3335)
							2 3 4 7 (3335)
							2 4 5 7 (3335)
							1 2 3 6 (3035)
							1 2 5 6 (3035)

							1 2 6 7 (3035) 2 3 5 6 (3035) 2 3 6 7 (3035) 2 5 6 7 (3035) 1 2 4 6 (3027) 2 3 4 6 (3027) 2 4 5 6 (3027) 2 4 6 7 (3027)
T1500_AT7_I10_P40_AP5.db	1500	7	10	40	5	8	2 3 4 5 6 7 8 9 (17685) 1 2 3 4 5 7 8 9 (17083) 1 2 3 4 5 6 7 9 (17057) 1 2 3 5 6 7 8 9 (16233) 1 2 3 4 6 7 8 9 (16047) 1 2 4 5 6 7 8 9 (15489) 1 2 3 4 5 6 7 8 (15438) 1 3 4 5 6 7 8 9 (15437) 1 2 3 4 5 6 8 9 (15306) 2 3 4 5 7 8 9 10 (14911) 2 3 4 5 6 7 9 10 (14826)

							1 2 3 4 5 7 9 10
							(14357)
							2 3 5 6 7 8 9 10
							(14069)
							2 3 4 6 7 8 9 10
							(13872)
							1 2 3 5 6 7 9 10
							(13470)
							1 2 3 5 7 8 9 10
							(13445)
							1 2 3 4 7 8 9 10
							(13401)
							2 4 5 6 7 8 9 10
							(13255)
							1 2 3 4 6 7 9 10
							(13252)
							2 3 4 5 6 7 8 10
							(13218)
							3 4 5 6 7 8 9 10
							(13199)
							2 3 4 5 6 8 9 10
							(13114)
							1 2 4 5 6 7 9 10
							(12814)
							1 3 4 5 6 7 9 10
							(12775)
							1 2 4 5 7 8 9 10
							(12771)
							1 2 3 4 5 7 8 10
							(12764)
							1 3 4 5 7 8 9 10

							(12718)
							1 2 3 4 5 6 7 10
							(12700)
							1 2 3 4 5 8 9 10
							(12632)
							1 2 3 6 7 8 9 10
							(12604)
							1 2 3 4 5 6 9 10
							(12593)
							1 2 5 6 7 8 9 10
							(12177)
							1 2 3 5 6 7 8 10
							(12134)
							1 3 5 6 7 8 9 10
							(12115)
							1 2 3 5 6 8 9 10
							(12032)
							1 2 3 4 6 7 8 10
							(12030)
							1 2 4 6 7 8 9 10
							(11991)
							1 3 4 6 7 8 9 10
							(11965)
							1 2 3 4 6 8 9 10
							(11892)
							1 4 5 6 7 8 9 10
							(11694)
							1 2 4 5 6 7 8 10
							(11581)
							1 3 4 5 6 7 8 10
							(11558)

							1 2 4 5 6 8 9 10 (11502) 1 2 3 4 5 6 8 10 (11466) 1 3 4 5 6 8 9 10 (11456)
T2000_AT10_I13_P50_AP8.db	2000	10	13	50	8	11	1 3 4 5 6 7 8 10 11 12 13 (35768) 1 2 3 4 5 6 7 8 10 12 13 (35520) 1 2 3 4 5 7 8 10 11 12 13 (34096) 1 2 4 5 6 7 8 10 11 12 13 (34089) 2 3 4 5 6 7 8 10 11 12 13 (33905) 1 2 3 4 5 6 7 8 10 11 13 (33897) 1 2 3 4 5 6 7 8 10 11 12 (33560) 1 3 4 5 6 7 8 9 10 12 13 (33514) 1 2 3 4 5 6 7 10 11 12 13 (32703) 1 2 3 4 6 7 8 10 11 12 13 (32628) 1 2 3 5 6 7 8 10 11 12 13 (32618) 1 2 3 4 5 6 7 8 11 12 13 (32484) 1 2 3 4 5 6 8 10

