AN EXACT APPROACH WITH MINIMUM SIDE-EFFECTS FOR ASSOCIATION RULE HIDING

A Thesis Submitted to
the Graduate School of Engineering and Sciences of
Izmir Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of

MASTER OF SCIENCE

in Computer Engineering

by
Engin LELOĞLU

March 2014
İZMİR

We approve the thesis of **Engin LELOĞLU**

Examining Committee Members:

Asst. Prof. Dr. Tolga AYAV
Computer Engineering Department, Izmir Institute of Technology

---

Asst. Prof. Dr. Belgin ERGENÇ
Computer Engineering Department, Izmir Institute of Technology

---

Prof. Dr. Oğuz DİKENELLİ
Computer Engineering Department, Ege University

20 March 2014

Asst. Prof. Dr. Tolga AYAV
Supervisor, Computer Engineering Department
Izmir Institute of Technology

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

Prof. Dr. R. Tuğrul SENGER
Dean of the Graduate School of
Engineering and Sciences

ACKNOWLEDGMENTS

I would like to thank my thesis adviser Asst. Prof. Tolga Ayav, PhD., for his supervision, his respect to my individual views all along the work. I appreciate his sincerity, patience, maturity, and attentive style.

I also owe special thanks to Asst. Prof. Belgin Ergenç, PhD., for his valuable contribution during my undergraduate and graduate studies, for his guidance and support.

I would also like to thank the people of the Department of Computer Engineering of İzmir Institute of Technology whose friendship and support made my return to academic world after all those years, possible.

Finally, I thank my family and friends for their love and patience.

ABSTRACT

AN EXACT APPROACH WITH MINIMUM SIDE-EFFECTS FOR ASSOCIATION RULE HIDING

Concealing sensitive relationships before sharing a database is of utmost importance in many circumstances. This implies to hide the frequent itemsets corresponding to sensitive association rules by removing some items of the database. Research efforts generally aim at finding out more effective methods in terms of convenience, execution time and side-effect. This paper presents a practical approach for hiding sensitive patterns while allowing as much nonsensitive patterns as possible in the sanitized database. We model the itemset hiding problem as integer programming whereas the objective coefficients allow finding out a solution with minimum loss of nonsensitive itemsets. We evaluate our method using three real datasets from FIMI repository and compared the results with previous exact solution and the heuristic study whose procedures are imposed by new approach. The results show that information loss is dramatically minimized without sacrificing so many modifications on databases.

ÖZET

İLİŞKİSEL KURAL GİZLEME İÇİN EN AZ YAN ETKİLİ BİR TAM YAKLAŞIM

Mahrem bilgilerin dahil oldukları veri tabanlarının bir başka şahıs ya da kurumla paylaşılmadan önce gizlenmesi işlemi, birçok iş alanı için büyük önem arz eder. Bu işlem, ilgili veri tabanı üzerinden bir miktar öğenin silinmesi neticesinde mahrem ilişkisel kurallara karşılık gelen sık rastlanan öğe setlerinin gizlenmesiyle sonuçlanır. Araştırma eğilimi, genelde bu konu dahilindeki uygulanabilirlik, yürütüm süresi ve yan etki terimleri için daha efektif metotlar bulmayı amaçlar. Bu çalışma mahrem verilerin gizlendiği sırada mahrem olmayan diğer verilere mümkün olduğunca zarar verilmemesine dayalı pratik bir uygulama içerir. Biz bu çalışmada öğe setlerini gizleme problemini modellerken, ürettiğimiz objective katsayıların integer programlamada kullanılmasıyla, mahrem olmayan verilerin varlığını mümkün olduğunca korumaktayız. Oluşturduğumuz metodu, FIMI ambarından aldığımız üç gerçek veri seti üzerinde deneyip sonuçları, çalışırken referans aldığımız diğer exact ve heuristic çalışma ile karşılaştırmaktayız. Sonuçlar gösteriyor ki, veri tabanı üzerinde çok büyük değişiklere neden olmadan, veri kaybı önemli derecede minimize edilmiştir.

