IMPACTS OF FREQUENT ITEMSET HIDING ALGORITHMS ON PRIVACY PRESERVING DATA MINING

A Thesis Submitted to
The Graduate School of Engineering and Sciences of
İzmir Institute of Technology
In Partial Fulfillment of the Requirements for the Degree of

MASTER OF SCIENCE

in Computer Engineering

by
Barış YILDIZ

July 2010
İZMİR

We approve the thesis of **Barış YILDIZ**

Assist. Prof. Dr. Belgin ERGENÇ
Supervisor

Assist. Prof. Dr. Murat Osman ÜNALIR
Committee Member

Assist. Prof. Dr. Tolga AYAV
Committee Member

5 July 2010

Prof. Dr. Sıtkı AYTAÇ
Head of the Department of
Computer Engineering

Assoc. Prof. Dr. Talat YALÇIN
Dean of the Graduate School of
Engineering and Sciences

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my advisor, Assist. Prof. Dr. Belgin Ergenç, for her guidance and encouragement. Her enthusiasm, inspiration, and great efforts during this research made this work valuable.

In addition, I thank TÜBİTAK-BİDEB for supporting my graduate study.

I am also grateful to my colleagues in İzmir Institute of Technology Department of Computer Engineering, especially former roommates Arda Müftüoğlu and Beray Atıcı. It was a pleasure to work in such an elite and friendly environment.

I would also like to state my special thanks to my girlfriend Hatice Şelale, for her encouragement, help and love.

Finally, I would like to express my greatest thanks to my parents especially my brother Özgür and my mother Fatma for motivating and supporting me throughout my whole life as well as in my graduate study. I must express my thanks again to my mother for her patience during preparation and presentation of the thesis.

ABSTRACT

IMPACTS OF FREQUENT ITEMSET HIDING ALGORITHMS ON PRIVACY PRESERVING DATA MINING

The invincible growing of computer capabilities and collection of large amounts of data in recent years, make data mining a popular analysis tool. Association rules (frequent itemsets), classification and clustering are main methods used in data mining research. The first part of this thesis is implementation and comparison of two frequent itemset mining algorithms that work without candidate itemset generation: Matrix Apriori and FP-Growth. Comparison of these algorithms revealed that Matrix Apriori has higher performance with its faster data structure

One of the great challenges of data mining is finding hidden patterns without violating data owners' privacy. Privacy preserving data mining came into prominence as a solution. In the second study of the thesis, Matrix Apriori algorithm is modified and a frequent itemset hiding framework is developed. Four frequent itemset hiding algorithms are proposed such that: i) all versions work without pre-mining so privacy breech caused by the knowledge obtained by finding frequent itemsets is prevented in advance, ii) efficiency is increased since no pre-mining is required, iii) supports are found during hiding process and at the end sanitized dataset and frequent itemsets of this dataset are given as outputs so no post-mining is required, iv) the heuristics use pattern lengths rather than transaction lengths eliminating the possibility of distorting more valuable data.

ÖZET

SIK KÜMELERİ GİZLEME ALGORİTMALARININ GİZLİLİĞİ KORUYAN VERİ MADENCİLİĞİ ÜZERİNE ETKİLERİ

Son yıllarda bilgisayar yeteneklerinin önlenemez büyümesi ve büyük miktarda verinin toplanması, veri madenciliğini gözde bir analiz aracı yapmıştır. Birliktelik kuralları (sık kümeler), sınıflandırma ve kümeleme veri madenciliğinin temel yöntemleridir. Bu tezin ilk çalışması aday küme üretmeyen iki algoritma Matrix Apriori ve FP-Growth sık küme bulma algoritmalarının uygulanması ve değerlendirilmesidir. Bu iki algoritmanın karşılaştırılması hızlı matris veri yapısıyla Matrix Apriori'nin daha yüksek başarıma sahip olduğunu açığa çıkarmıştır.

Veri madenciliğinin artan gücünün ortaya çıkardığı sorunlardan bir tanesi kişilerin ve şirketlerin gizliliğini ihlal etmeden saklı örüntülerin bulunmasıdır. Bu tezin ikinci bölümünde gözde veri madenciliği tekniklerinden biri olan sık kümelerin bulunması için gizliği koruyan bir yaklaşım önerilmiştir. İkinci olarak, Matrix Apriori algoritması üzerinde değişlik yapılmış ve sık küme gizleme çerçevesi geliştirilmiştir. Dört sık küme gizleme algoritması önerilmiştir, öyle ki: i) bütün sürümler ön madencilik olmadan çalışmakta ve sık kümelerin önceden bulunmasının neden olduğu gizlilik açığı önlenmektedir, ii) ön madencilik gerekmediğinden verimlilik artmıştır, iii) destek değerleri gizleme sürecinde bulunmaktadır ve sonunda temizlenmiş veri kümesi ve bu veri kümesinin sık kümeleri çıktı olarak verilmektedir yani sonradan madenciliğe gerek yoktur, iv) sezgiseller işlem uzunluğundansa örüntü uzunluğunu kullanarak daha değerli veri üzerinde bozma yapma olasılığını elemektedir.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

Data mining, defined as the process of discovering knowledge or patterns from massive amounts of data (Liu 2009), has become a popular way to discover strategic knowledge. Direct mail marketing, web site personalization, bioinformatics, credit card fraud detection, text analysis and market basket analysis are some examples where data mining techniques are commonly used.

Data mining models are divided into two as predictive and descriptive. Predictive models include tasks regression, classification, time series analysis and prediction. Descriptive models include tasks clustering, summarization, association rules and sequence discovery (Dunham 2002).

Association rule mining reveals relationships among set of items in a database in two steps frequent itemset mining and producing association rules from these itemsets. It was firstly introduced by (Agrawal 1993), followed by popular Apriori algorithm (Agrawal 1994) which listed in top ten data mining algorithms (Wu 2008). Although it boosted data mining research, Apriori algorithm has a bottleneck of multiple database scan for candidate itemset generation. In (Han 2000) FP-Growth algorithm proposed for frequent itemset mining without candidate generation. It stores information of database in tree structure called FP-tree and scans database only twice. Later in (Pavon 2006), Matrix Apriori algorithm is proposed. It is similar to FP-Growth in the way of database scanning and storing information of database in a compact data structure but matrix data structure is used instead of tree.

Data mining is efficiently applied to many fields like clustering in bioinformatics, association rules in market basket analysis, classification in credit scoring, time series analysis in financial decision supporting. However, the increasing power of computers handling huge amount data and malicious usage made data mining a risk to privacy of individuals and companies. In Figure 1.1 a simple example of privacy problem caused by combining information from different sites is given. Zip codes of medical records are anonymized to protect disclosure of patient and information in personal website and address in yellow pages do not cause a privacy problem solely. However, a macilious internal human and hacker may combine the information in different sites and label medical record of patient.

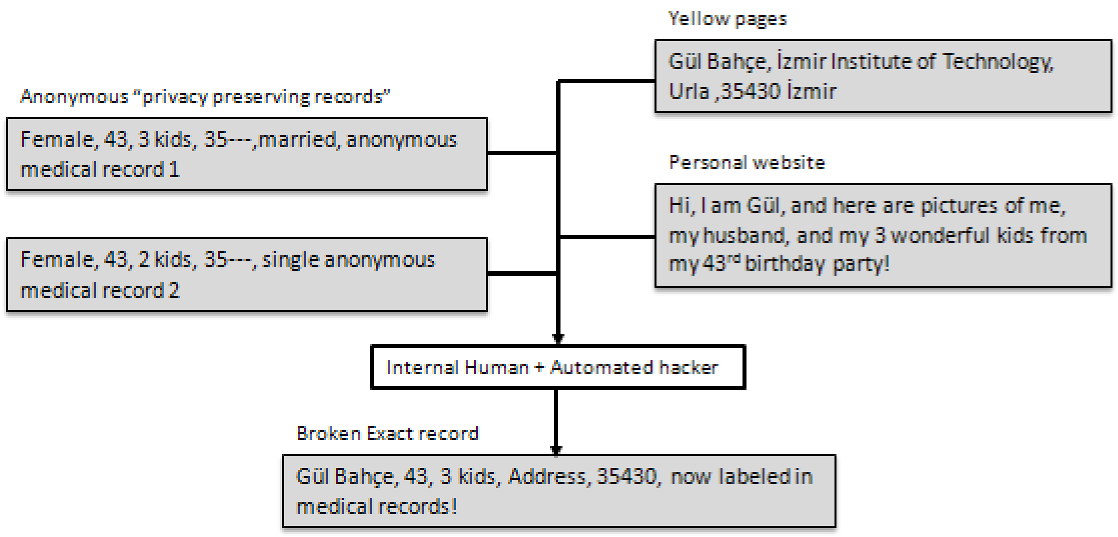

Yellow pages
Gül Bahçe, İzmir Institute of Technology, Urla ,35430 İzmir
Anonymous "privacy preserving records"
Female, 43, 3 kids, 35---,married, anonymous medical record 1
Personal website
Hi, I am Gül, and here are pictures of me, my husband, and my 3 wonderful kids from my 43rd birthday party!
Female, 43, 2 kids, 35---, single anonymous medical record 2
Internal Human + Automated hacker
Broken Exact record
Gül Bahçe, 43, 3 kids, Address, 35430, now labeled in medical records!

Figure 1.1. Privacy problem example

Public sensitivity against data mining increased because it is seen a threat to individuals private information as shown in the example above. On the other hand, data mining is important for efficiently discovering knowledge. Privacy preserving data mining arise from the need for continue performing data mining efficiently but preserving private data or knowledge of individuals and companies. It is defined as data mining techniques that use specialized approaches to protect against the disclosure of private information may involve anonymizing private data, distorting sensitive values, encrypting data, or other means to ensure that sensitive data is protected (Liu 2009).

Privacy preserving data mining is divided into two major categories: data hiding and rule hiding. Data hiding aims to design new protocols to perturb, anonymize or encrypt raw data while sensitive private data is protected and underlying patterns can still be discovered (Subramanian 2008). Rule hiding refers to design algorithms is such a way that sensitive rules or patterns stay unrevealed while remaining rules or patterns can still be mined. The original data is distorted or blocked by rule hiding algorithms.

Privacy, the new direction of data mining research is the main motivation for start point of this thesis study. It is decided to apply privacy preserving data mining techniques for frequent itemset mining. Surveying literature, it has been seen that many algorithms for association rule or frequent itemset hiding are Apriori based and as it is mentioned above it has a disadvantage of multiple database scanning. Therefore, algorithms without candidate generation are studied firstly. Matrix Apriori and FP-Growth algorithms are compared and a paper prepared (Yıldız 2010) from this first phase of thesis. Since results showed that Matrix Apriori performed better and its matrix data structure is easy to handle, thesis study is directed to proposing a frequent itemset hiding algorithm based on Matrix Apriori. As its matrix data structure gives pattern information, the algorithm is modified to have itemset hiding capabilities. In addition, innovative heuristics for selection of item distortion are proposed which use pattern length rather transaction length proposed by past studies on frequent itemset hiding. These algorithms are compared for different cases and results discussed. A new paper for the second phase of the thesis has been prepared and submitted. All the progress is depicted in this thesis. The goal and the structure of this thesis study are given in next subsections.

1.1. Thesis Aim and Objectives

Data mining is a growing area of study in computer science and it is applied to many fields. However, malicious usage may cause privacy problems. It is a challenge to perform data mining without violating privacy of data or knowledge. This necessity emerged privacy preserving data mining. It is a recently grown aspect of data mining and there is much work to do. Attracted by these and popularity of frequent itemset mining in data mining, frequent itemset hiding of privacy preserving data mining is studied in this thesis.

The objectives of this thesis are:

- To understand frequent itemset mining and compare two of algorithms Matrix Apriori and FP-Growth working without candidate generation.
- To understand privacy preserving data mining and frequent itemset hiding and propose frequent itemset hiding algorithm.
- To observe impacts of proposed frequent itemset hiding algorithms as side effects, runtimes and distortion for different cases and databases.

1.2. Organization of Thesis

The organization of this thesis is as follows:

- Chapter 2 presents related work giving general information about data mining and more detailed of frequent itemset mining. Following, introduction to privacy preserving data mining and information about techniques are given. Afterwards, more detailed past work of frequent itemset hiding.
- Chapter 3 presents frequent itemset mining and detailed explanation of two frequent itemset mining algorithms working without candidate generation. FP-Growth and Matrix Apriori algorithms are introduced and discussed with examples. Next, general information on frequent itemset hiding is given. Lastly, Matrix Apriori based frequent itemset hiding approach is explained in detail and four versions of proposed algorithms are discussed.
- Chapter 4 present performance evaluations of Matrix Apriori and FP-Growth algorithms followed by performance evaluation of Matrix Apriori based frequent itemset hiding algorithms. Comparison of Matrix Apriori and FP-Growth is done for total and for two phases as building data structure and finding frequent itemset. Two databases of different characteristics are used for evaluations. Afterwards, comparison of proposed Matrix Apriori based frequent itemset hiding algorithms given for increasing number of sensitive itemsets and increasing support of sensitive itemsets. Side effects as lost itemsets and runtimes are shown. To understand the effect of database size, two databases of different number of transactions are used for evaluations.
- Chapter 5 presents conclusion. A summary and the contribution of thesis is given and following future work is stated.

CHAPTER 2

RELATED WORK

2.1. Introduction

Data mining has attracted a great deal of attention in the information society in recent years, due to the wide availability of huge amounts of data and the need for turning such data into useful information and knowledge (Han 2005). It is applied to many fields ranging from bioinformatics, market analysis and fraud detection to earth sciences. However, there is a problem of keeping sensitive data private while continue data mining. As we cannot set aside the benefits of data mining, privacy preserving data mining has been introduced to take privacy into consideration of data mining research. In next sections of this chapter, related work about firstly data mining in general, followed by frequent itemet mining in detail, then privacy preserving data mining in general and lastly frequent itemset hiding in detail are given.

2.2. Overview of Data Mining

Data mining is a recently emerging field, connecting the three worlds of databases, artificial intelligence and statistics (Lindell 2002). It involves the use of data analysis tools to discover previously unknown, valid patterns and relationships in large datasets (Seifert 2004). Model created for data mining can be predictive or descriptive. Predictive models make a prediction about values of data using known results found from different data. Descriptive models identify patterns of relationships in data. Common tasks of predictive models are classification, regression, time series analysis and prediction. Clustering, summarization, association rules and sequence discovery are common tasks of predictive data mining models (Dunham 2002). These are depicted in Figure 2.1.

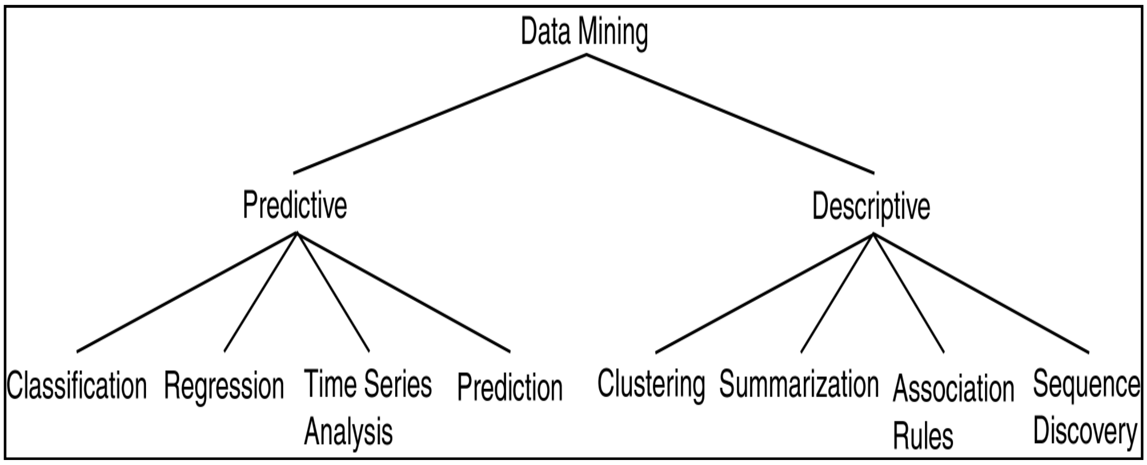


Figure 2.1. Data mining models and tasks
(Source: Dunham 2002)

There are mainly three data mining techniques: classification, clustering and association rule mining. Classification uses a training set and builds a classifier to predict the classes of new instances. Clustering divides dataset into clusters of which members are similar to each other and different from members of other clusters. Association rule mining finds patterns and relationships among dataset. These techniques are briefly introduced in following subsections.

2.2.1. Classification

Classification maps data into predefined groups or classes. Simply classifies data based on training set and uses it classifying new data (Han 2005). Classification algorithms can be divided into five as statistical-based, distance-based, decision tree-based, neural network-based and rule based algorithms (Dunham 2002). It is formally defined as

Given a database $D = \{t_1, \ldots, t_n\}$ of tuples (items, records) and a set of classes $C=\{C_1,\ldots C_m\}$, the classification problem is to define a mapping $f: D \rightarrow C$ where each t_i is assigned to one class. A class, Cj, contains precisely those tuples mapped to it; that is, $C_j=\{t_i|f(t_i)=C_j, 1 \leq i \leq n, \text{ and } t_i \in D\}$

A simple example for classification is teachers’ grading students as A, B, C, D, or F. Using boundaries we can classify grades as A if grade ≥ 90, B if 90 > grade ≥ 80, C if 80 > grade ≥ 70, D if 70 > grade ≥ 60, F if 60 > grade. Some popular classification algorithms are C4.5 (Quinlan 1993), CART (Breiman 1984), Naïve Bayes (Domingos 1997).

2.2.2. Clustering

The process of grouping a set of physical or abstract objects into classes of similar objects is called clustering. A cluster is a collection of data objects that are similar to one another within the same cluster and are dissimilar to the objects in other clusters (Han 2005). Unlike classification the groups are not pre defined. Clustering algorithms can be divided into three as hierarchical, partitional, categorical algorithms (Dunham 2002). Formal definition of clustering is

> *Given a database $D=\{t_1, t_2,\ldots,t_n\}$ of tuples and an integer value k, the clustering problem is to define a mapping $f: D\rightarrow \{1,\ldots,k\}$ where each ti is assigned to one cluster K_j, $1 \leq j\leq k$. A cluster, K_j, contains precisely those tuples mapped to it; that is, $K_j=\{t_i|f(t_i)=K_j, 1 \leq I \leq n$, and $t_i \in D\}$.*

A simple example for clustering is catalog design for targeted demographic groups based on attributes such as income, location, physical characteristics of potential customers. New specific catalogs design using results of clustering may be distributed to targeted population to attract customers. Some popular clustering algorithms are DBSCAN (Ester 1996) and k-means (Lylod 1982).

2.2.3. Association Rule Mining

Association rule miming finds relationships and patterns between items in a database. It is a two step process. Firstly, frequent itemsets are found and secondly from these itemsets, rules are produced. Formal definition of association rule mining is

> *Given a set of items $I=\{I_1, I_2, \ldots, I_m\}$ and a database of transactions $D=\{t_1,t_2,\ldots,t_n\}$ where $t_i=\{I_{i1}, I_{i2}, \ldots,I_{ik}\}$ and $I_{ij} \in I$ and X,Y are set of items, the association rule problem is to identify all association rules $X \rightarrow Y$ with a*

minimum support and confidence where support of association rule $X \rightarrow Y$ is the percentage of transactions in the database that contain $X \cup Y$ and confidence is the ratio of support of $X \cup Y$ to support of X.

Simply the purchasing of one product when another product is purchased in the market basket data represents an association rule. A well known illustrative example of association rules is ‘‘Diaper! Beer’’ which can be explained by the fact that, when dads buy diapers for their babies, they also buy beer at the same time for their weekend’s game watching (Liu 2008). Some popular association rule mining algorithms Apriori (Agrawal 1994), ECLAT (Zaki 1997) and FP-Growth (Han 2000).