							11 12 13 (32442)
							1 4 5 6 7 8 9 10
							11 12 13 (32234)
							1 3 4 5 7 8 9 10
							11 12 13 (32200)
							1 3 4 5 6 7 8 9
							10 11 13 (32089)
							3 4 5 6 7 8 9 10
							11 12 13 (32039)
							1 2 3 4 5 7 8 9
							10 12 13 (31946)
							1 2 4 5 6 7 8 9
							10 12 13 (31902)
							2 3 4 5 6 7 8 9
							10 12 13 (31767)
							1 2 3 4 5 6 7 8 9
							10 13 (31737)
							1 3 4 5 6 7 8 9
							10 11 12 (31593)
							1 2 3 4 5 6 7 8 9
							10 12 (31448)
							1 3 4 5 6 7 9 10
							11 12 13 (30752)
							1 3 4 6 7 8 9 10
							11 12 13 (30739)
							1 3 5 6 7 8 9 10
							11 12 13 (30735)
							2 3 4 5 7 8 9 10
							11 12 13 (30688)
							1 2 3 4 5 6 7 9
							10 12 13 (30596)

							1 2 4 5 7 8 9 10
							11 12 13 (30585)
							1 3 4 5 6 7 8 9
							11 12 13 (30567)
							1 3 4 5 6 8 9 10
							11 12 13 (30560)
							1 2 3 4 6 7 8 9
							10 12 13 (30534)
							1 2 3 5 6 7 8 9
							10 12 13 (30521)
							1 2 3 4 5 7 8 9
							10 11 13 (30479)
							2 4 5 6 7 8 9 10
							11 12 13 (30444)
							2 3 4 5 6 7 8 9
							10 11 13 (30440)
							1 2 4 5 6 7 8 9
							10 11 13 (30412)
							1 2 3 4 5 6 7 8 9
							12 13 (30372)
							1 2 3 4 5 6 8 9
							10 12 13 (30343)
							1 2 3 4 5 7 8 9
							10 11 12 (30094)
							1 2 4 5 6 7 8 9
							10 11 12 (30037)
							1 2 3 4 5 6 7 8 9
							10 11 (29957)
							2 3 4 5 6 7 8 9
							10 11 12 (29917)
							1 2 3 4 5 7 9 10

							11 12 13 (29344)
							1 2 3 4 7 8 9 10
							11 12 13 (29236)
							1 2 4 6 7 8 9 10
							11 12 13 (29230)
							1 2 4 5 6 7 9 10
							11 12 13 (29228)
							1 2 5 6 7 8 9 10
							11 12 13 (29224)
							1 2 3 5 7 8 9 10
							11 12 13 (29220)
							2 3 4 5 6 7 9 10
							11 12 13 (29177)
							1 2 3 4 5 7 8 9
							11 12 13 (29142)
							1 2 3 4 5 8 9 10
							11 12 13 (29140)
							1 2 3 4 5 6 7 9
							10 11 13 (29094)
							2 3 4 6 7 8 9 10
							11 12 13 (29083)
							2 3 5 6 7 8 9 10
							11 12 13 (29083)
							1 2 4 5 6 7 8 9
							11 12 13 (29069)
							1 2 3 4 6 7 8 9
							10 11 13 (29014)
							1 2 4 5 6 8 9 10
							11 12 13 (29006)
							1 2 3 5 6 7 8 9
							10 11 13 (29005)

							2 3 4 5 6 7 8 9
							11 12 13 (29001)
							2 3 4 5 6 8 9 10
							11 12 13 (28986)
							1 2 3 4 5 6 7 8 9
							11 13 (28907)
							1 2 3 4 5 6 8 9
							10 11 13 (28889)
							1 2 3 4 5 6 7 9
							10 11 12 (28858)
							1 2 3 4 6 7 8 9
							10 11 12 (28678)
							1 2 3 5 6 7 8 9
							10 11 12 (28669)
							1 2 3 4 5 6 8 9
							10 11 12 (28602)
							1 2 3 4 5 6 7 8 9
							11 12 (28587)
							1 2 3 6 7 8 9 10
							11 12 13 (28072)
							1 2 3 4 6 7 9 10
							11 12 13 (28015)
							1 2 3 5 6 7 9 10
							11 12 13 (28010)
							1 2 3 4 5 6 7 9
							11 12 13 (27922)
							1 2 3 4 5 6 9 10
							11 12 13 (27921)
							1 2 3 5 6 7 8 9
							11 12 13 (27881)
							1 2 3 4 6 7 8 9

							11 12 13 (27875) 1 2 3 4 6 8 9 10 11 12 13 (27796) 1 2 3 5 6 8 9 10 11 12 13 (27792) 1 2 3 4 5 6 8 9 11 12 13 (27693)

APPENDIX B

Table B.1 Experimental Results of MAFIA with Support Threshold "0.02"