TABLE OF CONTENTS

LIST OF FIGURES

Figure **Page**

LIST OF TABLES

Table **Page**

CHAPTER 1

INTRODUCTION

Progresses in the technology give an opportunity to establish transactional databases that can reserve large volumes of data. Analyzing data and extracting meaningful information from these huge piles of data come up as a result of these advances. Data mining field has efficient techniques for this knowledge discovery process. However, improper use of these techniques caused a rise of privacy concerns. Unauthorized access to not only sensitive personal information that is stored or inferred from the data, but also commercial information that provides remarkable benefit over rivals induces privacy issues. That is why comprehensive sanitization on databases is required when the information or data from these databases is shared or published.

Sharing databases allows researchers and policy-makers to examine the data and gain significant information benefiting the society as a whole, such as the strength of a medicine or treatment, social-economic inferences that can be the guide on the road to efficient public policies, and the factors that cause vital diseases. In other words, publishing databases eventuates in utility gain for the society as a whole [1]. However, due to privacy concerns, a privacy preserving method is needed to be applied on the databases. These methods make data imprecise and/or distorted so that no sensitive knowledge is disclosed. But, this distortion causes unwanted information loss and losses in potential utility gain.

Frequent itemset hiding is one of the important and widely used methods of privacy preserving data mining field. There are several frequent itemset hiding algorithms whose methodology can be classified such as heuristic [2], border-based [3 and 4] and exact [5, 6, 7 and 8]. They aim to impose small deviation in the original database to expose no sensitive itemsets. This deviation is tried to be minimized by various techniques with different quality metrics as a common feature of these algorithms. One determines the relative frequency of remaining itemsets [3] as a quality parameter while another approach uses the term of accuracy [8] that shows the impact of sanitization on transactions of the database. In addition to this, the information loss which is to conceal nonsensitive itemsets on the original database while hiding sensitive

knowledge is another critical point of the hiding process [3]. Studies generally concentrate on achieving the result database which has no sensitive knowledge with small deviation and minimum information loss.

Exact approaches produce more accurate solution than other types of approaches in frequent itemset hiding. However, they are impractical when the number of itemsets and length of itemsets increase. In addition to this, they mostly focus on minimizing deviation in terms of accuracy or distance. To our knowledge there is no practical solution providing frequent itemset hiding with the objective of minimum information loss and accuracy. In this paper, we propose an exact approach for frequent itemset hiding where all sensitive patterns are concealed. Our approach is based on the combination of integer programming and heuristic sanitization. While it prevents revealing sensitive information on published database, minimum information loss and maximum accuracy are also provided.

Our approach proposes the use of coefficients in the objective function of integer programming to minimize information loss. These coefficients reminding the approaches used by utility-based mining algorithms [9] are pre-computed such that they give a measure of information loss. Integer programming allows finding the optimum solution deciding about the transactions to be sanitized. Then, heuristic sanitization algorithm is executed to remove the sensitive itemsets. The experiments with real datasets demonstrate the efficacy of our approach and give useful insight into the efforts of minimizing nonsensitive information loss.

The following sections are organized as follows: Chapter 1 includes an introduction. The way of organizing our study and main concepts of this study are described briefly. Chapter 2 gives the background of the problem with the problem formulation and the literature of review. Section 2.1 includes the problem formulation with terms, concepts and tables that have information about the example database is used in Chapter 3. Section 2.2 mentions basis of data mining, association rule mining and privacy preserving data mining. In addition, the section contains an overview of the leading studies about association rule hiding which is one of the important field for privacy preserving data mining. Chapter 3 presents our approach in detail. Chapter 4 shows the results and evaluations of all experiments to prove the effectiveness of the technique the thesis presents. Finally, we conclude in Chapter 5.

CHAPTER 2

BACKGROUND

2.1. Problem Formulation

Let F be a set of items. An itemset is a subset of F and any transaction defined over F is tuple $< k, F_k >$, where k is the transaction id and F_k is the itemset. A transaction $< k, F_k >$ is said to contain an itemset X iff $F_k \supseteq X$. A database D is a set of transactions. Given a database D, the support of an itemset F_k in the database D is denoted as *the support* $\sigma(F_k, D)$. $\sigma(F_k, D)$ can be represented simply as σ_k for notational convenience. For a given threshold σ_{min}, F_k is said to be *frequent* if $\sigma(F_k, D) \geq \sigma_{min}$. The set of frequent itemsets $F\ (\sigma_{min})$ at minimum support level σ_{min} is the set of all itemsets with a minimum support of σ_{min}.