2.3. Frequent Itemset Mining

The progress in bar-code and computer technology has made it possible to collect data about sales and store as transactions which is called basket data. This stored data attracted researches to apply data mining to basket data. As a result association rules mining came into prominence which is mentioned as synonymous to market basket analysis. As stated before association rule mining is a two step process. Firstly, frequent itemsets are found using minimum support value, and this step is the main concentration of association rule mining algorithms. Later from these itemsets using minimum confidence value rules are produced. As the differing part of the algorithms are frequent itemset finding part, association rule mining, frequent itemset mining or frequent pattern mining terms are used interchangeably. Association rule mining which was first mentioned in (Agrawal 1993) is one of the most popular data mining approaches. Not only in market business but also in variety of areas association rule mining is used efficiently. In (Duru 2005), Apriori algorithm is used on a diabetic database and developed application is used to discover social status of diabetics and (Alves 2009) represents a survey of frequent pattern mining from gene expression data. In a report (Grossman 1998), association rules are listed in the success stories part and in a survey (Wu 2008) the Apriori algorithm is listed in top 10 data mining algorithms.

The proposed algorithm in (Agrawal 1993) makes multiple passes over database. In each pass, beginning from one element itemsets, the support values of itemsets are counted. These itemsets are called candidate itemsets which are extended from the frontier sets delivered from previous pass. If a candidate itemset is measured as frequent then it is added to frontier sets for the next pass.

The Apriori algorithm proposed in (Agrawal 1994) boosted data mining research with its simple way of implementation. The algorithm generates candidate itemsets to be counted in a pass by using only the itemsets found large in previous pass – without considering all of the transactions in the database. So too many unnecessary candidate generation and support counting is avoided. Apriori is characterized as a level-wise complete search algorithm using anti-monotonicity of itemsets, “if an itemset is not frequent, any of its superset is never frequent” (Han 2005 and Wu 2008).

There have been many improvements for Apriori algorithm. Partitioning approach proposed in (Savasere 1995). Sampling approach is proposed in (Toivonen 1996). (Zaki 1997) proposed vertical data format for clustering transactions and producing frequent itemsets from these clusters. Although these algorithms are showed to perform better than Apriori, most significant improvement is lately proposed FP-Growth algorithm in (Han 2000). The main objective is to skip candidate generation and test step which is the bottleneck of the Apriori like methods. The algorithm uses a compact data structure called FP-tree and pattern fragment growth mining method is developed based on this tree. FP-growth algorithm scans database only twice. It uses a divide and conquer strategy. Algorithm relies on Depth First Search scans while in Apriori Breath First Search scan is used (Hipp 2000). It is stated in (Han 2000) that FP-growth is at least an order of magnitude faster than Apriori.

In several extensions for both Apriori and FP-growth accuracy of results is sacrificed for better speed. Matrix Apriori proposed in (Pavon 2006), combines positive properties of these two algorithms. Algorithm employs two simple structures: A matrix of frequent items called MFI and a vector storing the support of candidates called STE. Matrix Apriori consists of three procedures. First builds matrix MFI and populates vector STE. Second modifies matrix MFI to speed up frequent pattern search. Third identifies frequent patterns using matrix MFI and vector STE.

Detailed studies for comparing performances of Apriori and FP-Growth algorithms can be found in (Han 2000, Hipp 2000 and Zheng 2001). These studies reveal out that FP-Growth perform better than Apriori when minimum support value is decreased. Matrix Apriori algorithm combining the advantages of Apriori and FP-Growth was proposed as a faster and simpler alternative to these algorithms but there is no work showing its performance and this motivated first step of this thesis.

2.4. Overview of Privacy Preserving Data Mining

In today’s information age, data collection is ubiquitous, and every transaction is recorded somewhere. This increase in data collection with the tools capable of analyzing this huge volume of information, has led to privacy concerns. For the society protecting private data is important. For example, several laws now require explicit consent prior to analysis of an individuals’ data (Vaidya 2004). To add this, the privacy concern is not limited to the individuals. Companies may be willing to share their information for their common benefits; however, they may also be aware of sharing private information.

The trade-off between using private information for data mining and keeping it secret is a growing challenge. On the other hand privacy preservation in data mining is a rising field of research. Many researchers are studying on this topic and many techniques have been proposed. Privacy preserving data mining (PPDM) is divided into two as data hiding and rule hiding. Data hiding techniques aim to preserve individual’s sensitive data private and modify data mining algorithms in such a way that sensitive data cannot be inferred from results of data minig algorithm. In other words input privacy is preserved. Rule hiding techniques aim to preserve the sensitive rules or private patterns and modify original data in such a way that all sensitive patterns or rules stay unrevealed while remaining ones can still be discovered. Rule hiding is known as to preserve output privacy.

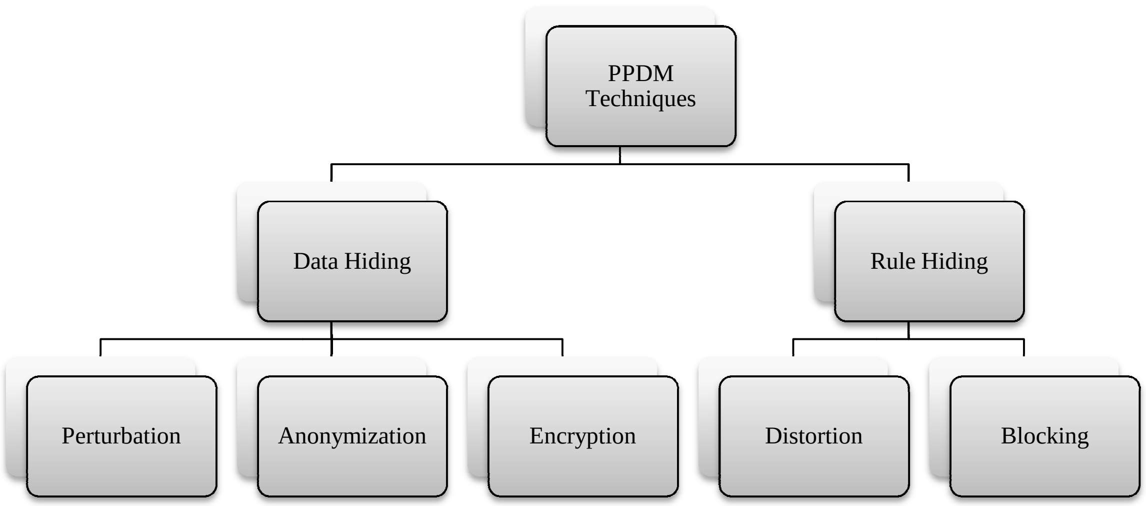


Figure 2.2. PPDM Techniques

Taxonomy of PPDM techniques is given in Figure 2.2. These techniques are briefly introduced in following subsections.

2.4.1. Data Hiding

The main objective of data hiding is to design new protocols to perturb, anonymize or encrypt raw data so that sensitive data remains sensitive during and after the mining operation while underlying data patterns can still be discovered (Subramanian 2008). In the following subsections, techniques used in data hiding are introduced.

2.4.1.1. Perturbation

One approach to privacy-preserving data mining is based on perturbing the original data, then providing the perturbed dataset as input to the data mining algorithm. The privacy-preserving properties are a result of the perturbation. Data values for individual entities are distorted, and thus individually identifiable (private) values are not revealed (Vaidya 2006).

The randomization technique uses data distortion methods in order to create private representations of the records. In most cases, the individual records cannot be recovered, but only aggregate distributions can be recovered. These aggregate distributions can be used for data mining purposes. Two kinds of perturbation are possible with the randomization method (Aggarwal 2008):

Additive Perturbation: In this case, randomized noise is added to the data records. The overall data distributions can be recovered from the randomized records. Data mining and management algorithms redesigned to work with these data distributions.

Multiplicative Perturbation: In this case, the random projection or random rotation techniques are used in order to perturb the records.

In (Agrawal 2000), randomization technique is applied. Noise is added to the original data and the attribute values of records are masked. Decision tree classification is evaluated over perturbed data.

2.4.1.2. Anonymization

Data anonymization aims at preventing an adversary from mapping sensitive information to an individual with the help of information provided in attributes known as quasi-identifiers. Information is routinely made public by removing primary identifiers such as names and SSNs. However, by combining records attributes individual records can be exactly identified (Aggarwal 2008 and Subramanian 2008). Anonymization, simply reduces granularity of data representation and k-anonymity and l-diversity are approaches for anonymization.

In k-anonymity quasi-identifers are generalized or suppressed in such a way that they become identical for k records, where $k > 1$.

L-diversity in addition to k-anonymity ensures that all tuples with similar values of quasi-identifiers have diverse values for their sensitive attributes.

In (Sweeney 1998), two algorithms Datafly and µ-Argus are proposed for k-anonymity. Later in (Machanavajjhala 2007), l-diversity approach proposed which overcomes weaknesses of k-anonymity.

2.4.1.3. Encryption

In many cases, multiple parties may wish to share aggregate private data, without leaking any sensitive information at their end. For example, different superstores with sensitive sales data may wish to coordinate among themselves in knowing aggregate trends without leaking the trends of their individual stores. This requires secure and cryptographic protocols for sharing the information across the different parties. The data may be distributed in two ways across different sites (Aggarwal 2008):

Horizontal Partitioning: In this case, the different sites may have different sets of records containing the same attributes.

Vertical Partitioning: In this case, the different sites may have different attributes of the same sets of records.

In (Kantarcioglu 2004), secure mining of association rules over horizontally partitioned data and in (Vaidya 2002) secure mining of association rules over vertically portioned data approaches are proposed. The methods incorporate cryptographic techniques to continue data mining without revealing individual transactions at each distributed site. (Lindell 2002 and Pinkas 2002) proposed protocols for privacy preserving distributed classification by decision tree learning.

2.4.2. Rule Hiding

The main focus of rule hiding is association rules and frequent patterns. Association rule hiding refers to the process of modifying the original database in such a way that certain sensitive association rules disappear without seriously affecting the data and the non-sensitive rules (Aggarwal 2008). The main goal here is to hide as many sensitive rules as possible, while keeping preserved as many non-sensitive rules as possible.

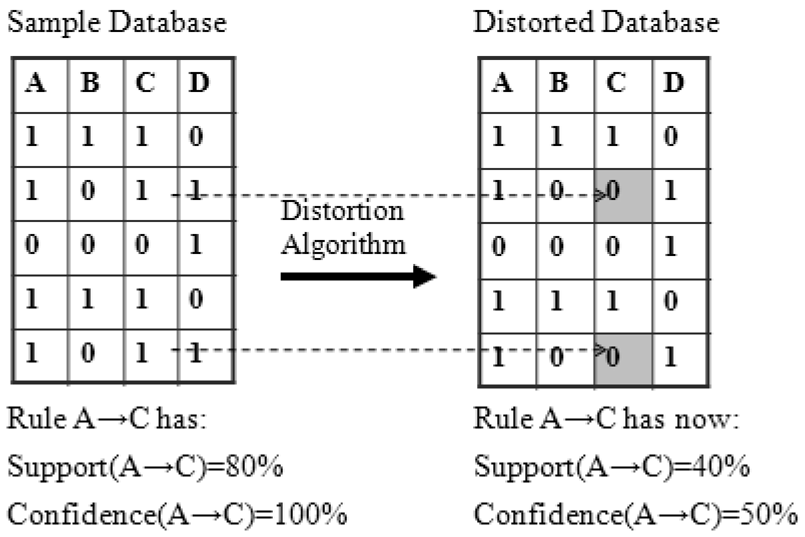
To make the necessity of hiding association rules clear here is a scenario. Let us suppose that we are negotiating with Dedtrees Paper Company, as purchasing directors of BigMart, a large supermarket chain. They offer their products in reduced prices, provided that we agree to give them access to our database of customer purchases. We accept the deal and Dedtrees starts mining our data. By using an association rule mining tool, it can be found that people who purchase skim milk also purchase Green Paper. Dedtrees now runs a coupon marketing campaign offering a 50 cents discount on skim milk with every purchase of a Dedtrees product. The campaign cuts heavily into the sales of Green Paper, which increases the prices to us, based on the lower sales. During our next negotiation with Dedtrees, we found out that with reduced competition they are unwilling to offer to us a low price. Finally, we start losing business to our competitors, who were able to negotiate a better deal with Green Paper. In other words, the aforementioned scenario indicates that BigMart should sanitize competitive information (and other important corporate secrets of course) before delivering their database to Dedtrees, so that Dedtrees does not monopolize the paper market (Verykios 2004b).

As seen in the example above hiding association rules which are sensitive is a should be considered subject in information sharing for data mining. Distortion and blocking are techniques used in rule hiding and introduced in following subsections.

2.4.2.1. Distortion

Distortion based association rule hiding algorithms run on the strategy that is based on reducing the support and confidence of rules. Remember that these two specify how significant the rules are. The transactions are modified by removing some items or inserting new items. On the other hand we should ensure that the information loss incurred by the process is minimal.

We will use the bitmap notation to represent transactions in the database. If an item exists in a transaction then it is represented with “1” and “0” if it does not exist. For instance, consider a database with items A, B, C and D. A transaction T with items A and C is represented as T (1010) using the bitmap notation. The distortion approach simply changes these bit values of items in transactions (Verykios 2004b). A simple example is shown in Figure 2.3. Each row represents a transaction. A, B, C, and D are the items in the database. We simply change the bit values representing item C of second and fifth transactions. It is clear that support and confidence for the rule A→C is decreased. If the new support and confidence values are lower than our thresholds defined then the rule is hidden.



A	B	C	D
1	1	1	0
1	0	1	1
0	0	0	1
1	1	1	0
1	0	1	1

A	B	C	D
1	1	1	0
1	0	0	1
0	0	0	1
1	1	1	0
1	0	0	1

Figure 2.3. Distortion based rule hiding

2.4.2.2. Blocking

Blocking based algorithms provide safer alternative especially in critical real life applications where the distinction between false and unknown is vital (Aggarwal 2008). Consider a medical institution that will make some of its data public, and the data is sanitized by replacing actual attribute values by false values. Researchers may use this data, but obtain misleading results (for example, by using data mining tools to learn rules). In the worst case, such misleading rules could be used for critical purposes (like diagnosis). Therefore, for many situations it is safer if the sanitization process place unknown values instead of false values. This obscures the sensitive rules, while protecting the user of the data from learning “false” rules (Saygin 2001).

Blocking based algorithms are similar to distortion based algorithms. These algorithms run on strategy that is based on reducing the support and confidence of rules. The goal of the algorithms are to obscure a given set of sensitive rules by replacing known values with unknowns while minimizing the side effects on non-sensitive rules (Saygin 2001).

In order to hide a rule, decreasing the support of item set or decreasing the confidence of the rule below the minimum thresholds is adequate. To accomplish this, again using the bitmap notation, we replace actual values with “?” so uncertainty for support and confidence is increased. A simple example is shown in Figure 2.4. The values of item C in second transaction and item A in third transaction are changed. By doing this the support and the confidence values of the rule A→C is blurred. It can be any value between the minimum and maximum thresholds.

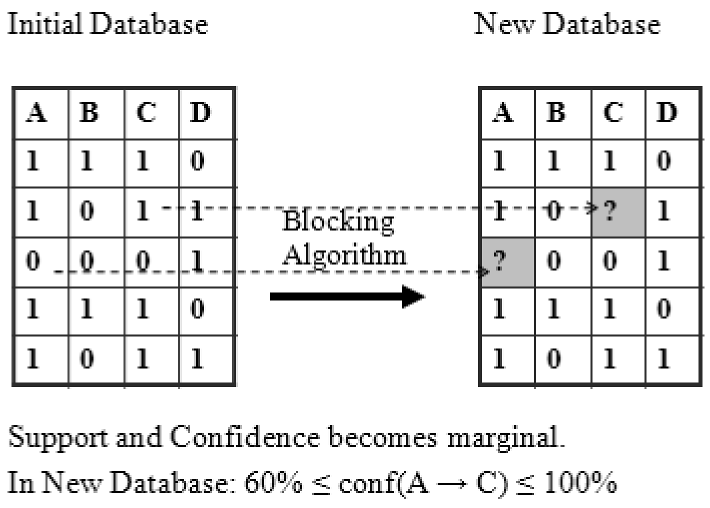

Initial Database
New Database
Blocking Algorithm
Support and Confidence becomes marginal.
In New Database: 60% ≤ conf(A → C) ≤ 100%

Figure 2.4. Blocking based rule hiding

2.5. Frequent Itemset Hiding

In rule hiding, sensitive knowledge which can be mined from the database is hidden while non-sensitive knowledge can still be mined (Verykios 2004a). Rule hiding research focuses on association rule hiding and frequent itemset hiding. It refers to the process of modifying the original database in such a way that certain sensitive association rules or frequent itemsets disappear without seriously affecting the data and non-sensitive rules or itemsets. (Aggarwal 2008) is most wide ranging source about PPDM; it divides association rule hiding approaches as heuristic, border based and exact approaches.

Exact approaches give optimal solution and have no side effect, on the other hand have much computational cost. In (Gkoulalas-Divanis 2006 and 2008) exact techniques are given which formulate sanitization as constraint satisfaction problem and solve these by integer programming.

Border based approaches uses border theory (Manilla 1997). In (Sun 2005, 2007 and Mousakides 2008) border based techniques for association rule hiding are proposed. The idea behind these approaches is that the elements on the border are boundary to the infrequent itemsets. During hiding process, instead of considering non-sensitive frequent itemsets, they are focused on preserving the quality of the border.

Heuristic approaches uses heuristics for modifications in the database. These techniques are efficient, scalable and fast algorithms however they do not give optimal solution and may have side effects. These techniques based on support and confidence decreasing. There are two types of techniques: distortion and blocking. Distortion techniques select transactions which hold sensitive itemsets and then selected items are deleted from transaction and database is modified. Blocking techniques replaces items with unknown values instead of deletion of items to modify database. The first algorithm is based on support reduction (Atallah 1999). In (Veykios 2004b) five algorithms are proposed based on hiding strategies. Not only itemsets but also rules are considered through hiding in the algorithms.

A framework for frequent itemset hiding is proposed in (Oliveira 2002). Algorithms require two database scans. At first scan the inverted file index is created and at second scan items are deleted from selected transactions. In (Saygin 2001) blocking is used instead of distortion of items in the database. The idea behind this approach is that sometimes replacing false values may have bad consequences. The aim in the algorithms is hide given sensitive rules by replacing unknown values and minimize side effects on non-sensitive rules.

Many association rule hiding algorithms are Apriori (Agrawal 1994) based and needs multiple database scans to find support of sensitive itemsets because these techniques require data mining done prior to the hiding process. In (Wang 2008) a tree structure P-tree (Huang 2002) which is similar to FP tree (Han 2000) is used to store information about database. This algorithm gets predictive item and sanitize informative rule set which is the smallest set of association rules that makes the same prediction as the entire rule set. The algorithm does not need data mining to be done before hiding process and does not scan database many times.

CHAPTER 3

FREQUENT ITEMSET MINING AND FREQUENT ITEMSET HIDING

3.1. Introduction

Among many techniques in data mining association rule mining is one of the most important and well researched one. It aims to extract interesting correlations, frequent patterns, associations or casual structures among sets of items in the transactional databases or other data repositories (Kotsiantis 2006). Association rule mining process consists of two steps: finding frequent itemsets and generating rules. The main concentration of most association rule mining algorithms is to find frequent itemsets in an efficient way to reduce the overall cost of the process. The rules are generated from frequent itemsets. Therefore, usually frequent itemset or pattern mining term is used instead of association rule mining. If you consider market basket data, the purchasing of product(X) and product(Y) frequently together represents a frequent itemset. An itemset is a set of items in the database. Frequent itemset is an itemset of which support value (percentage of transactions in the database that contain both X and Y) is above the threshold defined as minimum support. From found itemsets association rules can be produced.

Data mining became popular in last decades by the help of increase in abilities of computers and collection of large amount of data however; it is a challenge to extract knowledge without violating data owner's privacy (Grossman 1998, Kantardzic 2002, Dunham 2002, Han 2005, Yang 2006 and Zhang 2007). Privacy preserving data mining (PPDM) come up with the need for protecting sensitive data or knowledge to conserve privacy while data mining techniques can still be applied efficiently.