DATASET	T	AT	I	P	AP	MFI SIZE	FOUND MFI'S
T1000_AT5_I7_P4_AP4.db	1000	5	7	4	4	4	7 5 3 1 (1000)
							4 5 3 1 (698)
							4 7 3 1 (698)
							4 7 5 1 (698)
							4 7 5 3 (698)
							6 5 3 1 (406)
							6 7 3 1 (406)
							6 7 5 1 (406)
							6 7 5 3 (406)
							2 5 3 1 (275)
							2 7 3 1 (275)
							2 7 5 1 (275)
							2 7 5 3 (275)
							6 4 3 1 (406)
							6 4 5 1 (406)
							6 4 7 1 (406)
							6 4 5 3 (406)
							6 4 7 3 (406)
							6 4 7 5 (406)
							2 4 3 1 (275)
							2 4 5 1 (275)
							2 4 7 1 (275)
							2 4 5 3 (275)
							2 4 7 3 (275)
							2 4 7 5 (275)
							2 6 3 1 (275)
							2 6 5 1 (275)

							2 6 7 1 (275) 2 6 5 3 (275) 2 6 7 3 (275) 2 6 7 5 (275)
T1500_AT7_I10_P40_AP5.db	1500	7	10	40	5	8	6 8 4 5 9 7 3 2 (380) 1 8 4 5 9 7 3 2 (318) 1 6 4 5 9 7 3 2 (332) 1 6 8 5 9 7 3 2 (300) 1 6 8 4 9 7 3 2 (301) 1 6 8 4 5 9 7 2 (299) 1 6 8 4 5 7 3 2 (301) 1 6 8 4 5 9 7 3 (299) 1 6 8 4 5 9 3 2 (300) 10 8 4 5 9 7 3 2 (77) 10 6 4 5 9 7 3 2 (72) 10 1 4 5 9 7 3 2 (70) 10 6 8 5 9 7 3 2 (71) 10 6 8 4 9 7 3 2

							(71) 10 1 6 5 9 7 3 2 (69) 10 1 8 5 9 7 3 2 (69) 10 1 8 4 9 7 3 2 (69) 10 6 8 4 5 9 7 2 (71) 10 1 6 4 9 7 3 2 (69) 10 6 8 4 5 7 3 2 (71) 10 6 8 4 5 9 7 3 (71) 10 6 8 4 5 9 3 2 (72) 10 1 6 4 5 9 7 2 (69) 10 1 6 4 5 9 7 3 (69) 10 1 8 4 5 9 7 2 (69) 10 1 8 4 5 7 3 2 (69) 10 1 8 4 5 9 7 3 (69) 10 1 6 4 5 7 3 2 (69) 10 1 8 4 5 9 3 2 (70)

							10 1 6 8 9 7 3 2
							(68)
							10 1 6 4 5 9 3 2
							(70)
							10 1 6 8 5 9 7 2
							(68)
							10 1 6 8 5 7 3 2
							(68)
							10 1 6 8 5 9 7 3
							(68)
							10 1 6 8 5 9 3 2
							(69)
							10 1 6 8 4 7 3 2
							(68)
							10 1 6 8 4 9 7 2
							(68)
							10 1 6 8 4 9 7 3
							(68)
							10 1 6 8 4 9 3 2
							(69)
							10 1 6 8 4 5 9 7
							(70)
							10 1 6 8 4 5 7 2
							(68)
							10 1 6 8 4 5 7 3
							(68)
							10 1 6 8 4 5 9 2
							(69)
							10 1 6 8 4 5 3 2
							(69)
							10 1 6 8 4 5 9 3

							(69)
T2000_AT10_I13_P50_AP8.db	2000	10	13	50	8	11	11 3 6 12 1 13 8 7 5 4 10 (336) 2 3 6 12 1 13 8 7 5 4 10 (367) 2 11 3 12 1 13 8 7 5 4 10 (275) 2 11 6 12 1 13 8 7 5 4 10 (273) 2 11 3 6 12 13 8 7 5 4 10 (269) 2 11 3 6 1 13 8 7 5 4 10 (273) 2 11 3 6 12 1 8 7 5 4 10 (312) 9 3 6 12 1 13 8 7 5 4 10 (162) 2 11 3 6 12 1 13 7 5 4 10 (267) 2 11 3 6 12 1 13 8 7 4 10 (267) 2 11 3 6 12 1 13 8 7 5 10 (267) 2 11 3 6 12 1 13 8 7 5 4 (267) 2 11 3 6 12 1 13 8 5 4 10 (267) 9 11 6 12 1 13 8 7 5 4 10 (92) 9 11 3 12 1 13 8 7 5 4 10 (88)