$F^R(\sigma_{min}) \subseteq F(\sigma_{min})$ is a group of restrictive patterns that the owner of the data would like to conceal while publishing. A transaction that supports any of these patterns is said to be *sanitized* if any alteration is made on it in such a way that it no longer supports any itemset in $F^R(\sigma_{min})$. This sanitization implies reducing the support for every $j \in F^R(\sigma_{min})$ below σ_{min} and concealing itemset j.

In the process of transforming a database D to a sanitized D', we have the following considerations:

1) Suppose that $F'(\sigma_{min})$ be the set of frequent itemsets in the sanitized D'. Any $j \in F^R(\sigma_{min})$ in D should not be in $F'(\sigma_{min})$. In other words, it is aimed that no sensitive knowledge is involved in the sanitized database.

2) The accuracy, which is the ratio of the number of transactions that are not sanitized and the total number of transactions in the database D, should be maximized by keeping the number of sanitized transactions at minimum.

3) Suppose that $F^n(\sigma_{min})$ be the set of non-sensitive frequent itemsets determined by $F(\sigma_{min})/F^R(\sigma_{min})$ in database D. $|F^n(\sigma_{min}) - F'(\sigma_{min})|$ should be minimized to avoid overconcealing nonsensitive frequent itemsets and keeping the information loss at minimum.

Table 2. 1. Example Database *D*

Id	**Items**
T_1	1 2 3 7 8 10
T_2	3 9 10
T_3	4 5 6
T_4	1 2 3 6 7 8 9
T_5	1 2 3 6 7
T_6	10
T_7	4
T_8	3 6 7 8 9
T_9	3 8 9
T_{10}	5 6 7

Table 2. 2. Frequent (Nonsingleton) Itemsets for *D* at $\sigma_{min} = 2$

Itemsets	σ_j	Itemsets	σ_j	Itemsets	σ_j	Itemsets	σ_j
5, 6	2	9, 6	2	2, 8, 7	2	1, 2, 8, 7	2
1, 2	3	9, 7	2	2, 8, 3	2	1, 2, 8, 3	2
1, 8	2	9, 3	4	2, 6, 7	2	1, 2, 6, 7	2
1, 6	2	r_3 →**6, 7**	**4**	2, 6, 3	2	1, 2, 6, 3	2
1, 7	3	6, 3	3	2, 7, 3	3	1, 2, 7, 3	3
1, 3	3	7, 3	4	8, 9, 6	2	1, 8, 7, 3	2
2, 8	2	1, 2, 8	2	8, 9, 7	2	1, 6, 7, 3	2
2, 6	2	1, 2, 6	2	8, 9, 3	3	2, 8, 7, 3	2
2, 7	3	1, 2, 7	3	8, 6, 7	2	2, 6, 7, 3	2
2, 3	3	r_4 →**1, 2, 3**	**3**	8, 6, 3	2	8, 9, 6, 7	2
10, 3	2	1, 8, 7	2	8, 7, 3	3	8, 9, 6, 3	2
r_1 →**8, 9**	**3**	1, 8, 3	2	9, 6, 7	2	8, 9, 7, 3	2
8, 6	2	1, 6, 7	2	9, 6, 3	2	8, 6, 7, 3	2
8, 7	3	1, 6, 3	2	9, 7, 3	2	9, 6, 7, 3	2
r_2 →**8, 3**	**4**	1, 7, 3	3	6, 7, 3	3	1, 2, 8, 7, 3	2
						1, 2, 6, 7, 3	2
						8, 9, 6, 7, 3	2

For example, we assume that the sensitive patterns are {8, 9}, {8, 3}, {6, 7}, {1, 2, 3} that are bold and represented such as r_1, r_2, r_3 and r_4. Although, it is possible to define different support thresholds for each sensitive pattern, we assume that the support threshold $\sigma_{min} = 2$ for all patterns, which is practical and common in many circumstances. At the end of the process, we expect that 28 nonsensitive frequent itemsets that are supersets of the sensitive ones would also get concealed as the process of hiding the sensitive itemsets. The question is how to transform *D* into the sanitized database D' in an effective way such that aforementioned considerations 1, 2 and 3 are maintained.