PPDM has two aspects as input and output privacy. To protect input privacy, data hiding techniques are applied such that data mining can still be done without violating private individual data. To protect output privacy, rule or knowledge hiding techniques are applied. These techniques ensure that private rules or patterns which can be extracted from given data are hidden while remaining ones can still be mined. Rule hiding is concentrated on association rules and frequent itemsets. Algorithms operate to distort items in transactions of database in such a way that as many as sensitive itemsets or association rules hidden and as many as non-sensitive itemsets or association rules extracted.

Many approaches for frequent itemset hiding are Apriori based and needs multiple database scans. Besides, these techniques require pre-mining to calculate support of sensitive itemsets. Therefore, it is decided to propose such algorithm that it avoids multiple database scans and pre-mining of frequent patterns. Matrix Apriori, a two database scan frequent itemset mining algorithm, is used for proposed itemset hiding algorithm. The approach does not require pre-mining and supports are calculated during hiding process. In addition, four distortion strategies are proposed which use pattern lengths instead of transaction lengths for item selection.

In this chapter, firstly, frequent itemset mining is introduced and two algorithms FP-Growth and Matrix Apriori are explained and demonstrated to be self contained for itemset hiding section. Following, proposed Matrix Apriori based frequent itemset hiding algorithms are explained and an example hiding case is given.

3.2. Frequent Itemset Mining

Association rule mining was first introduced by (Agrawal 1993), and in (Agrawal 1994) the popular Apriori algorithm was proposed. It computes the frequent itemsets in the database through several iterations. Each iteration has two steps: candidate generation and candidate selection (Kantardzic 2002). Database is scanned at each iteration. Apriori algorithm uses large itemset property: any subset of a large itemset must be large. Candidate itemsets are generated as supersets of only large itemsets found at previous iteration. This reduces the candidate itemset number. Among many versions of Apriori (Savasere 1995 and Toivonen 1996), FP-Growth has been proposed in association rule mining research with the idea of finding frequent itemsets without candidate generation (Han 2000). FP-Growth uses tree data structure and scans database only twice showing notable impact on the efficiency of itemset generation phase. Lately an approach named Matrix Apriori is introduced with the claim of combining positive properties of Apriori and FP-Growth algorithms (Pavon 2006). In this approach, database is scanned twice as in the case of FP-Growth and matrix structure used is simpler to maintain. Although it is claimed to perform better than FP-Growth, performance comparison of both algorithms are lately shown in (Yıldız 2010) which is the first step of this thesis. Next two subsections explains and demonstrates FP-Growth and Matrix Apriori algorithms.

3.2.1. FP-Growth Algorithm

The FP-Growth method adopts a divide and conquer strategy as follows: compress the database representing frequent items into a frequent-pattern tree, but retain the itemset association information, and then divide such a compressed database into a set of condition databases, each associated with one frequent item, and mine each such database (Han 2000).

The algorithm is given in Figure 3.1. Firstly database is read and frequent items are found which are the items are occurring in transactions less than minimum support. Secondly database is read again to build FP-tree. After creating the root, every transaction is read in an ordered way and pattern of frequent items in the transaction is added to FP-tree and nodes are connected to frequent items list and each other. This interconnection makes frequent pattern search faster avoiding the traversing of the entire tree. When considering the branch to be added for a transaction, the count of each node along a common prefix is incremented by 1. Nodes of same items are interconnected where most left one is connected to item in frequent items list. If the prefix of branch to be added does not exists then it is added as a new branch to root. After constructing the tree the mining proceeds as follows. Start from each frequent length-1 pattern (frequent item), construct its conditional pattern base, then construct its conditional FP-tree and perform mining recursively on such a tree. The support of a candidate (conditional) itemset is counted traversing the tree. The sum of count values at least frequent item’s nodes (base node) gives the support value.

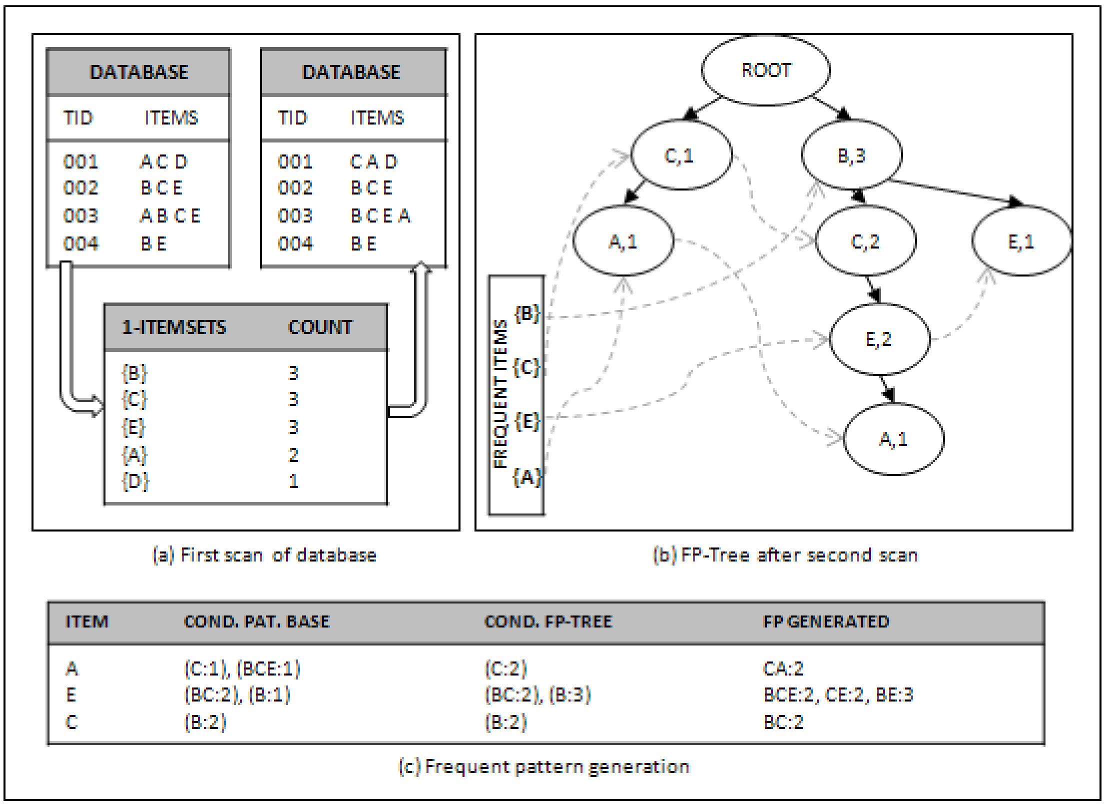
```
INPUT: Database D, minimum support minsup

OUTPUT: Frequent itemsets FIs

BEGIN
//first scan of database
1   Read D
2   Count item occurrences
3   Exclude items below minsup and generate frequent items list FIlist in descending order
//second scan of database and building FP-tree
4   Create root of FP-tree
5   Read D
6   FOR every transaction read frequent items in descending order as in FIlist
7       If first item has already connected to root increase node's count and set as    currentnode
8       Else add new node to root and set as currentnode and connect to FIlist
9       FOR remaining items
10          IF item exists in one of child nodes set it as current node and increase count
11          ELSE add item as new node and set it as current node
12              IF there is node connected to FIlist connect to last node of that item
13              ELSE connect to FIlist
14      END
15  END
//find frequent itemsets
16  FROM least frequent item TO most frequent item in FIlist
17      FROM I=2 TO FIlist length
18          Generate candidate length(I) itemset by extending item with other items in FIlist
19          Count support by traversing base nodes of item
20          IF support>=minsup give itemset as output
21              Continue extending itemset and go to line 18
22          ELSE stop extending and go to line 17
23      END
24  END
END
```

Figure 3.1. FP-Growth algorithm

In Figure 3.2 FP-Growth algorithm is visualized for an example database with minimum support value 2 (50%). First, a scan of database derives a list of frequent items in descending order (see Figure 3.2a). Then in second step FP-tree is constructed (see Figure 3.2b). Step by step creation of FP-tree is given in Appendix B Figure B.1. In Figure 3.2b, we can see the transactions and the tree constructed. The frequent pattern generation process is demonstrated in Figure 3.2c. Details of pattern finding for item "A" is given in Appendix B Figure B.2.



DATABASE	
TID	ITEMS
001	A C D
002	B C E
003	A B C E
004	B E

DATABASE	
TID	ITEMS
001	C A D
002	B C E
003	B C E A
004	B E

1-ITEMSETS	COUNT
{B}	3
{C}	3
{E}	3
{A}	2
{D}	1

(a) First scan of database

(b) FP-Tree after second scan

ITEM	COND. PAT. BASE	COND. FP-TREE	FP GENERATED
A	(C:1), (BCE:1)	(C:2)	CA:2
E	(BC:2), (B:1)	(BC:2), (B:3)	BCE:2, CE:2, BE:3
C	(B:2)	(B:2)	BC:2

(c) Frequent pattern generation

Figure 3.2. FP-Growth example

3.2.2. Matrix Apriori Algorithm

Matrix Apriori (Pavon 2006) is similar to FP-Growth in the database scan step. However, the data structure build for Matrix Apriori is a matrix representing frequent items (MFI) and a vector holding support of candidates (STE). The search for frequent patterns is executed on this two structures, which are easier to build and use compared to FP-tree.

In Figure 3.3 Matrix Apriori algorithm is given. Firstly database is read and frequent items are found which are the items are occurring in transactions less than minimum support. Secondly database is read again to build MFI and STE. Following this, a second scan on database is executed. During the scan the MFI and STE is built as follows. Each transaction is read. If the transaction has any item that is in the frequent item list then it is represented as “1” and otherwise “0”. This pattern is added as a row to MFI matrix and its occurrence is set to 1 in STE vector. While reading remaining transactions if the transaction is already included in MFI then in STE its occurrence is incremented. Otherwise it is added to MFI and its occurrence in STE is set to 1. After reading transactions, the MFI matrix is modified to speed up frequent pattern search. For each column of MFI, beginning from the first row, the value of a cell is set to the row number in which the item is "1". If there is not any "1" in remaining rows then the value of the cell is set to "1" which means down to the bottom of the matrix, no row contains this item. After constructing the MFI matrix, finding patterns is simple. Beginning from the least frequent item, create candidate itemsets and count its support value. The support value of an itemset is the sum of the items at STE of which index are rows where all the items of the candidate itemset are included in MFI's related row.

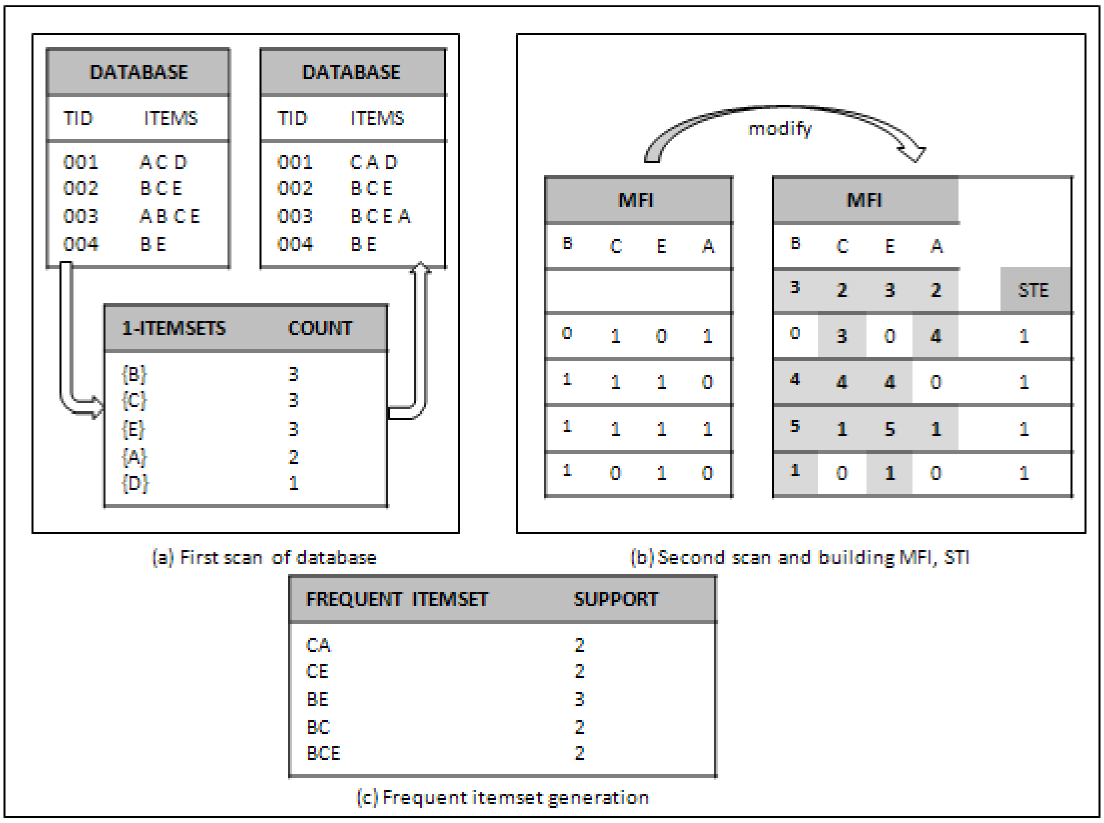
```
INPUT: Database D, minimum support minsup

OUTPUT: Frequent itemsets FIs

BEGIN
//first scan of database
1  Read D
2  Count item occurrences
3  Exclude items below minsup and generate frequent items list FIlist in descending order
//second scan of database and building MFI, STE
4  Create MFI (clolumns represents frequent items in descending order) and STE (cells give pattern
   occurences)
5  Leave one blank row for MFI
6  Read D
7  Add pattern of frequent items for first transaction  as a new row (if item exists put 1 else put 0)
8  Set first cell of STE to 1
9  FOR every transaction read patterns of frequent items
10     IF pattern exists in MFI increase count of related cell in STE
11     ELSE add pattern as a new row to MFI and set new cell of STE to 1
12 END
//modify MFI
13 FOR every column of MFI beginning from the first
14     UNTIL no 1s left in remaining rows for that column
15         Write the number of next row containing 1 for that column
16         Jump to row of next 1
17     END
18 END
//find frequent itemsets
19 FROM least frequent item to most frequent item in FIlist
20     FROM l=2 to length of FIlist
21         Generate candidate length(l) itemset by extending item with other items in FIlist
22         Count support by traversing MFI for columns of items in itemset and STE
23         IF support>=minsup give itemset as output
24             Continue extending itemset and go to line 21
25         ELSE stop extending and go to line 20
26     END
27 END
END
```

Figure 3.3. Matrix Apriori algorithm

In Figure 3.4, Matrix Apriori algorithm is demonstrated. The example database is the same database used in previous section and minimum support value is again 2 (%50). Firstly, a database scan to determine frequent items is executed and a frequent items list is obtained. The list is in descending order (see Figure 3.4a). A second scan is done to build MFI and STE. Following MFI is modified to make frequent pattern search faster (see Figure 3.4b). Details of constructing MFI and STE are given in Appendix B Figure B.3. Frequent itemsets found as explained before and can be seen in Figure 3.4c. An example support counting for itemset "CA" is given in Appendix B Figure B.4.



DATABASE

TID	ITEMS
001	A C D
002	B C E
003	A B C E
004	B E

DATABASE

TID	ITEMS
001	C A D
002	B C E
003	B C E A
004	B E

1-ITEMSETS	COUNT
{B}	3
{C}	3
{E}	3
{A}	2
{D}	1

(a) First scan of database

MFI

B	C	E	A
0	1	0	1
1	1	1	0
1	1	1	1
1	0	1	0

MFI

B	C	E	A	
3	2	3	2	STE
0	3	0	4	1
4	4	4	0	1
5	1	5	1	1
1	0	1	0	1

(b) Second scan and building MFI, STI

FREQUENT ITEMSET	SUPPORT
CA	2
CE	2
BE	3
BC	2
BCE	2

(c) Frequent itemset generation

Figure 3.4. Matrix Apriori example

3.2.3. Discussion on FP-Growth and Matrix Apriori Algorithms

It will be beneficial to give a short comparison of given algorithms with an example to show the execution of the algorithms. First scans of both algorithms are carried out in the same way. Frequent items are found and listed in order. During second scan, FP-Growth adds transactions to tree structure and Matrix Apriori to matrix structure. Addition of a transaction to the tree structure needs less control compared to matrix structure. For example, consider 2nd and 3rd transactions. Second transaction is added as a branch to the tree and as a row to the matrix. But addition of third transaction shows the difference. For tree structure we need to control only the branch that has the same prefix with our transaction. So addition of a new branch to node E is enough. On the other hand, for the matrix structure we need to control all the items of rows. If we find the same pattern then we increase the related item of STE. Otherwise we need to scan matrix till we find the same pattern. If we cannot find then a new row is added to matrix. It seems that building matrix needs more control and time, however, management of matrix structure is easier compared to tree structure.

Finding patterns for both algorithms need producing candidate itemsets and control. This is called conditional pattern base in FP-Growth and there is no specific name for Matrix Apriori. Counting support value is easy to handle in Matrix Apriori by sequentially top down sum of related rows of STE. However, in FP-Growth counting support is complex by traversing the tree, selecting related nodes and sum values in selected nodes.

3.3. Frequent Itemset Hiding

One of the prominence techniques in data mining is frequent itemset or association rule mining however; obtained outputs may cause violation of knowledge privacy. There may be some situations where knowledge extracted by rule mining algorithms includes rules or itemsets that should stay unrevealed. These itemsets are called sensitive itemsets. Itemset hiding intends to modify database in such a way that sensitive itemsets are hidden with minimum side effects on non-sensitive ones. The first study on rule hiding shows that sanitization of the database is NP-Hard and heuristic approaches are needed (Atallah 1999). Heuristic approaches are based on support and confidence reduction. Following studies propose algorithms for itemset hiding and association rule hiding respectively (Oliveira 2002 and Verykios 2004b). These algorithms distort items in the database. However, there may be such conditions that writing false values may cause problems. The approach used in (Saygin 2001) use unknown values instead of writing false values on the database.

Many itemset or rule hiding approaches are based on Apriori algorithm which needs multiple database scans and pre-mining of association rules. On the other hand FP-Growth algorithm, which has a better performance compared to Apriori, makes two database scans for finding frequent itemsets (Han 2000). The work presented in (Wang 2008) uses hiding algorithm based on P-tree (Huang 2002) similar to FP-tree of FP-Growth algorithm. They sanitize informative rules and eliminate need for pre-mining of association rules. Another, frequent itemset mining algorithm with two database scans is Matrix-Apirori explained in section 3.2.2. It is simpler than FP-Growth in terms of maintenance of the compact data structure and performs better (Yıldız 2010) which leads to propose itemset hiding algorithms. Proposed algorithms for frequent itemset hiding are explained and demonstrated in following subsections.

3.3.1. Matrix Apriori Based Frequent Itemset Hiding Algorithms

As displayed in Figure 3.5, proposed privacy preserving frequent itemset mining approach gets dataset *D,* sensitive itemsets *Ls* and minimum support *minsup* as input and returns sanitized dataset *Ds* with frequent itemsets which can be found from *Ds* as *FIs*. Sensitive itemsets are given without any knowledge if those itemsets are frequent or not. If any itemset given as sensitive is frequent in original database then it is hidden through itemset hiding process. Most hiding approaches first do mining and calculate support of all frequent itemsets then start hiding process. This has two disadvantages i) it might cause a privacy breech if the one performing hiding process is not trusted because all frequent itemsets are required to be known before the hiding process and ii) it requires pre-mining causing decrease in efficiency. Proposed approach ensures that user does not know whether given sensitive itemset was frequent in original dataset because frequent itemsets are found during hiding process and eliminates the need for pre-mining process.