							9 11 3 6 1 13 8 7 5 4 10 (92)
							9 11 3 6 12 13 8 7 5 4 10 (85)
							9 2 3 12 1 13 8 7 5 4 10 (126)
							9 2 6 12 1 13 8 7 5 4 10 (129)
							9 2 3 6 12 13 8 7 5 4 10 (141)
							9 2 3 6 1 13 8 7 5 4 10 (128)
							9 11 3 6 12 1 8 7 5 4 10 (91)
							9 2 3 6 12 1 8 7 5 4 10 (135)
							9 11 3 6 12 1 13 7 5 4 10 (85)
							9 11 3 6 12 1 13 8 7 4 10 (85)
							9 11 3 6 12 1 13 8 7 5 10 (85)
							9 2 11 3 12 13 8 7 5 4 10 (96)
							9 2 3 6 12 1 13 7 5 4 10 (125)
							9 2 11 12 1 13 8 7 5 4 10 (66)
							9 11 3 6 12 1 13 8 7 5 4 (85)
							9 11 3 6 12 1 13

							8 5 4 10 (85)
							9 2 3 6 12 1 13 8
							7 4 10 (125)
							9 2 3 6 12 1 13 8
							7 5 10 (125)
							9 2 11 3 1 13 8 7
							5 4 10 (66)
							9 2 11 6 12 13 8
							7 5 4 10 (63)
							9 2 11 3 6 13 8 7
							5 4 10 (66)
							9 2 11 6 1 13 8 7
							5 4 10 (65)
							9 2 3 6 12 1 13 8
							7 5 4 (125)
							9 2 3 6 12 1 13 8
							5 4 10 (125)
							9 2 11 3 12 1 8 7
							5 4 10 (69)
							9 2 11 6 12 1 8 7
							5 4 10 (69)
							9 2 11 3 6 1 8 7
							5 4 10 (70)
							9 2 11 3 6 12 8 7
							5 4 10 (68)
							9 2 11 3 12 1 13
							7 5 4 10 (63)
							9 2 11 3 12 1 13
							8 7 4 10 (63)
							9 2 11 6 12 1 13
							8 7 4 10 (63)

							9 2 11 6 12 1 13 7 5 4 10 (63)
							9 2 11 6 12 1 13 8 7 5 10 (63)
							9 2 11 3 12 1 13 8 7 5 10 (63)
							9 2 11 3 6 12 13 7 5 4 10 (62)
							9 2 11 3 12 1 13 8 7 5 4 (63)
							9 2 11 3 12 1 13 8 5 4 10 (63)
							9 2 11 3 6 1 13 7 5 4 10 (64)
							9 2 11 3 6 12 13 8 7 4 10 (62)
							9 2 11 3 6 12 13 8 7 5 10 (62)
							9 2 11 6 12 1 13 8 7 5 4 (63)
							9 2 11 3 6 1 13 8 7 4 10 (64)
							9 2 11 6 12 1 13 8 5 4 10 (63)
							9 2 11 3 6 1 13 8 7 5 10 (64)
							9 2 11 3 6 12 13 8 7 5 4 (62)
							9 2 11 3 6 12 13 8 5 4 10 (62)
							9 2 11 3 6 1 13 8

							7 5 4 (64)
							9 2 11 3 6 1 13 8
							5 4 10 (64)
							9 2 11 3 6 12 1 7
							5 4 10 (68)
							9 2 11 3 6 12 1 8
							7 4 10 (68)
							9 2 11 3 6 12 1 8
							7 5 10 (68)
							9 2 11 3 6 12 1 8
							5 4 10 (68)
							9 2 11 3 6 12 1 8
							7 5 4 (68)
							9 2 11 3 6 12 1
							13 8 7 10 (62)
							9 2 11 3 6 12 1
							13 7 4 10 (62)
							9 2 11 3 6 12 1
							13 7 5 10 (62)
							9 2 11 3 6 12 1
							13 7 5 4 (62)
							9 2 11 3 6 12 1
							13 5 4 10 (62)
							9 2 11 3 6 12 1
							13 8 7 5 (62)
							9 2 11 3 6 12 1
							13 8 7 4 (62)
							9 2 11 3 6 12 1
							13 8 4 10 (62)
							9 2 11 3 6 12 1
							13 8 5 10 (62)

							9 2 11 3 6 12 1 13 8 5 4 (62)