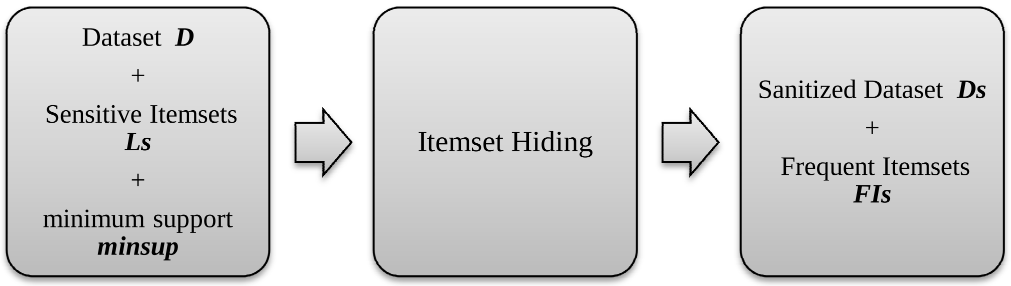


Figure 3.5. Sanitization framework

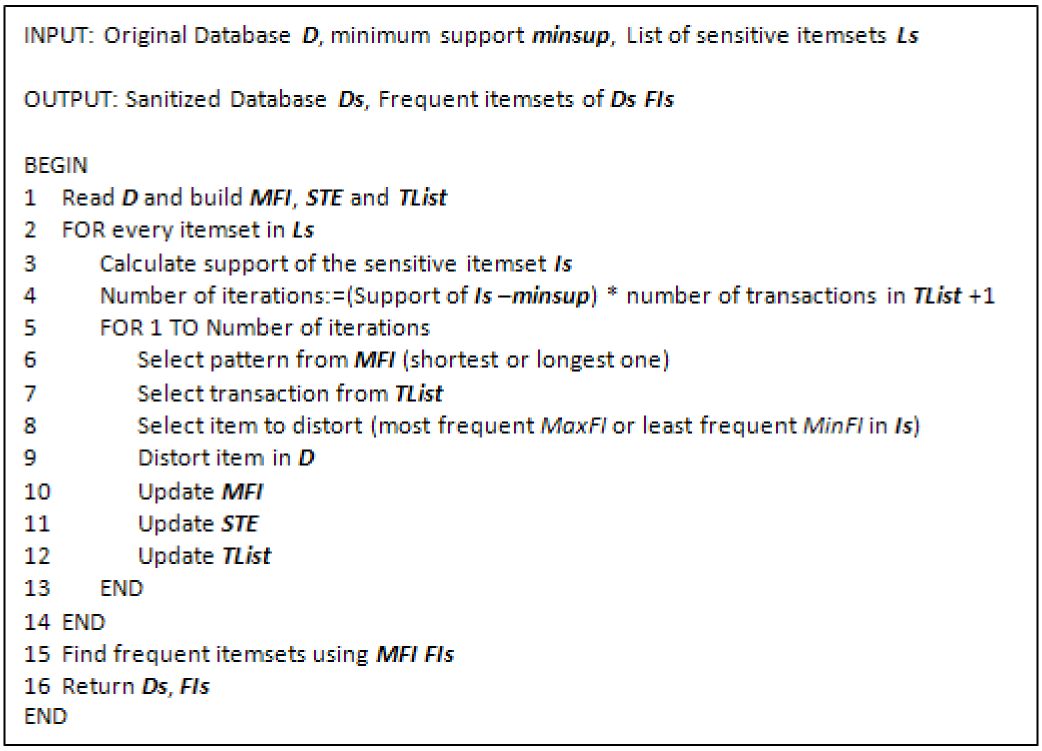
Itemset hiding process is based on the Matrix Apriori algorithm given in (Pavon 2006). Matrix Apriori is a frequent itemset mining algorithm without candidate generation and scans database only twice. At first scan, for the specified minimum support frequent items are found. At second scan, matrix data structure called MFI and support holder vector STE is build. For every transaction in database, the pattern consists of frequent items is read and added to MFI matrix and occurrences are updated on STE vector. Frequent itemset mining is done on this compact data structure which eliminates the need for database scan for itemset support counting.

Matrix Apriori algorithm is modified in proposed approach to have capabilities for itemset hiding (Figure 3.6). As stated in line 1, database *D* is read and matrix data structure *MFI* and *STE* is build. This time, while building *MFI* and *STE*, we also construct a transaction list as *TList* which keeps the transaction ids of transactions containing the itemset in each row of *MFI*. In proposed approach, transaction selection for modifying is done on *MFI* and database scan in order to find transaction is eliminated.

Between lines 2 and 14 for every itemset in sensitive itemsets list *Ls*, hiding process is run. Support value for sensitive itemset is calculated using *MFI* and *STE*. If the support of the itemset is above *minsup* then the number of iterations to hide itemset is calculated (line 4). This number indicates number of distortions to be done on the dataset to reduce the support of the sensitive itemset *Is* below *minsup*. Following this, at each iteration transaction to modify is selected (lines 6 and 7). There are two strategies for transaction selection. First one is to find shortest pattern in *MFI* that includes the sensitive itemset and select the last transaction from *TList*. Second strategy is to find longest pattern and select the transaction from *TList*. Most approaches use transaction length to decide transaction to modify. However, compact matrix structure of proposed approach has more valuable information as patterns of frequent items so length of the pattern is used instead of length of the transaction. This approach also eliminates the need for database access in choosing decision.

When transaction is selected we need to select an item of the transaction for distortion (line 8). There are two strategies for selection of item to distort: *maxFI* and *minFI*. Using *maxFI*, most frequent item of sensitive itemset is distorted on transaction. If *minFI* is used then least frequent item of sensitive itemset is distorted on transaction. Selected item is distorted in transaction (line 9), the distortion technique is replacing “1” with “0” in related cell. Matrix structure *MFI* is updated after distortion (line 10). We decrease the value of related row in *STE* (line 11) and delete transaction modified in that row of *TList* (line 12). By this way it is ensured that we have compact mirror of semi-sanitized dataset in *MFI*, *STE* and *TList* throughout the hiding process.

The selection and distortion process is repeated until the support of sensitive itemset *Is* is below *minsupport*. After sanitization of a *Is* the next itemset is read from *Ls* and sanitized. At final step (line 16) frequent itemsets *FIs* of sanitized dataset *Ds* are found using up-to-date *MFI* and *STE*.



```
INPUT: Original Database D, minimum support minsup, List of sensitive itemsets Ls

OUTPUT: Sanitized Database Ds, Frequent itemsets of Ds FIs

BEGIN
1  Read D and build MFI, STE and TList
2  FOR every itemset in Ls
3      Calculate support of the sensitive itemset Is
4      Number of iterations:=(Support of Is –minsup) * number of transactions in TList +1
5      FOR 1 TO Number of iterations
6          Select pattern from MFI (shortest or longest one)
7          Select transaction from TList
8          Select item to distort (most frequent MaxFI or least frequent MinFI in Is)
9          Distort item in D
10         Update MFI
11         Update STE
12         Update TList
13     END
14 END
15 Find frequent itemsets using MFI FIs
16 Return Ds, FIs
END
```

Figure 3.6. Itemset hiding algorithm

Now, let us explain an itemset hiding process using an example. Shortest pattern and most frequent item *maxFI* stragety is applied and itemset of BA is sensitive *(Is)*. In

Figure 3.7 sample database is given with MFI, STE and TList found in line 1. For *minsupport* value 3 (50%) 4 frequent itemsets (length 1 itemsets are not included) are found. These are CB, CA, CBA, BA. But remember that proposed approach does not need frequent itemset mining to be performed before hiding process.

TID	Items
T1	ABC
T2	ABC
T3	ABC
T4	AB
T5	AD
T6	CD

MFI			STE	TIDs
A	B	C		
2	2	2		
3	3	5	3	T1,T2,T3
4	1	0	1	T4
1	0	0	1	T5
0	0	1	1	T6

Figure 3.7. Database D and MFI, STE and TList for D

As in line 3, using MFI and STE support of BA is calculated to be 4 (66%). Since the *minsupport* value is 3 (50%), number of iterations to sanitize BA can be calculated as 2 (line 4). At first iteration shortest pattern that holds BA is found as third row of *MFI* and related transaction is T4 from *TList*. Most frequent item of sensitive itemset BA is A so it will be deleted from selected transaction (Figure 3.8). Meanwhile *STE* value of selected row is decreased and modified transaction id is deleted from the list. After deletion the new pattern B is added to matrix and T4 is added to transaction list which is now the sixth row of the matrix. At second iteration second row is selected as shortest and T3 is selected for modification. In Figure 3.8 sanitized database *Ds* and shows *MFI*, *STE*, *TList* after sanitization process is shown. Steps of execution are demonstrated in Appendix B Figure B.5.

TID	Items
T1	ABC
T2	ABC
T3	BC
T4	B
T5	AD
T6	CD

MFI			STE	TIDs
A	B	C		
2	2	2		
3	3	5	2	T1,T2
4	6	0	0	
1	0	0	1	T5
0	0	7	1	T6
0	7	0	1	T4
0	1	1	1	T3

Figure 3.8. Sanitized database Ds and MFI, STE and TList after itemset hiding process

After sanitization process we are able to find frequent itemsets for sanitized database using up-to-date matrix structure. Support values of itemsets are calculated as CB(50%), CA(33%), CBA(33%) and BA(33%). Support of itemset BA is now under *minsupport* and it is hidden. CBA is also hidden because it is a superset of BA. However, CA is now under minimum support and cannot be find as frequent although it was not sensitive. This is the side effect and CA is called lost itemset.

3.3.2. Discussion on Matrix Apriori Based Frequent Itemset Hiding Algorithms

In previous subsection, an example for hiding an itemset was given. spmaxFI algorithm was used. Now, spmaxFI and other three algorithms will be compared for the same case. Minimum support is 3 (50%) and itemsets can be found from the database are CB, CA, CBA, BA. Sensitive itemset is again BA.

Firstly, discuss results of spminFI algorithm. Shortest pattern is selected as third row of MFI and T4 is the transaction. B is the minimum of frequent items of sensitive itemset BA so it is deleted from transaction T4. Following new pattern of T4 is appended to MFI. Later second row is selected as it contains shortest pattern for BA. T3 is our transaction and again B is deleted. In Figure 3.9 sanitized database Ds and MFI, STE and Tlist after itemset hiding process are given. If we calculate new support for itemsets we will obtain CB(33%), CA(50%), CBA(33%) and BA(33%). BA and its superset CBA is hidden. However, CB is also hidden which was not sensitive itemset. It is lost itemset rather than CA in spmaxFI example.

TID	Items
T1	ABC
T2	ABC
T3	AC
T4	A
T5	AD
T6	CD

MFI			STE	TIDs
A	B	C		
2	2	2		
3	3	5	2	T1,T2
4	1	0	0	
6	0	0	1	T5
0	0	7	1	T6
7	0	0	1	T4
1	0	1	1	T3

Figure 3.9. Ds and MFI, STE and TList after itemset hiding with spminFI

Secondly, discuss results of lpmaxFI algorithm. Longest pattern is selected as second row of MFI and T3 is the transaction. A is the maximum of frequent items of sensitive itemset BA so it is deleted from transaction T3. Following new pattern of T3 is appended to MFI. In next iteration second row is selected as it contains longest pattern for BA. T2 is our transaction and again A is deleted. In Figure 3.10 sanitized database Ds and MFI, STE and TList after itemset hiding process are given. If we calculate new support for itemsets we will obtain CB(50%), CA(17%), CBA(17%) and BA(33%). BA and its superset CBA are hidden. However, CA is lost itemset since it is hidden although it was not sensitive itemset.

TID	Items
T1	ABC
T2	BC
T3	BC
T4	AB
T5	AD
T6	CD

MFI			STE	TIDs
A	B	C		
2	2	2		
3	3	5	1	T1
4	6	0	1	T4
1	0	0	1	T5
0	0	6	1	T6
0	7	7	1	T3
0	1	1	1	T2

Figure 3.10. Ds and MFI, STE and TList after itemset hiding with lpmaxFI

Lastly, discuss results of lpminFI algorithm. Longest pattern is selected as second row of MFI and T3 is the transaction. B is the minimum of frequent items of sensitive itemset BA so it is deleted from transaction T3. Following new pattern of T3 is appended to MFI. In next iteration second row is selected as it contains longest pattern for BA. T2 is our transaction and again B is deleted. In Figure 3.11 sanitized database Ds and MFI, STE and TList after itemset hiding process are given. If we calculate new support for itemsets we will obtain CB(17%), CA(50%), CBA(17%) and BA(33%). BA and its superset CBA are hidden. However, CB is lost itemset since it is hidden although it was not sensitive itemset.

TID	Items
T1	ABC
T2	AC
T3	AC
T4	AB
T5	AD
T6	CD

MFI			STE	TIDs
A	B	C		
2	2	2		
3	3	5	1	T1
4	6	0	1	T4
6	0	0	1	T5
0	0	6	1	T6
7	0	7	1	T3
1	0	1	1	T2

Figure 3.11. Ds and MFI, STE and TList after itemset hiding with lpminFI

Results of itemset hiding using different algorithms are given in order to compare strategies and side effects encountered. All algorithms hide sensitive itemset successfully. Superset of sensitive itemset is also hidden by all algorithms. There are one lost itemset for all simulations however the lost itemset differs from algorithm to algorithm. Although side effects as number of lost itemset are same for all simulations there are dramatic decreases in support value of some itemsets for lpmaxFI and lpminFI algorithms. However, spmaxFI and spminFI algorithms seem to distribute this effect we do not have any support value under 33% for itemsets found previously frequent. In next chapter comparison of these algorithms is given and performance differences are clearly seen.

CHAPTER 4

PERFORMANCE EVALUATION

4.1. Introduction

Association rule and frequent itemset mining is popular technique in data mining. It was firstly introduced in (Agrawal 1993) and following it Apriori algorithm was proposed in (Agrawal 1994). The large itemset property saying “any subset of large itemset must be large” made Apriori so popular that it boosted data mining research. However, it has a bottleneck that for generating candidate itemsets Apriori scans database several times. FP-Growth (Han 2000) come up with the idea of eliminating database scan for candidate itemset generation and testing. It uses a compact data structure called FP-tree which can be thought as a summary of original database. FP-Growth scans database only twice and Matrix Apriori (Pavon 2006) is another algorithm that scans database only twice. Instead of tree Matrix Apriori deploys a matrix structure which speeds up the search for frequent itemsets. Although it is claimed to be faster than FP-Growth there is no work showing performances. This is done as the first part of thesis study.

Knowledge extracted by frequent itemset mining may cause privacy problems if some itemsets are sensitive which means they must be remain unrevealed. Privacy preserving data mining techniques for itemset mining are developed to come over this problem. They simply distort items in transactions of database and prevent sensitive itemsets to be exracted. Many hiding techniques are Apriori based. Therefore, multiple database scan is required as mentioned above. In (Wang 2008), a sanitization approach is proposed using P-tree which is similar to FP-tree. This technique eliminates pre-mining needed for sensitive itemset support calculation. Being inspired from this study, a frequent itemset hiding approach based on Matrix Apriori is proposed. Matrix Apriori is a two database scan frequent itemset algorithm using a compact data structure. With four different strategies algorithm is varied. The strategies differ from existing approaches in the way of selecting distorted item. Pattern length information obtained from matrix data structure is used instead of transaction length.

This chapter presents the results and discussions of performance evaluations. All applications are developed using Lazarus IDE (9.28.2). Firstly, two frequent itemset mining algorithms working without candidate generation FP-Growth and Matrix Apriori are compared. This study formed the basis for proposing frequent itemset hiding algorithms using Matrix Apriori. Secondly, four algorithms proposed for itemset hiding are compared. This study gives not only comparison of proposed algorithms but also effects of hiding process for different cases.

4.2. Comparison of Two Frequent Itemset Mining Algorithms

In this section, Matrix Apriori and FP-Growth algorithms which were discussed in previous chapter are compared. ARTool dataset generator (Cristofor 2002) is used for our synthetic datasets. Two case studies analyzing the algorithms are carried out step by step using two synthetic datasets generated in order i) to see their performance on datasets having different characteristics, ii) to understand the causes of performance differences in different phases. In order to keep the system state similar for all test runs, we assured all back-ground jobs which consume system resources were inactive. It is also ensured that test runs give close results when repeated.

4.2.1. Simulation Environment for Frequent Itemset Mining Evaluations

The test runs are performed on a computer with 2.4 GHz dual core processor and 3 GB memory. At each run, both programs give results about data mining process (see Appendix C for Figure C.1 and Figure C.2). These are

- time cost for first scan of database,
- number of frequent items found at first scan of database,
- time cost for second scan of database and building the data structure,
- time cost for finding frequent itemsets,
- number of frequent itemsets found after mining process,
- total time cost for whole data mining process.

Although real life data has different characteristics from synthetically generated data as mentioned in (Zheng 2001), synthetic data is used since the control of parameters were easily manageable. In (Omari 2008), the drawbacks of using real world data and synthetic data and comparison of some dataset generators are given. The reason of using synthetic data was to have datasets with different characteristics as representing different domain needs. Synthetic databases are generated using ARtool software (Cristofor 2002).

In the following subsections, performance analysis on the algorithms for two case studies is given. For the generated data sets, it is aimed to observe how change of minimum support affects the performance of algorithms. The algorithms are compared for six minimum support values in the range of 15% and 2,5%.

4.2.2. Case 1: Database of Long Patterns with Low Diversity of Items

A database is generated for having long patterns and low diversity of items where number of items=10000, number of transactions=30000, average size of transactions=20, average size of patterns=10. Number of frequent items is given in Figure 4.1 decrease in minimum support increases the number of frequent items from 16 to 240.

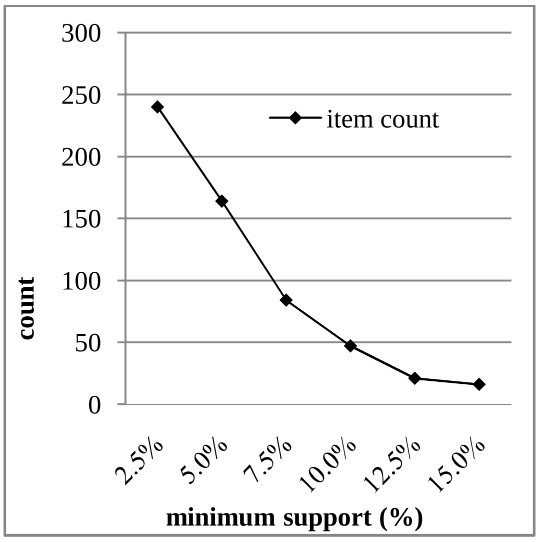


Figure 4.1. Number of frequent items for Case 1

Number of frequent itemsets is given in Figure 4.2 while minimum support value is varied. It is clear that decrease in minimum support increases the number of frequent itemsets from 1014 to 198048.

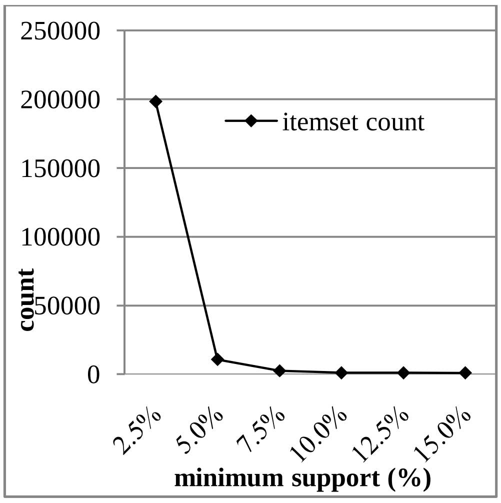


Figure 4.2. Number of frequent itemsets for Case 1

The total performance of Matrix Apriori and FP-Growth is demonstrated in Figure 4.3. It is seen that their performance is identical for minimum support values above 7,5%. On the other hand below 7,5% minimum support value Matrix Apriori performs clearly better such that at 2,5% threshold it is 230% faster.

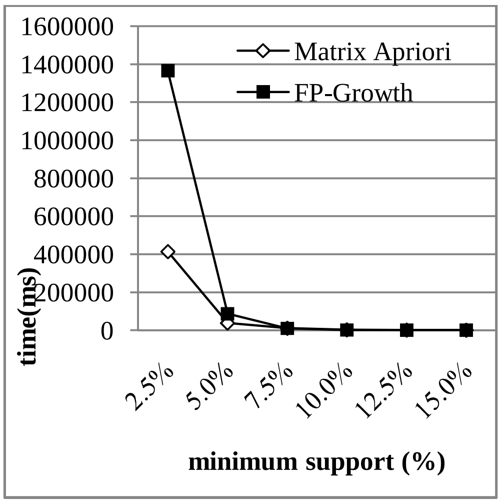


Figure 4.3. Total performance for Case 1

The reason of FP-Growth's falling behind at total performance can be understood by looking at the performance of phases of evaluation. First phase performances of algorithms demonstrated in Figure 4.4 showed that building matrix data structure of Matrix Apriori needs 20% to 177% more time compared to building tree data structure of FP-Growth. First phase of Matrix Apriori shows similar pattern with the number of frequent items demonstrated in Figure 4.1.

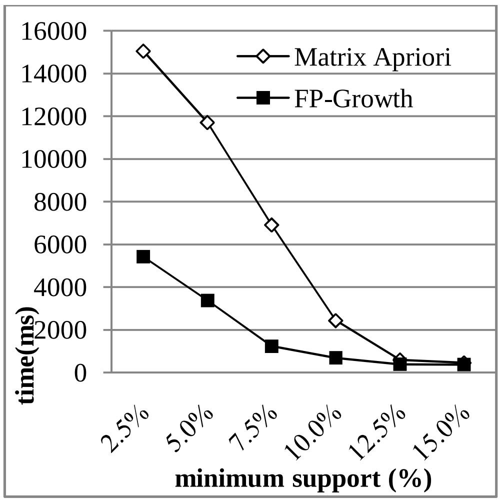


Figure 4.4. First phase performance for Case 1

The second phase of evaluation is finding frequent itemsets. As displayed in Figure 4.5 Matrix Apriori is faster at minimum support values below 10%. Although at 10% threshold, FP-Growth is 20% faster, Matrix Apriori is 240% faster at 2,5% threshold. As its expected, performance of second phases are related to number of frequent itemsets (see Figure 4.2).

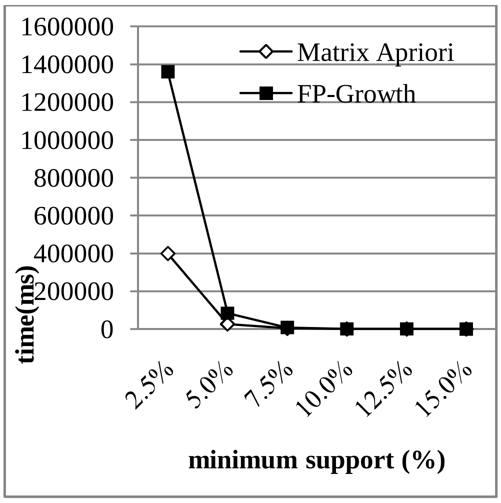


Figure 4.5. Second phase performance for Case 1

Our first case study showed that Marix Apriori performed better with decreasing threshold values for given database.

4.2.3. Case 2: Database of Short Patterns with High Diversity of Items

A database is generated for short patterns and high diversity of items using the parameters where number of items=30000, number of transaction=30000, average size of transactions=20, average size of patterns=5. The change of frequent items is given in Figure 4.6. Frequent items found changes from 58 to 127 with decreasing minimum support values.

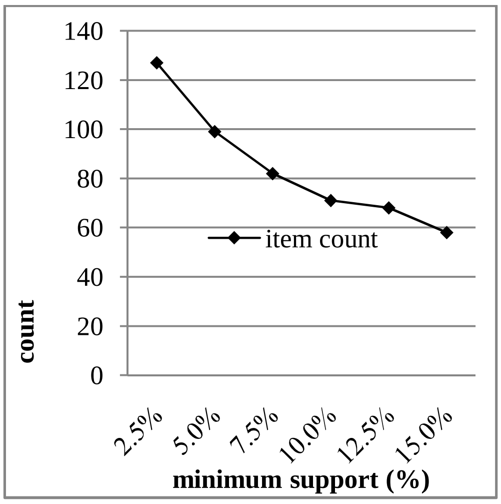


Figure 4.6. Number of frequent items for Case 2

The change of frequent itemsets count is given in Figure 4.7. While minimum support increases, frequent itemsets found changes from 254 to 71553.

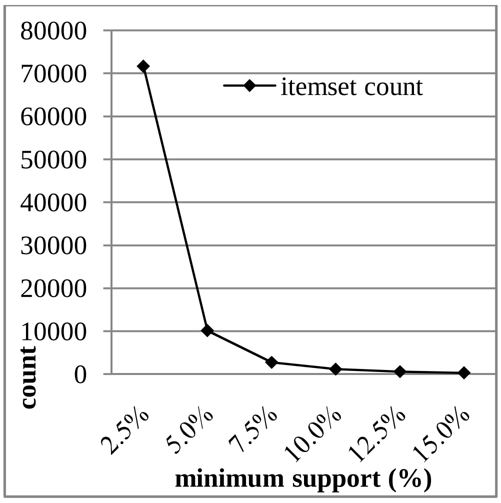


Figure 4.7. Number of frequent itemsets for Case 2

The total performance of both algorithms is given in Figure 4.8. Increase in minimum support decreases runtime for both algorithms. For minimum support values 12,5% and 15% FP-Growth performed faster by up to 56%. However, for lower minimum support values Matrix Apriori performed better up to 150%.

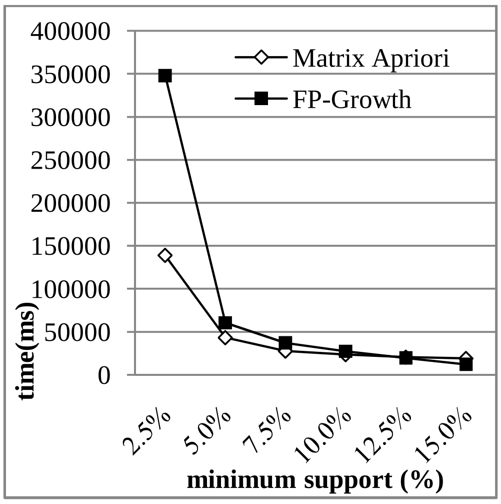


Figure 4.8. Total performance for Case 2

First phase performance of algorithms is demonstrated in Figure 4.9. FP-Growth is observed to have better first phase performance.

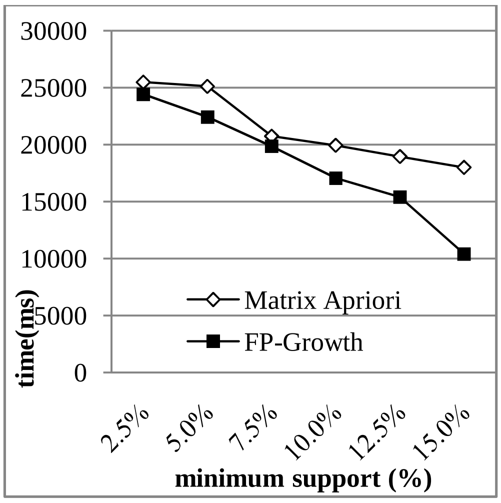


Figure 4.9. First phase performance for Case 2

The second phase evaluation of algorithms as it is given in Figure 4.10 shows that Matrix Apriori performed better at all threshold values and the performance gap increases with decreasing threshold. This difference varies between 71% and 185%.

Second phase performances of algorithms are related to number of frequent itemsets found like it was in first case study.

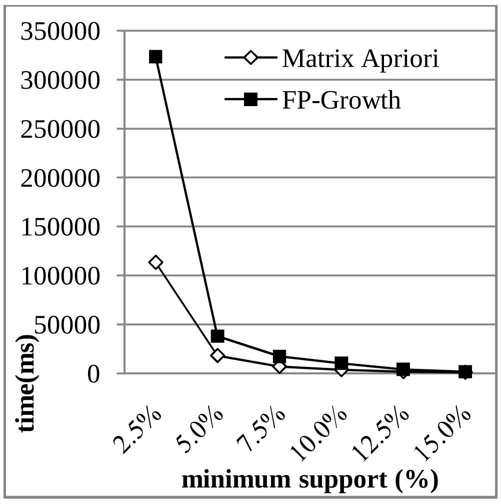


Figure 4.10. Second phase performance for Case 2

4.2.4. Discussion on FP-Growth and Matrix Apriori Comparison

In this section, the performance of FP-Growth and Matrix Apriori algorithms are analyzed phase by phase, when minimum support threshold is changed. Two databases with different characteristics are used for our case studies. In both case studies, performances of algorithms are observed between minimum support values of 2,5% and 15%.

First case study is carried out on a database of long patterns with low diversity of items. It is seen that at 10%-15% minimum support values, performances of both algorithms are close. However, below 10% value, the performance gap between the algorithms becomes larger in favor of Matrix Apriori. Another point is that first phase of Matrix Apriori is affected from minimum support change more than FP-Growth. This is a result of increase in frequent items count. This increment affects building data structure step of Matrix Apriori dramatically. On the other hand, matrix data structure is faster leading to better total performance of Matrix Apriori.

Our second case study is performed on a database of short patterns with high diversity of items. It is seen that at 12,5%-15% minimum support values, performances of both algorithms are close. However, below 12,5% value, the performance gap between the algorithms becomes larger in favor of Matrix Apriori. It is seen that the impacts of having more items and less average pattern length caused both algorithms to have more runtime values compared to first case study. At 15% at first case study 1014 itemsets are found in 1031-1078 ms however at second case study 254 itemsets are found in 12172-19030 ms. In addition, for all threshold values first phase runtime values are higher in second case study.

Common points in both case studies are i) Matrix Apriori is faster at finding itemset phase compared to FP-Growth and slower at building data structure phase, ii) for threshold values below 10% Matrix Apriori is more efficient by up to 230%, iii) first phase performance of Matrix Apriori is correlated with number of frequent items, iv) second phase performance of FP-Growth is correlated with number of frequent itemsets.

4.3. Comparison of Matrix Apriori Based Frequent Itemset Hiding Algorithms

In this section, performance evaluation of four versions of our itemset hiding algorithms is given. These are spmaxFI (select ***s***hortest ***p***attern and ***max***imum of ***f***requent ***i***tems in the itemset), spminFI (select ***s***hortest ***p***attern and ***min***imum of ***f***requent ***i***tems in the itemset), lpmaxFI (select ***l***ongest ***p***attern and ***max***imum of ***f***requent ***i***tems in the itemset) and lpminFI (select ***l***ongest ***p***attern and ***min***imum of ***f***requent ***i***tems in the itemset). Two synthetic databases are used to see effect of different database size. The algorithms are executed on databases i) to see effect of increasing number of sensitive itemsets, ii) to see effect of increasing support of sensitive itemset. The effects observed are number of lost itemsets as side effect, time cost for hiding process and number of items distorted for hiding itemsets. During evaluations, it is ensured that the system state is similar for all test runs and results are checked for consistency.

4.3.1. Simulation Environment for Frequent Itemset Hiding Evaluations

Test runs are performed on a computer with 2.7 GHz dual core processor and 1 GB memory. At each run inputs are original database and sensitive itemsets where the outputs are sanitized database and frequent itemsets which can be mined from this sanitized database (see Appendix C for Figure C.3). Synthetic databases used in the evaluations are generated using ARtool software (Cristofor 2002). Two databases are used for evaluations are different in the number of transactions since the effects of the size of database on hiding process wanted to be compared. One database has 5000 transactions while number of items is 50 and average length of transactions is 5. Other database has 10000 transactions while number of items is 50 and average length of transactions is 5. Minimum support is defined as 2.5% for all evaluations and if no hiding is applied then 2714 frequent itemsets from 5k database and 5527 frequent itemsets from 10k database can be found.

4.3.2. Case 1: Increasing Number of Sensitive Itemsets

For both databases five of length three itemsets which are closest to 3.0% support are selected as sensitive itemsets. These itemsets are given in Table 4.1 below. Selected itemsets are mutual exclusive to ensure that one is not hidden by hiding process of previous itemsets. The aim of this study is to understand the effect of increasing the number of sensitive itemsets on itemset hiding. For each run next itemset in the table is added to the sensitive itemsets given to program. At first run itemset no 1 is given as sensitive, at second run itemset no 1 and itemset no 2 are given as sensitive and so on.

Table 4.1. Sensitive itemsets for Case 1

Itemset no	Itemsets for 5k database	Support (%)	Itemsets for 10k database	Support (%)
1	37 31 32	2.96%	36 20 6	3.00%
2	7 47 41	3.06%	50 13 10	3.01%
3	5 6 4	2.92%	33 49 42	2.93%
4	24 13 46	3.08%	29 14 11	3.07%
5	45 34 20	2.94%	39 41 18	2.95%

The side effect which is the number of lost itemsets for increasing number of sensitive itemsets is given in Figure 4.11 for 5k database and in Figure 4.12 for 10k database. In both databases number of lost itemsets is increased for all hiding algorithms while number of sensitive itemsets is increased. It is clear that spmaxFI (select shortest pattern and maximum frequent item of itemset) algorithm has least side effects. The difference reaches up to 100% at five sensitive itemsets case. What more can be inferred from this figure is that side effect is related to the characteristics of sensitive itemsets, not to the database size. Even for the best algorithm we come across higher number of lost itemsets for 5k database such that for 5 itemset hiding point 29 itemsets are lost in 5k database while 22 itemsets are lost in 10k database.

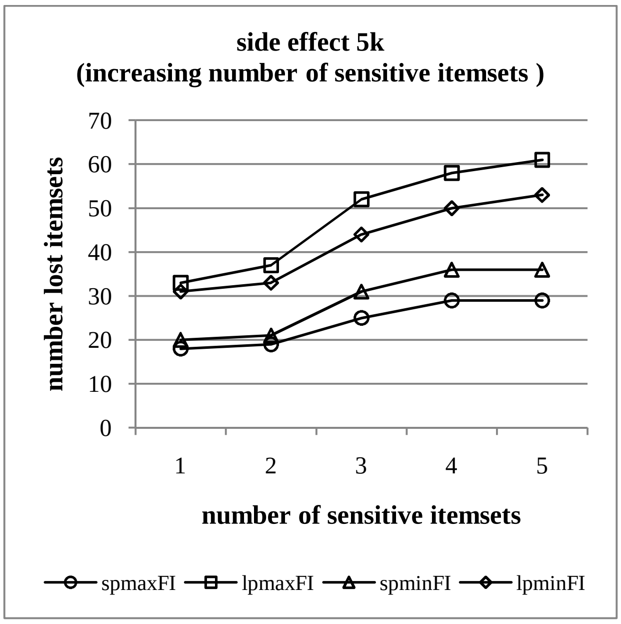


Figure 4.11. Side effect while increasing number of sensitive itemsets for 5k database

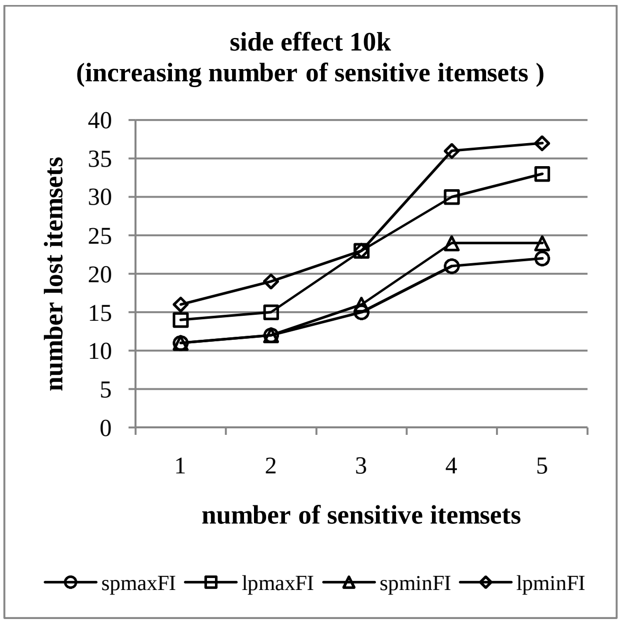


Figure 4.12. Side effect while increasing number of sensitive itemsets for 10k database

Time cost for itemset hiding is given in Figure 4.13 and Figure 4.14 for 5k database and 10k database respectively. Selecting shortest pattern seems as a better method no matter maximum or minimum frequent item is selected for distortion. Selecting longest pattern needs 20% to 100% more time compared to selecting shortest pattern method. It is clear from the figure that database size effects time to hide itemsets for same cases. The database size is doubled and time needed for hiding itemsets is increased more than 100%. The reason behind this is the cost of travelling on matrix to select pattern. It is clear that matrix size is bigger for 10k database compared to 5k database.

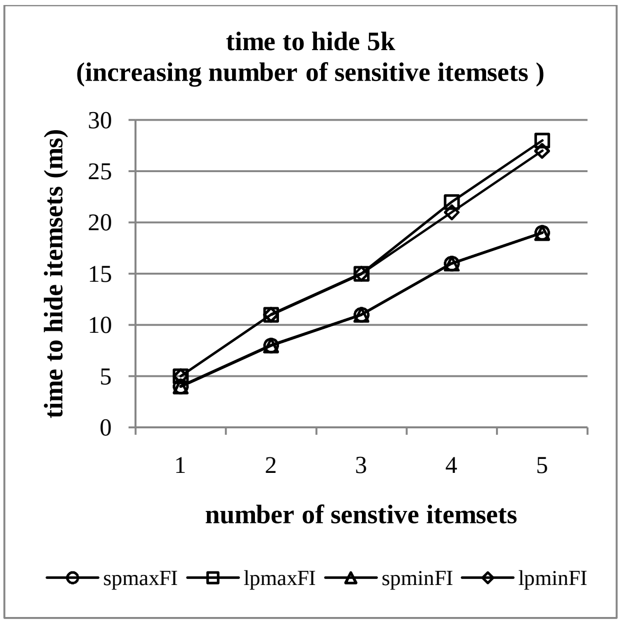


Figure 4.13. Time to hide itemsets while increasing number of sensitive itemsets for 5k database

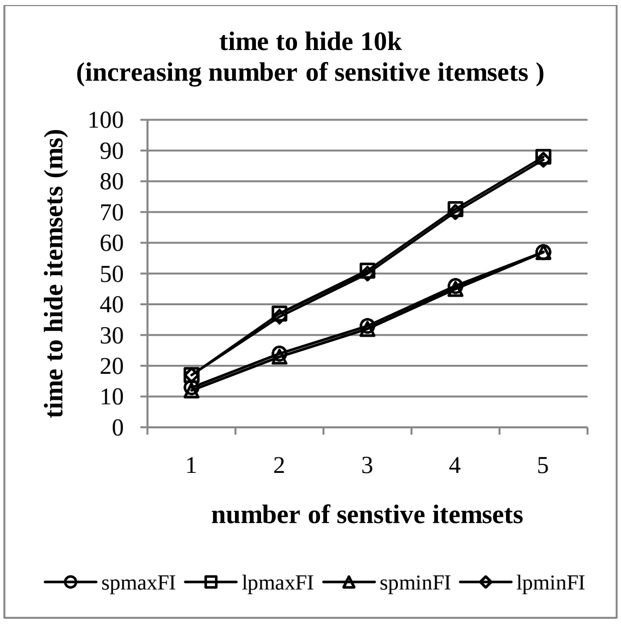


Figure 4.14. Time to hide itemsets while increasing number of sensitive itemsets for 10k database

The number of items to distort is similar for all algorithms. In Figure 4.15 and Figure 4.16 number of distorted items for increasing number of sensitive itemsets is given for 5k and 10k databases. Values are identical because calculation of number of items to distort is identical for all algorithms. For our case study this is also equal to number of transactions distorted since selected sensitive itemsets are mutual exclusive and there is no frequent itemset includes more than one sensitive itemset. Figures demonstrate that increasing the size of database will increase distorted items. This is a result of support count. For itemsets with same support value we have different support counts such that 3.0% support itemset in 5k database has 150 support count and itemset with same support value in 10k databse has 300 support count.

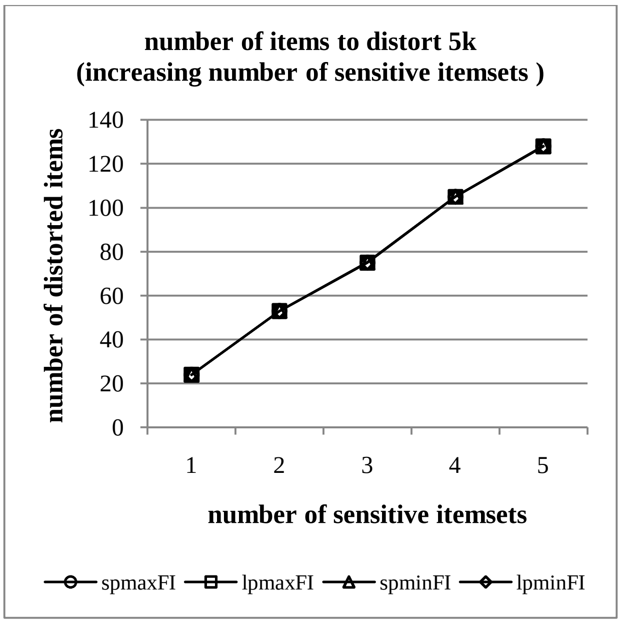


Figure 4.15. Number of items to distort while increasing number of sensitive itemsets for 5k database

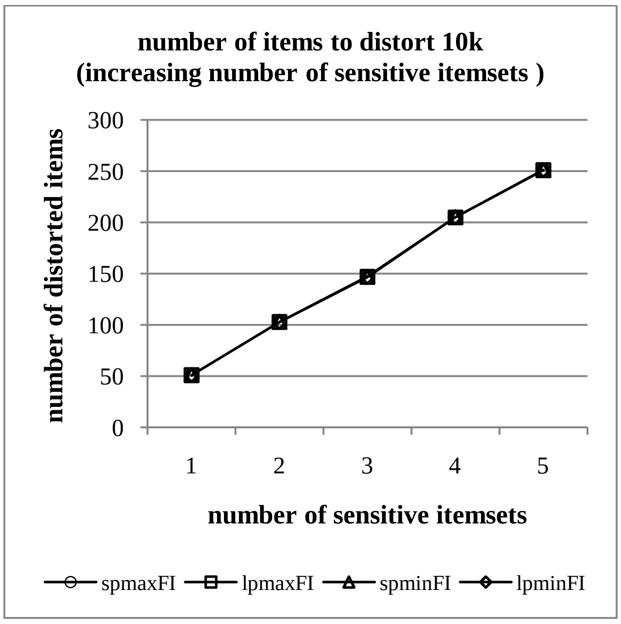


Figure 4.16. Number of items to distort while increasing number of sensitive itemsets for 10k database

4.3.3. Case 2: Increasing Support of Sensitive Itemsets

For both databases five of length three itemsets which have increasing support values between 3.0% and 5.0% are selected as sensitive itemsets. These itemsets are given in Table 4.2 below. The aim of this study is to understand the effect of increasing the support value of sensitive itemsets on itemset hiding. For each run next itemset in the table is selected as the sensitive itemsets given to program. At first run itemset no 1 is given as sensitive, at second run itemset no 2 is given as sensitive and so on.

Table 4.2. Sensitive itemsets for Case 2

Itemset no	Itemsets for 5k database	Support (%)	Itemsets for 10k database	Support (%)
1	37 31 32	2.96%	36 20 6	3.00%
2	18 28 47	3.50%	4 49 42	3.54%
3	14 17 24	4.00%	9 8 3	4.23%
4	28 47 4	4.50%	7 33 18	4.47%
5	46 20 4	5.00%	24 39 13	5.03%

The side effect of increasing support value for sensitive itemset is given in Figure 4.17 and Figure 4.18 for 5k and 10k databases. Like it was in first case study selecting shortest pattern has better performance. Selecting shortest pattern and maximum frequent item (spmaxFI) for distortion is the best algorithm to have less number of lost itemsets. The statement “side effect is related to characteristics of selected itemsets” which was written in the first case study is approved in this study. For example, in the 5k database using the strategy spmaxFI, for itemset no 2 the number of lost itemsets is 4 however, for itemset no 4 the number of lost itemsets is 57. One interesting point in the figure is the itemset no 3 for 10k database. This is a good example how pattern selection has effect on the results. Selecting shortest pattern results about 10 lost itemsets while selecting longest pattern results about 300 lost itemsets.

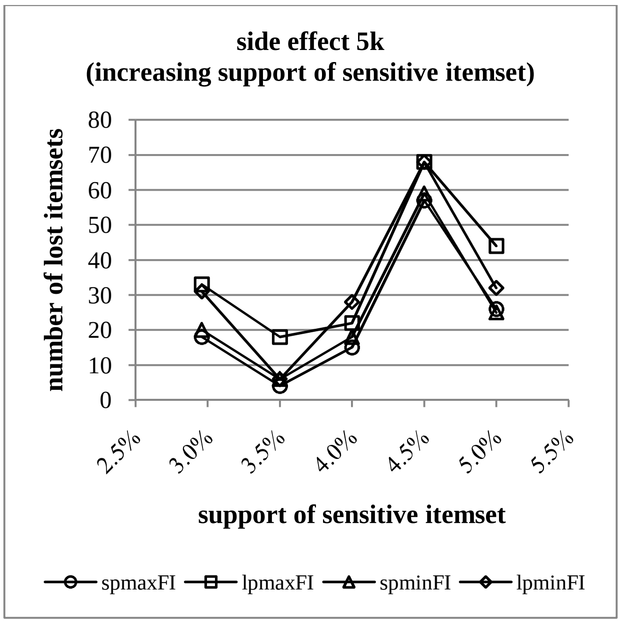


Figure 4.17. Side effect while increasing support of sensitive itemsets for 5k database

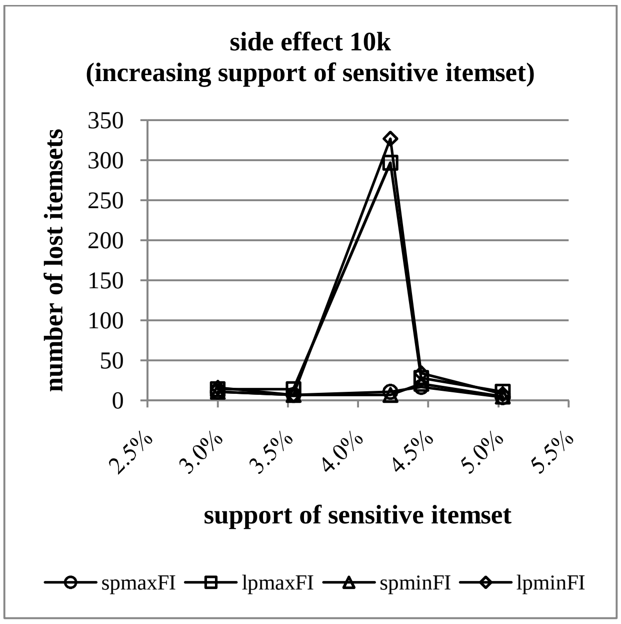


Figure 4.18. Side effect while increasing support of sensitive itemsets for 10k database

Time cost for itemset hiding is given in Figure 4.19 and Figure 4.20 for 5k and 10k databases. Selecting shortest pattern seems as a better method. In addition, spminFI algorithm is slightly faster than spmaxFI algorithm. It is clear from the figure that database size effects time to hide itemsets for same cases. The database size is doubled and like in the first case study time needed for hiding itemsets is increased more than 100%.

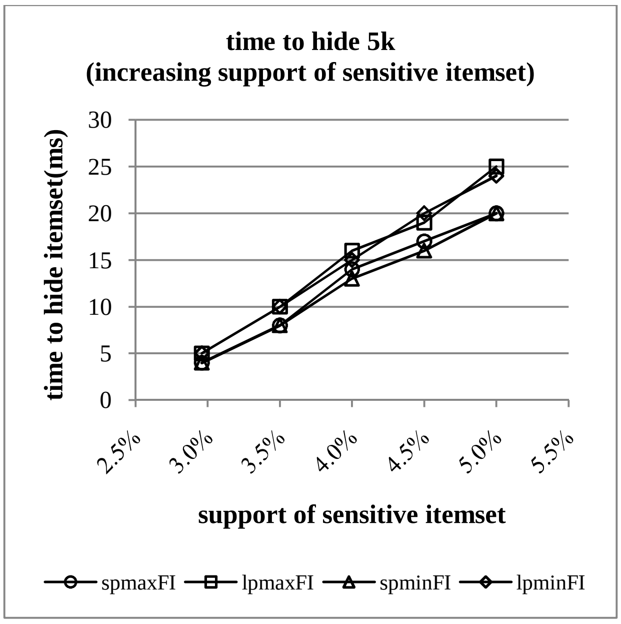


Figure 4.19. Time to hide itemsets while increasing support of sensitive itemsets for 5k database

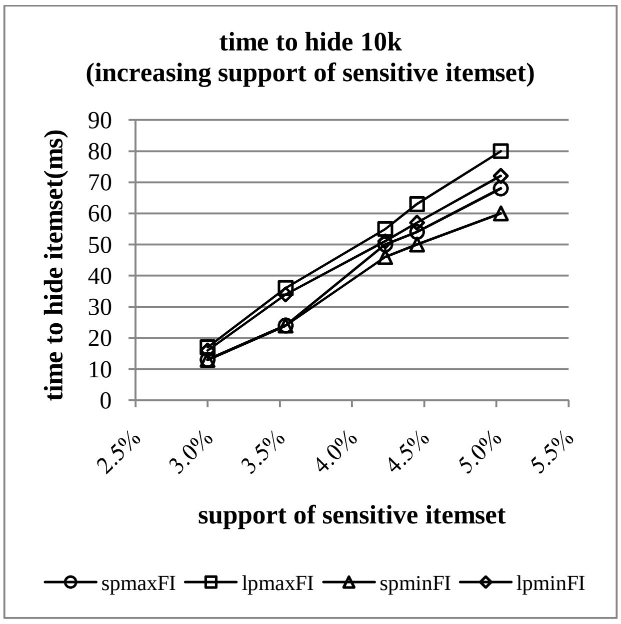


Figure 4.20. Time to hide itemsets while increasing support of sensitive itemsets for 10k database

The number of distortions is related to support count of sensitive itemsets as it was stated in previous part and so we will have increasing number of distorted items with increasing support.

4.3.4. Discussion on Matrix Apriori Based Frequent Itemset Hiding Algorithms

In this section, effects of spmaxFI, spminFI, lpmaxFI and lpminFI algorithms on number of lost itemsets, time for hiding process and number of distortions needed for hiding itemsets are analyzed. Two different databases are used to understand the effect of database size and two different set of sensitive itemsets to understand the effects of number of sensitive items and support of sensitive items. Simply comparing the algorithms, it is clear that spmaxFI algorithm has least side effects at any case. Another point is that selecting shortest pattern causes fewer side effects compared to selecting longest pattern and selecting shortest pattern needs less time for hiding. Number of distorted items is the same for all algorithms because items are distorted upon difference between support count of sensitive itemsets and minimum support count no matter which algorithm is used. The most important result from these studies is that side effect is related to characteristics of selected sensitive itemsets because subsets or supersets of that itemset are affected too.

CHAPTER 5

CONCLUSION

The rapid development in computer technology made it possible to collect, store huge amount of data and apply data mining. Data mining aims to discover knowledge or patterns from the data especially large databases. However, there may be such situations that private data may be under violation because of gained knowledge or extracted knowledge by itself contains some private knowledge. Privacy preserving data mining arise from the need for do data mining without violating privacy of data or knowledge. Data hiding and rule hiding are two branches of PPDM. Data hiding techniques preserve the private data while rule hiding techniques preserve the private rules or patterns. The aim of this thesis is to propose algorithms for privacy preserving frequent pattern mining. To achieve this, master study is divided into two steps.

In the first step, we benchmark and explain the FP-Growth and the Matrix Apriori frequent itemset mining algorithms that work without candidate generation. Since the characteristics of data repositories of different domains vary, each algorithm is analyzed using two different synthetic databases consisting of different characteristics, i.e., one database has long patterns with a low diversity of items and the other database has short patterns with a high diversity of items.

Our case studies indicate that the performances of the algorithms are related to the characteristics of the given data set and the minimum support threshold applied. When the performances of the various algorithms are considered, we noticed that in constructing a matrix data structure, the Matrix Apriori takes more time in comparison to constructing the tree structure for the FP-Growth. On the other hand, during finding itemsets phase we discovered that the matrix data structure is considerably faster than the FP-Growth at finding frequent itemsets--thus retrieving and presenting the results in a more efficient manner.

In the second step, by the help of gained knowledge of frequent itemset mining algorithms and benefits of Matrix Apriori a new algorithm for frequent itemset hiding is proposed with four different versions. The algorithm is based on Matrix-Aprirori which is an efficient algorithm since it eliminates multiple database scans by using a compact matrix structure as a summary of the original database. Each version uses different heuristic in selecting the transaction and the item in itemset for distortion; spmaxFI (select *s*hortest *p*attern and *max*imum of *f*requent *i*tems in the itemset), spminFI (select *s*hortest *p*attern and *min*imum of *f*requent *i*tems in the itemset), lpmaxFI (select *l*ongest *p*attern and *max*imum of *f*requent *i*tems in the itemset) and lpminFI (select *l*ongest *p*attern and *min*imum of *f*requent *i*tems in the itemset).

Main strengths of the algorithm are i) all versions work without pre-mining so privacy breech caused by the knowledge obtained by finding frequent itemsets in advance is prevented, ii) efficiency is increased since no pre-mining is required, iii) supports are found during hiding process and at the end sanitized database and frequent itemsets of this database are given as outputs so no post-mining is required, iv) the heuristics used transaction selection for distortion is from matrix data structure rather than transaction lengths eliminating the possibility of distorting more valuable data.

Performance evaluation study is done on different databases to show the efficiency of the versions of the algorithms while the size of the original database, the number of itemsets and the itemset supports change. The efficiency of four versions are observed as side effects (lost itemsets), time to hide itemsets and amount of distortion caused on the original database. Our findings are as follows. Among four versions, *spmaxFI* has better overall performance. The algorithms *spmaxFI* and *spminFI* are better in any case than *lpmaxFI* and *lpminFI* algorithms. Results show that side effect is related to given sensitive itemset. Neither support count nor database size is directly related to the number of lost itemsets. Time to hide sensitive itemset is a function of distortion and database size where distortion is related to support count.

In conclusion, this master thesis study shows that Matrix Apriori is a better performer compared to FP-Growth algorithm and it is an efficient way to use it for frequent itemset hiding. The main contributions of the study are i) sanitization framework eliminating the need for pre-mining and the database scan for post-mining after sanitization, ii) four versions of Matrix Apriori based frequent itemset hiding algorithms, iii) the idea of using pattern lengths for distortion strategy.

As a future study, the efficiency of Matrix Apriori algorithm can be increased and may be parallelized. In addition, for Matrix Apriori based frequent itemset mining algorithms we plan to carry out further evaluations on different databases, especially those having bigger average transaction lengths, to see the impact of having multiple sensitive itemsets in a single transaction on distortion. Secondly, the effect of the sensitive itemset sanitization order can be observed since in this work we chose mutually exclusive sensitive itemsets.

REFERENCES

Aggarwal, C., Yu, P. 2008. *Privacy-Preserving Data Mining: Models and Algorithms*. Springer.

Agrawal, R., Imieliński, T., Swami, A. 1993. Mining Association Rules Between Sets of Items in Large Databases. *ACM SIGMOD Record* (22): 207-216.

Agrawal, R., Srikant, R. 1994. Fast Algorithms for Mining Association Rules in Large Databases. *In Proceedings of the 20th International Conference on Very Large Data Bases*: 487-499.

Agrawal, R., Srikant, R. 2000. Privacy-Preserving Data Mining. *ACM SIGMOD Record* (29): 439-450.

Alves , R., Rodriguez-Baena, D., Aguilar-Ruiz, J. 2009. Gene association analysis: a survey of frequent pattern mining from gene expression data. *Briefings in Bioinformatics* (11): 210-224.

Atallah, M., Bertino, E., Elmagarmid, A., Ibrahim, M., Verykios, V. 1999. Disclosure Limitation of Sensitive Rules. *In Proceedings of the 1999 Workshop on Knowledge and Data Engineering Exchange*: 45-52.

Breiman, L., Freidman, J., Olshen, R. 1984. *Classification and regrassion trees*. Wadsworth.

Cristofor, L. 2002. ARtool Project. http://www.cs.umb.edu/~laur/ARtool/ (last access May 13 2010).

Domingos, P., Pazzani, M. 1997. On the optimality of the simple Bayesian classifier under zero-one loss. *Machine Learning* (29): 103-130.

Dunham, M. 2002. *Data Mining: Introductory and Advanced Topics.* Prentice Hall.

Duru, N. 2005. An Application of Apriori Algorithm on a Diabetic Database. *In Proceedings of the 9th International Conference Knowledge-Based Intelligent Information and Engineering Systems*: 398-404.

Ester, M., Kriegel, H.P., Sander, J. Xu, X. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise. *In the Proceedings of 2nd International Conference on Knowledge Discovery and DataMining*: 226-231.

Gkoulalas-Divanis, A., Verykios, V. 2006. An Integer Programming Approach for Frequent Itemset Hiding. *In Proceedings of the 15th ACM International Conference on Information and Knowledge Management*: 748-757.

Gkolalas-Divanis, A., Verykios, V. 2008. Exact Knowledge Hiding Through Database Extension. *IEEE Transactions on Knowledge and Data Engineering* (21): 699-713.

Grossman, R., Kasif, S., Moore, R., Rocke, D., Ullman, J. 1998. Data Mining Research: Opportunities and Challenges. A Report of three NSF Workshops on Mining Large, Massive, and Distributed Data, http://pubs.rgrossman.com/dl/misc-001.pdf (last access May 31, 2010).

Han, J. and Kamber, M. 2005. *Data Mining: Concepts and Techniques*. Morgan Kaufman.

Han, J., Pei, J., Yin, Y. 2000. Mining Frequent Patterns without Candidate Generation. *In Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*: 1-12.

Hipp, J., Güntzer, U., Nakhaeizadeh, G. 2000. Algorithms for Association Rule Mining — A General Survey and Comparison. *SIGKDD Explorations Newsletter* (2): 58-64.

Huang, H., Wu, X., Relue, R. 2002. Association Analysis with One Scan of Databases. *In Proceedings of the 2002 IEEE International Conference on Data Mining*: 629-632.

Kantarcioglu, M. and Clifton, C. 2004. Privacy-Preserving Distributed Mining of Association Rules on Horizontally Partitioned Data. *IEEE Transsactions on Knowledge and Data Engineering* (16): 1026-1037.

Kantardzic, M. 2002. *Data Mining: Concepts, Models, Methods, and Algorithms*. IEEE Press.

Kotsiantis, S., Kanellopoulos, D. 2006. Association Rules Mining: A Recent Overview. *International Transactions on Computer Science and Engineering* (32): 71-82.

Lindell, Y., Pinkas, B. 2002. Privacy Preserving Data Mining. *Journal of Cryptology* (15): 177-206.

Liu, L., Özsu, M. 2009. *Encyclopedia of Database Systems*. Springer.

Lloyd, P. 1982. Least squares quantization in PCM. *IEEE Transactions on Information Theory* (28): 129-137.

Machanavajjhala, A., Kifer, D., Gehrke, J., and Venkitasubramaniam, M. 2007. *L*-diversity: Privacy beyond *k*-anonymity. *ACM Transactions on Knowledge Discovery from Data* (1): 1-52.

Mannila, H., Toivonen, H. 1997. Levelwise search and borders of theories in knowledge discovery. *Data Mining and Knowledge Discovery* (1): 241-258.

Mousakides, G., Verykios, V. 2008. A Max Min approach for hiding frequent itemsets. *Data and Knowledge Engineering* (65): 75-89.

Oliveira, S. and Zaiane, O. 2002. Privacy Preserving Frequent Itemset Mining. *In Proceedings of the 2002 IEEE International Conference on Privacy, Security and Data Mining*: 43-54.

Omari, A., Langer, R., Conrad, S. 2008. TARtool: A Temporal Dataset Generator for Market Basket Analysis. *In Proceedings of the 4th international Conference on Advanced Data Mining and Applications*: 400-410.

Pavon, J., Viana, S., Gomez, S. 2006. Matrix Apriori: Speeding up the Search for Frequent Patterns. *In Proceedings of the 24th IASTED International Conference on Databases and Applications*: 75-82.

Pinkas, B. 2002. Cryptographic techniques for privacy-preserving data mining. *ACM SIGKDD Explorations* (4): 12-19.

Quinlan, J. 1993. *C4.5: Programs for machine learning*. Morgan Kaufmann.

Savasere, A., Omiecinski E., Navathe, S. 1995. An Efficient Algorithm for Mining Association Rules in Large Databases. *In Proceedings of the 21st Iinternational Conference on Very Large Data Bases*: 432-444.

Saygin, Y., Verykios, V., Clifton, C. 2001. Using Unknowns to Prevent Discovery of Association Rules. *ACM SIGMOD Records* (30): 45-54.

Seifert, J. 2004. Data Mining: An Overview. CRS Report for Congress RL31798, http://www.fas.org/irp/crs/RL31798.pdf (last access June 5 2010).

Subramanian, R. 2008. *Computer Security, Privacy and Politics: Current Issues, Challenges and Solutions*. IGI Global.

Sun, X., Yu, P. 2005. A Border-Based Approach for Hiding Sensitive Frequent Itemsets. *In Proceedings of the 5th IEEE International Conference on Data Mining*: 426-433.

Sun, X., Yu, P. 2007. Hiding Sensitive Frequent Itemsets by a Border-Based Approach. *Journal of Computing Science and Engineering* (1): 74-94.

Sweeney, L. 1998. Achieving k-anonymity privacy protection using generalization and suppression. *International Journal on Uncertain, Fuzziness Knowledge-Based Systems* (10): 571-588.

Toivonen, H. 1996. Sampling Large Databases for Association Rules. *In Proceedings of the 22nd international Conference on Very Large Data Bases*: 134-145.

Vaidya, J. and Clifton, C. 2002. Privacy preserving association rule mining in vertically partitioned data. In *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining: 639-644.*

Vaidya, J., Clifton, C. 2004. Privacy Preserving Data Mining: Why, How, and When. *IEEE Security and Privacy* (2): 19-27.

Vaidya, J., Clifton, C., Zhu, Y. 2006. *Privacy Preserving Data Mining*. Springer.

Verykios, V., Bertino, E., Fovino, I., Provenza, L., Saygin, Y., Theodoridis, Y. 2004. State-of-the-art in Privacy Preservng Data Mining. *ACM SIGMOD Record* (33): 50-57.

Verykios, V., Elmagarmid, A., Bertino, E., Saygin, Y., Dasseni, E. 2004. Association Rule Hiding. *IEEE Transactions on Knowledge and Data Engineering* (16): 434-447.

Wang, S., Maskey, R., Jafari, A., Hong, T. 2008. Efficient sanitization of informative association rules. *Expert Systems with Applications* (35): 442-450.

Wu, X., et al. 2008. Top 10 Algorithms in Data Mining, *Knowledge Information Systems* (14): 1-37.

Yang, Q. and Wu, X. 2006. 10 Challenging Problems in Data Mining Research. *International Journal of Information Technology and Decision Making* (5): 597-604.

Yıldız, B., Ergenç, B. 2010. Comparison of Two Association Rule Mining Algorithms without Candidate Generation. *In Proceedings of the 10th IASTED International Conference on Artificial Intelligence and Applications*: 450-457.

Zaki, M., Parthasarathy, S., Ogihara, M. and Li, W. 1997. New algorithms for fast discovery of association rules. *In Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining*: 283-286.

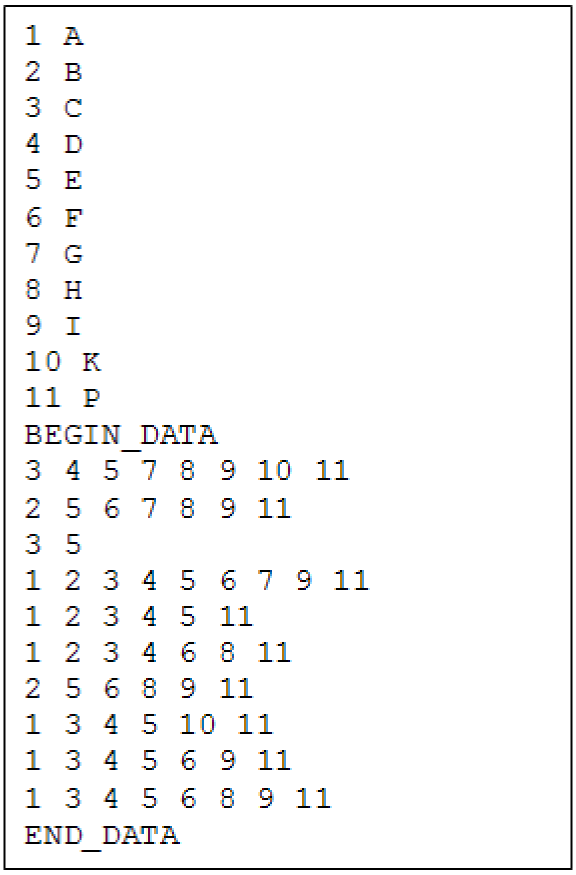
Zhang, N. and Zhao, W. 2007. Privacy-Preserving Data Mining Systems. *Computer* (40): 52-58.

Zheng, Z., Kohavi, R., Mason, L. 2001. Real World Performance of Association Rule Algorithms. *In Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*: 401-406.

APPENDIX A

INPUT FILE STRUCTURE

All of the implementations use “asc” file format. These files are created by ARtool software depending on parameters given. A simple file is displayed in Figure A.1. File includes items and numbers that represent these items until “BEGIN_DATA” phrase. Between “BEGIN_DATA” and “END_DATA” transactions are listed. Transactions are consists of representing numbers for related transaction.



```
1 A
2 B
3 C
4 D
5 E
6 F
7 G
8 H
9 I
10 K
11 P
BEGIN_DATA
3 4 5 7 8 9 10 11
2 5 6 7 8 9 11
3 5
1 2 3 4 5 6 7 9 11
1 2 3 4 5 11
1 2 3 4 6 8 11
2 5 6 8 9 11
1 3 4 5 10 11
1 3 4 5 6 9 11
1 3 4 5 6 8 9 11
END_DATA
```

Figure A.1. Structure of a simple input file

APPENDIX B

STEPS OF ALGORITHMS

In Figure B.1 below the creation of FP-tree is given step by step to make it clear. FP-tree is constructed in second scan of the database. Every transaction in the database is read in frequency order of the items excluding the ones below minimum support threshold.

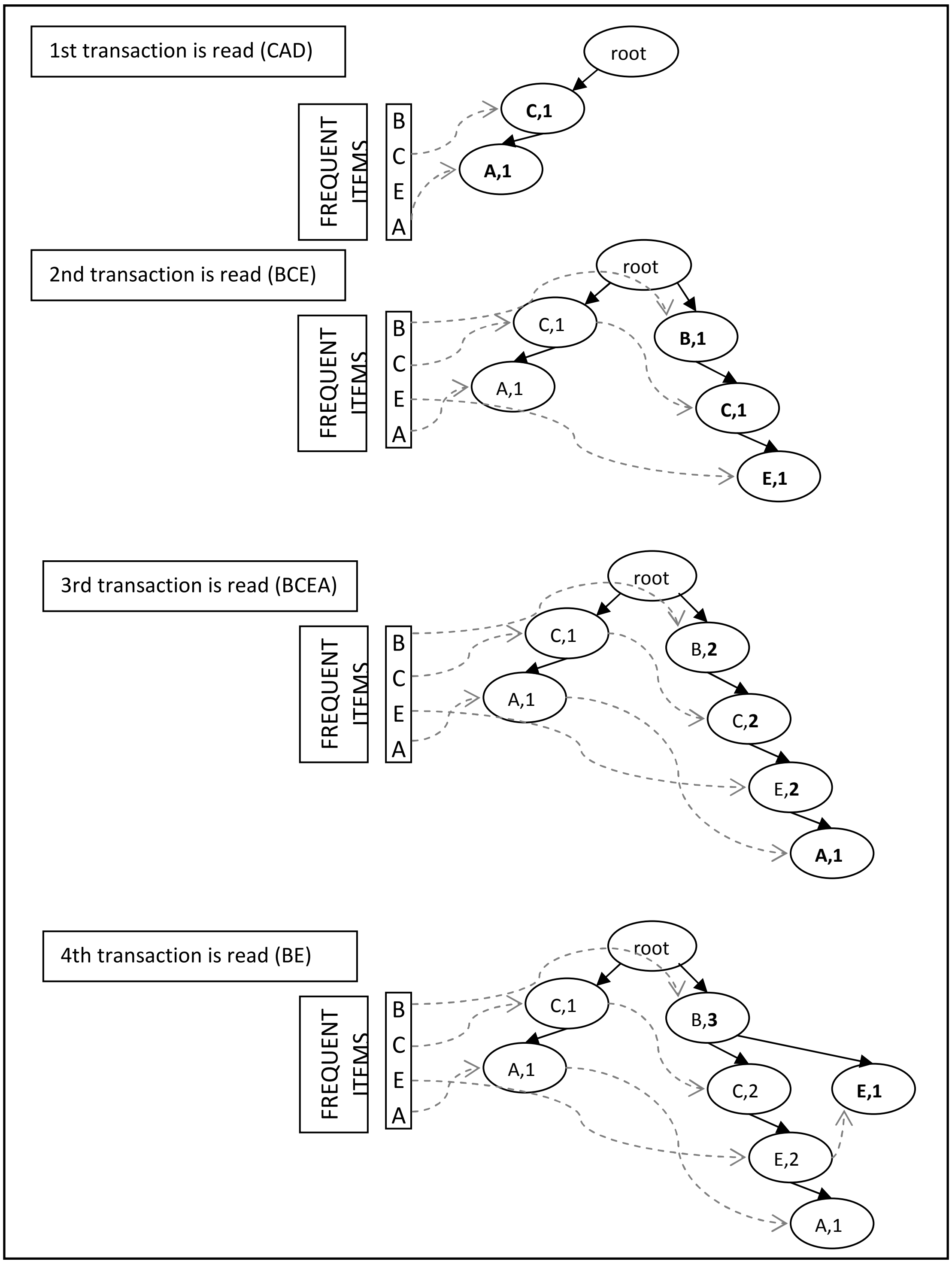


Figure B.1. FP-tree generation

From the FP-tree generated frequent itemsets can be found. In Figure B.2 for the item A conditional pattern bases extracted (branch from leaf to root), conditional FP-trees (only C for this example), frequent patterns found are given.

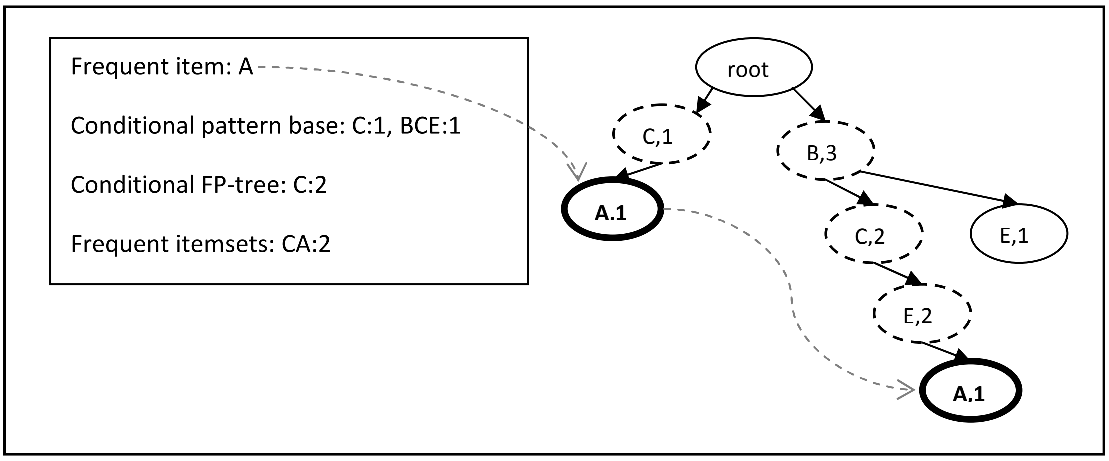
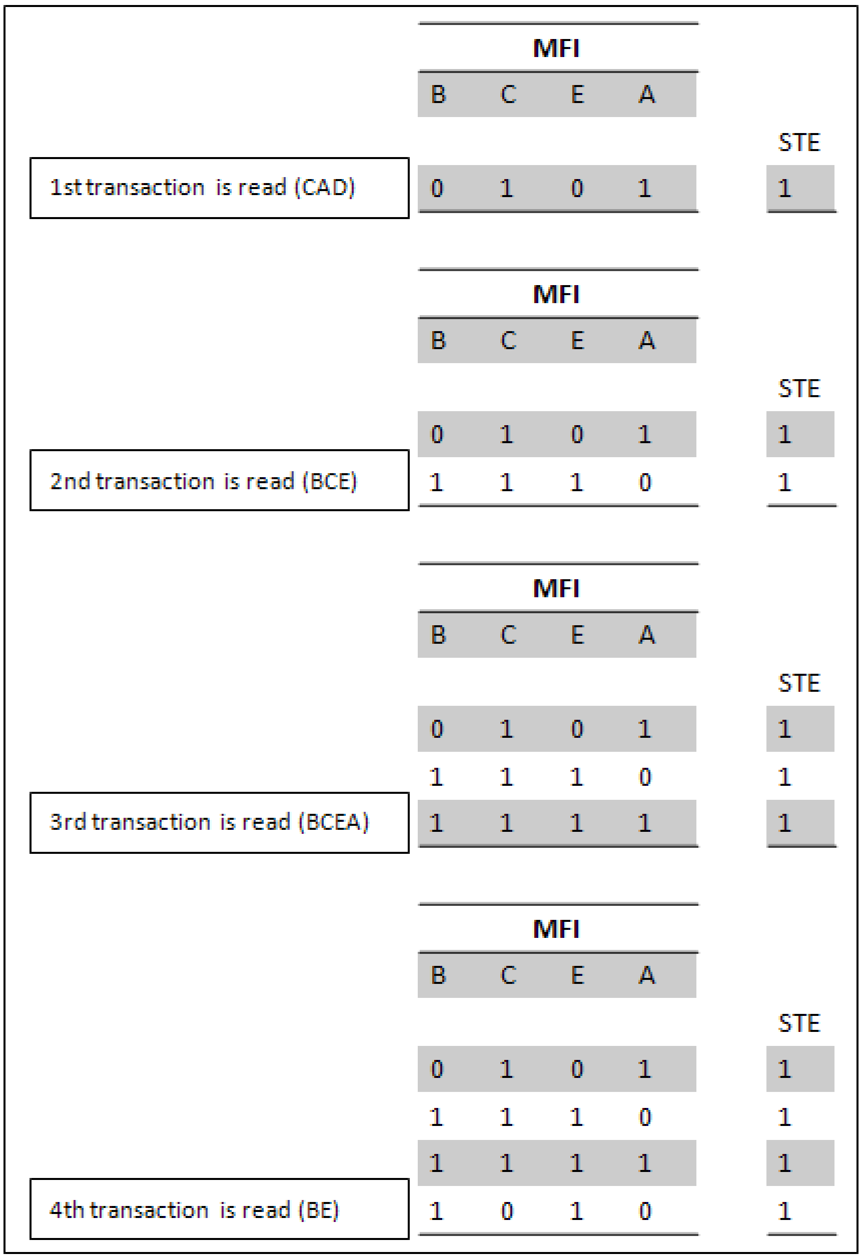


Figure B.2. Frequent pattern generation

In Figure B.3 below the creation of MFI and STE is given step by step to make it clear. Every transaction in the database is read in frequency order of the items excluding the ones below minimum support threshold (see Figure 3.4 for frequent items list).



1st transaction is read (CAD)

B	C	E	A	STE
0	1	0	1	1

2nd transaction is read (BCE)

B	C	E	A	STE
0	1	0	1	1
1	1	1	0	1

3rd transaction is read (BCEA)

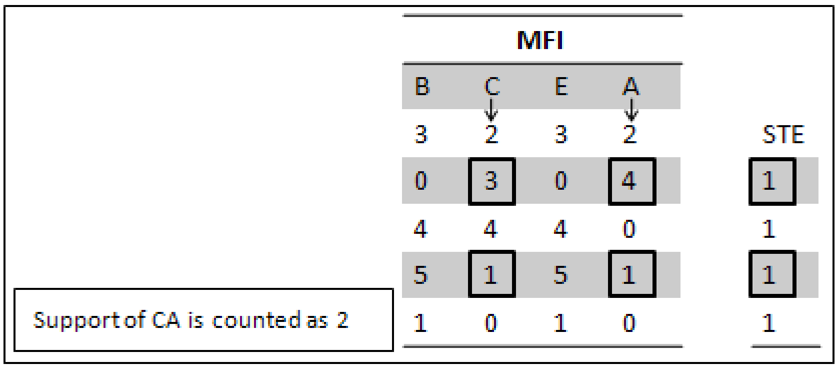
B	C	E	A	STE
0	1	0	1	1
1	1	1	0	1
1	1	1	1	1

4th transaction is read (BE)

B	C	E	A	STE
0	1	0	1	1
1	1	1	0	1
1	1	1	1	1
1	0	1	0	1

Figure B.3. MFI and STE generation

MFI and STE is constructed in second scan of the database and in Figure B.4 STE and MFI after modification is given which will speed up the frequent itemset finding. From the MFI and STE generated frequent itemsets can be found. For the candidate itemset CA counting support is given in Figure B.4 below.

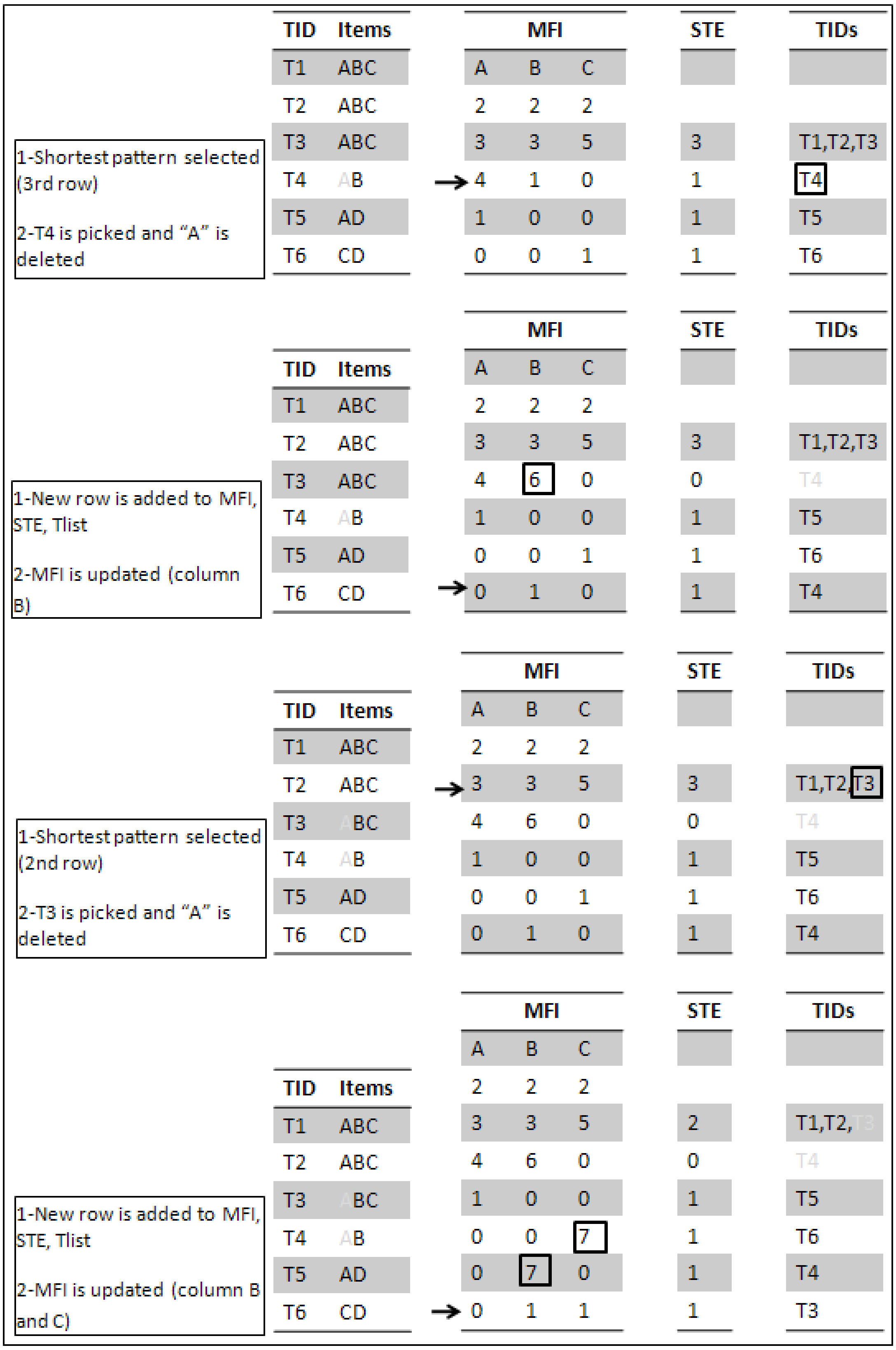


MFI				
B	C ↓	E	A ↓	
3	2	3	2	STE
0	$\boxed{3}$	0	$\boxed{4}$	$\boxed{1}$
4	4	4	0	1
5	$\boxed{1}$	5	$\boxed{1}$	$\boxed{1}$
1	0	1	0	1

Support of CA is counted as 2

Figure B.4. Itemset support counting example on MFI and STE

In Figure B.5 below the steps of frequent itemset hiding is given for the example in chapter 3. SpmaxFI algorithm is used in the example. Firstly shortest pattern in the MFI is selected and last transaction in the TList is picked. Later from the database most frequent item of sensitive itemset is deleted that is “A”. And MFI is updated. This is repeated until support of sensitive itemset is below minimum support.



1-Shortest pattern selected (3rd row)

2-T4 is picked and “A” is deleted

TID	Items
T1	ABC
T2	ABC
T3	ABC
T4	AB
T5	AD
T6	CD

	MFI			STE	TIDs
	A	B	C		
	2	2	2		
	3	3	5	3	T1,T2,T3
→	4	1	0	1	$\boxed{\text{T4}}$
	1	0	0	1	T5
	0	0	1	1	T6

1-New row is added to MFI, STE, Tlist

2-MFI is updated (column B)

TID	Items
T1	ABC
T2	ABC
T3	ABC
T4	AB
T5	AD
T6	CD

	MFI			STE	TIDs
	A	B	C		
	2	2	2		
	3	3	5	3	T1,T2,T3
	4	$\boxed{6}$	0	0	T4
	1	0	0	1	T5
	0	0	1	1	T6
→	0	1	0	1	T4

1-Shortest pattern selected (2nd row)

2-T3 is picked and “A” is deleted

TID	Items
T1	ABC
T2	ABC
T3	ABC
T4	AB
T5	AD
T6	CD

	MFI			STE	TIDs
	A	B	C		
	2	2	2		
→	3	3	5	3	T1,T2,$\boxed{\text{T3}}$
	4	6	0	0	T4
	1	0	0	1	T5
	0	0	1	1	T6
	0	1	0	1	T4

1-New row is added to MFI, STE, Tlist

2-MFI is updated (column B and C)

TID	Items
T1	ABC
T2	ABC
T3	ABC
T4	AB
T5	AD
T6	CD

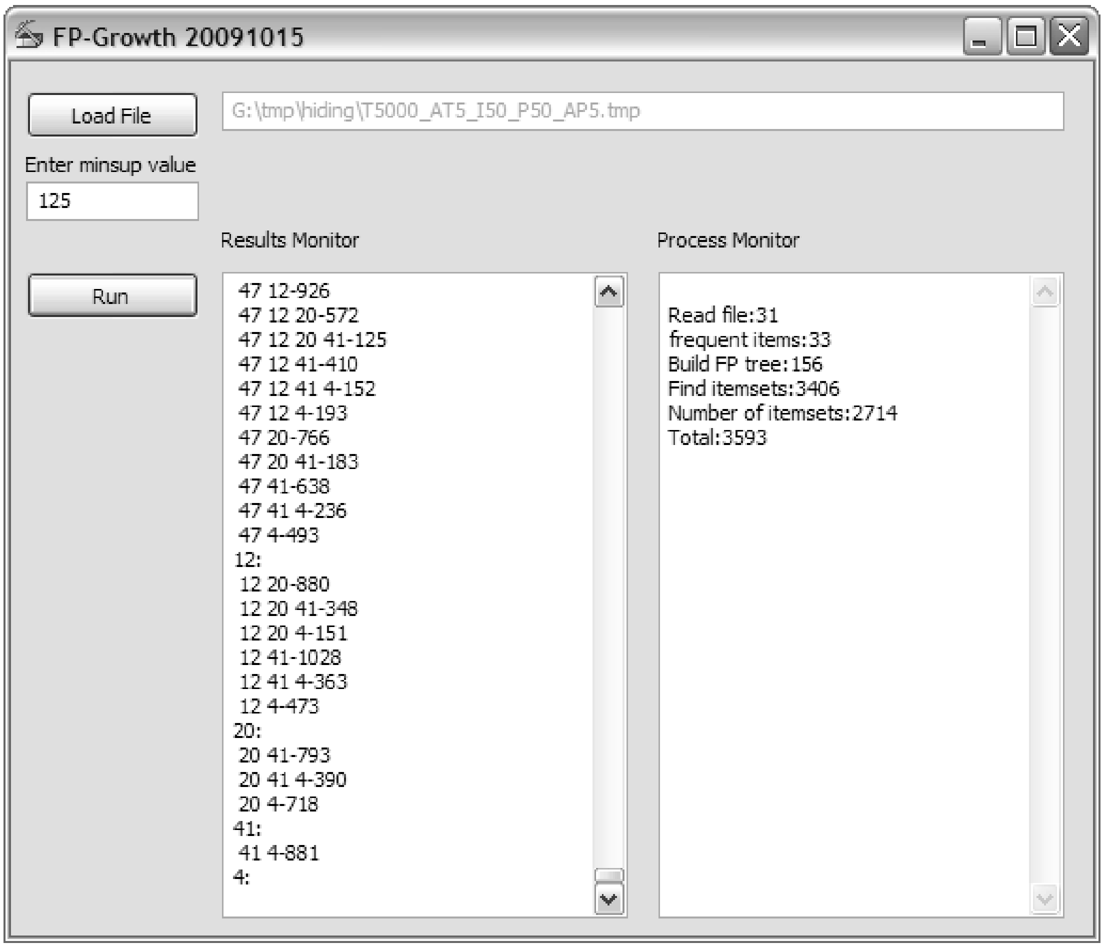
	MFI			STE	TIDs
	A	B	C		
	2	2	2		
	3	3	5	2	T1,T2,T3
	4	6	0	0	T4
	1	0	0	1	T5
	0	0	$\boxed{7}$	1	T6
	0	$\boxed{7}$	0	1	T4
→	0	1	1	1	T3

Figure B.5. Steps of itemset hiding by spmaxFI algorithm

APPENDIX C

GUI OF IMPLEMENTATIONS

In Figure C.1 below a simple execution of FP-Growth implementation is given. Firstly, “Load File” button is clicked and from open file dialog a file is selected. Here the file of 5k transactions used for frequent itemset hiding is selected. Given the minimum support count 125 (2.5%), “Run” button is clicked and program executed. In “Results Monitor” frequent itemsets and their supports are displayed where found itemsets are appended until no frequent itemsets are left. In “Process Monitor” information for performance evaluation is given: time to read file in ms, number of frequent items found, time to build FP-tree in ms, time to find frequent itemsets in ms, number of frequent itemsets and total time of execution.



G:\tmp\hiding\T5000_AT5_I50_P50_AP5.tmp

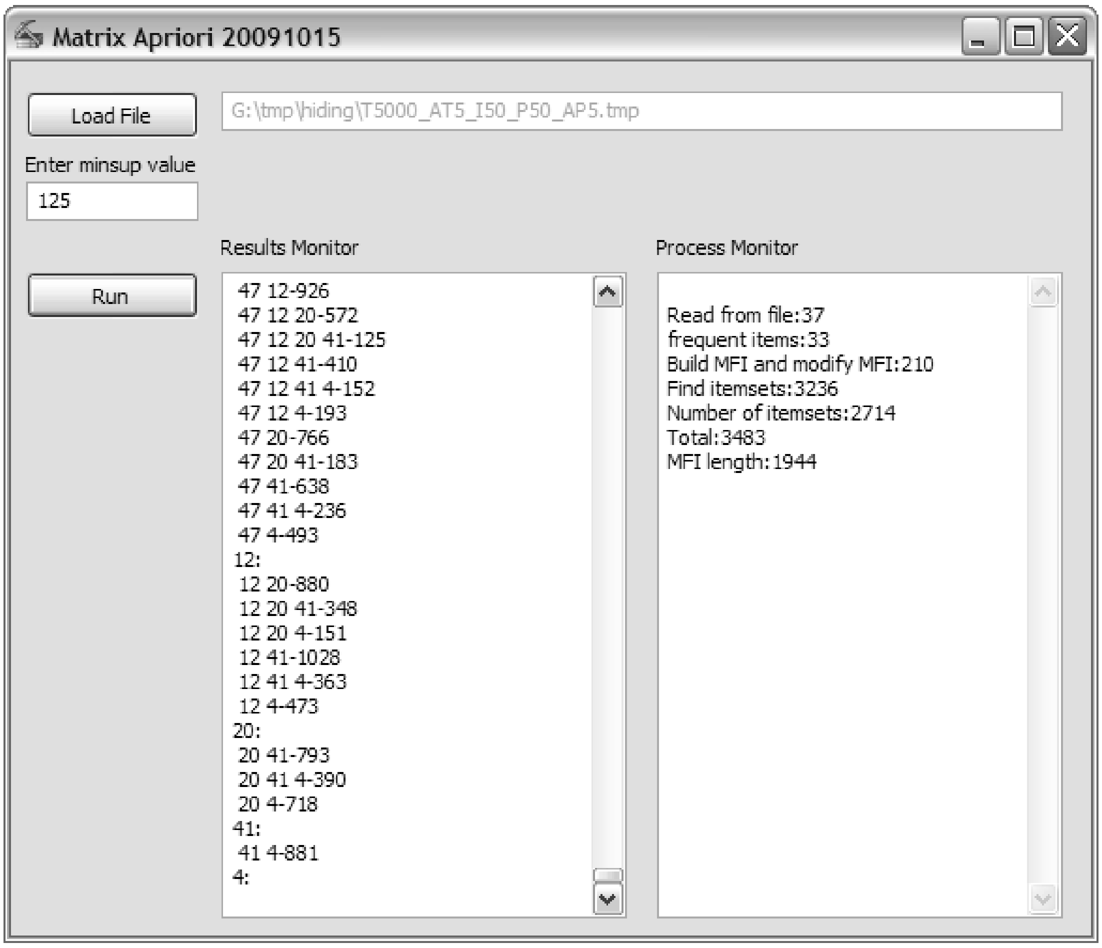
125

```
 47 12-926
 47 12 20-572
 47 12 20 41-125
 47 12 41-410
 47 12 41 4-152
 47 12 4-193
 47 20-766
 47 20 41-183
 47 41-638
 47 41 4-236
 47 4-493
12:
 12 20-880
 12 20 41-348
 12 20 4-151
 12 41-1028
 12 41 4-363
 12 4-473
20:
 20 41-793
 20 41 4-390
 20 4-718
41:
 41 4-881
4:
```

```
Read file:31
frequent items:33
Build FP tree:156
Find itemsets:3406
Number of itemsets:2714
Total:3593
```

Figure C.1. FP-Growth implementation

In Figure C.2 below a simple execution of Matrix Apriori implementation is given. Firstly, “Load File” button is clicked and from open file dialog a file is selected. Here again the file of 5k transactions used for frequent itemset hiding is selected. Given the minimum support count 125 (2.5%), “Run” button is clicked and program executed. In “Results Monitor” frequent itemsets and their supports are displayed where found itemsets are appended until no frequent itemsets are left. In “Process Monitor” information for performance evaluation is given: time to read file in ms, number of frequent items found, time to build MFI matrix and modify MFI matrix in ms, time to find frequent itemsets in ms, number of frequent itemsets and total time of execution. Row number of MFI matrix is also given as “MFI length”, however; it is not used for evaluation.



G:\tmp\hiding\T5000_AT5_I50_P50_AP5.tmp

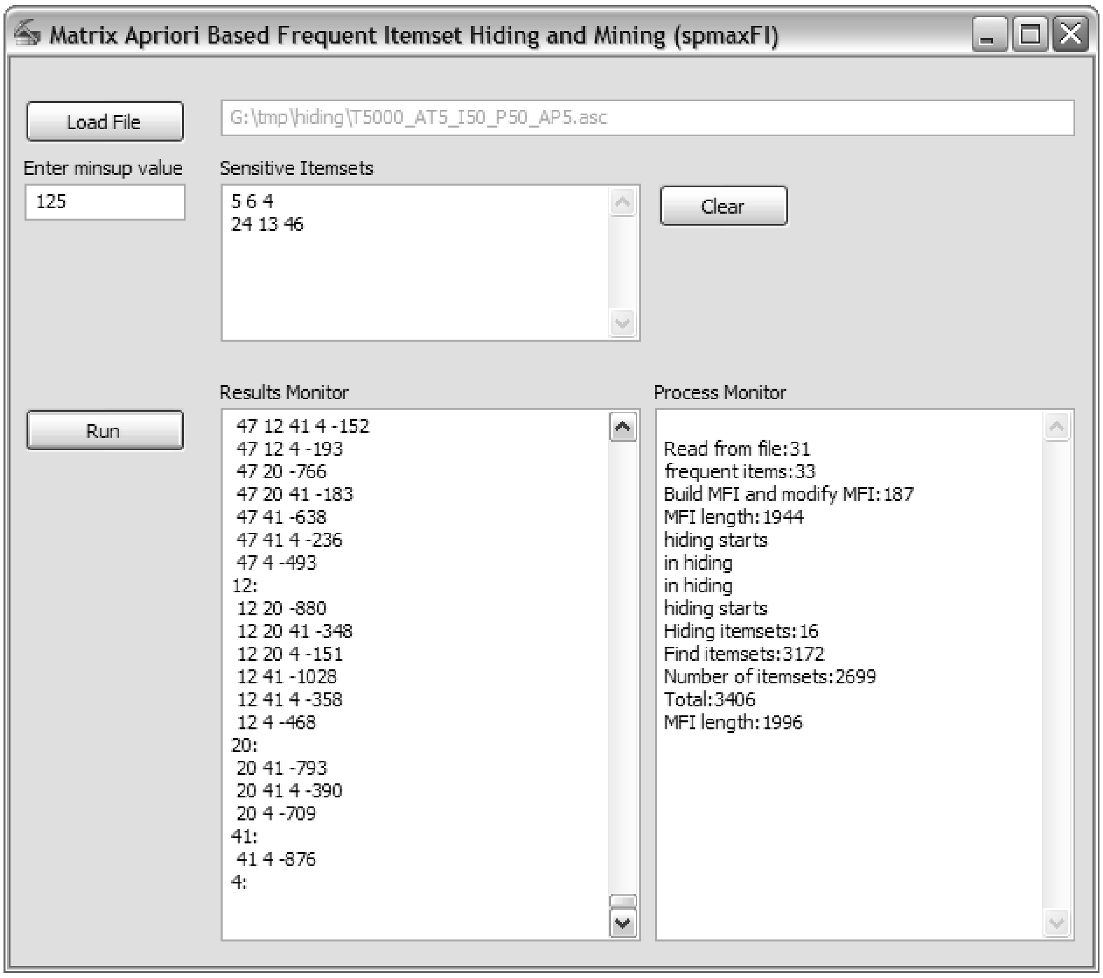
125

```
 47 12-926
 47 12 20-572
 47 12 20 41-125
 47 12 41-410
 47 12 41 4-152
 47 12 4-193
 47 20-766
 47 20 41-183
 47 41-638
 47 41 4-236
 47 4-493
12:
 12 20-880
 12 20 41-348
 12 20 4-151
 12 41-1028
 12 41 4-363
 12 4-473
20:
 20 41-793
 20 41 4-390
 20 4-718
41:
 41 4-881
4:
```

```
Read from file:37
frequent items:33
Build MFI and modify MFI:210
Find itemsets:3236
Number of itemsets:2714
Total:3483
MFI length:1944
```

Figure C.2. Matrix Apriori implementation

In Figure C.3 below a simple execution of proposed algorithm Matrix Apriori based frequent itemset hiding using spmaxFI strategy is displayed. Like frequent itemset mining implementations 5k database file is selected and minimum support count 125 (2.5%) is given. Sensitive itemsets can be entered in “Sensitive Itemsets” text box, each line representing an itemset. At this time we enter sensitive itemsets without the information if they are frequent or not. As mentioned in thesis this protects privacy against probable malicious user of itemset hider. Following, we click “Run” button and see two save file dialogs: first for saving frequent itemsets of sanitized database and second for saving sanitized database. The itemset file is used for finding supersets of sensitive itemsets to calculate number of lost itemsets after sanitization for comparison. “Results Monitor” gives found frequent itemsets after sanitization with support counts. This eliminates post-mining of new sanitized database. “Process Monitor” gives time to read file in ms, number of frequent items found, time to build MFI matrix and modify MFI matrix in ms, time to find frequent itemsets in ms, number of frequent itemsets and total time of execution. Row number of MFI matrix is also given as “MFI length” before and after sanitization to calculate distorted items. And in addition to Matrix Apriori implementation time for hiding process is given in ms.



```
G:\tmp\hiding\T5000_AT5_I50_P50_AP5.asc
```

Enter minsup value: 125

Sensitive Itemsets:

```
5 6 4
24 13 46
```

Results Monitor:

```
 47 12 41 4 -152
 47 12 4 -193
 47 20 -766
 47 20 41 -183
 47 41 -638
 47 41 4 -236
 47 4 -493
12:
 12 20 -880
 12 20 41 -348
 12 20 4 -151
 12 41 -1028
 12 41 4 -358
 12 4 -468
20:
 20 41 -793
 20 41 4 -390
 20 4 -709
41:
 41 4 -876
4:
```

Process Monitor:

```
Read from file:31
frequent items:33
Build MFI and modify MFI:187
MFI length:1944
hiding starts
in hiding
in hiding
hiding starts
Hiding itemsets:16
Find itemsets:3172
Number of itemsets:2699
Total:3406
MFI length:1996
```

Figure C.3. Matrix Apriori Based Frequent Itemset Hider for spmaxFI implementation