Depending on the consideration 1, support values of our sensitive patterns in the database *D* should be dropped below 2, that is minimum support value. For example, the support value of r_2 is 4 and to satisfy the consideration 1, transactions that include r_2 should be found and at least one of two items in r_2 should be deleted from as many transactions as needed. The proper selection of transactions to be sanitized and the items to be removed is of paramount importance, since the number of sanitized transactions and/or the number of items to be removed should be kept minimum. Moreover, nonsensitive itemsets that contain one of items, 8 or 3, are in danger of being concealed while the support value of r_2 is decreased. According to consideration 3, the number of nonsensitive itemsets that are concealed should be minimum.

2.2. Literature Review

2.2.1. Data Mining

It is no surprise that data mining, as a truly interdisciplinary subject, can be defined in many different ways. Even the term data mining does not really present all the major components in the picture. To refer to the mining of gold fromrocks or sand, we say gold mining instead of rock or sand mining. Analogously, data mining should have been more appropriately named "knowledge mining from data," which is unfortunately somewhat long. However, the shorter term, knowledge mining may not reflect the emphasis on mining from large amounts of data. Nevertheless, mining is a vivid term characterizing the process that finds a small set of precious nuggets from a great deal of raw material (Figure 2. 1). Thus, such a misnomer carrying both "data" and "mining" became a popular choice. In addition, many other terms have a similar meaning to data mining—for example, knowledge mining from data, knowledge extraction, data/pattern analysis, data archaeology, and data dredging.

Many people treat data mining as a synonym for another popularly used term, knowledge discovery from data, or KDD, while others view data mining as merely an essential step in the process of knowledge discovery. The knowledge discovery process is shown in Figure 2. 2 as an iterative sequence of the following steps:

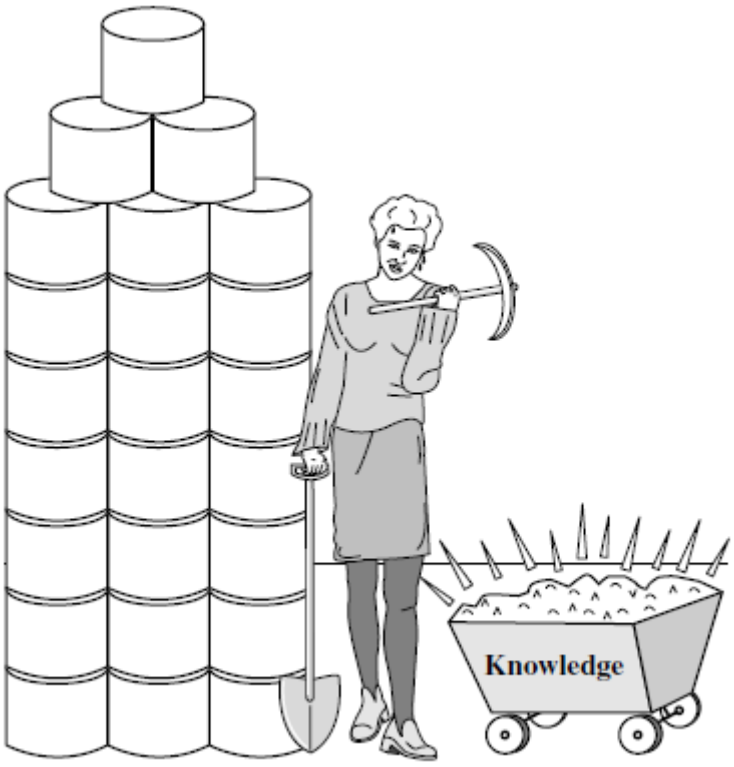

Figure 2. 1. Data mining—searching for knowledge (interesting patterns) in data [10].

1. Data cleaning (to remove noise and inconsistent data)

2. Data integration (where multiple data sources may be combined)

3. Data selection (where data relevant to the analysis task are retrieved from the database)

4. Data transformation (where data are transformed and consolidated into forms appropriate for mining by performing summary or aggregation operations)

5. Data mining (an essential process where intelligent methods are applied to extract data patterns)

6. Pattern evaluation (to identify the truly interesting patterns representing knowledge based on interestingness measures)

7. Knowledge presentation (where visualization and knowledge representation techniques are used to present mined knowledge to users)

Steps 1 through 4 are different forms of data preprocessing, where data are prepared for mining. The data mining step may interact with the user or a knowledge base. The interesting patterns are presented to the user and may be stored as new knowledge in the knowledge base.

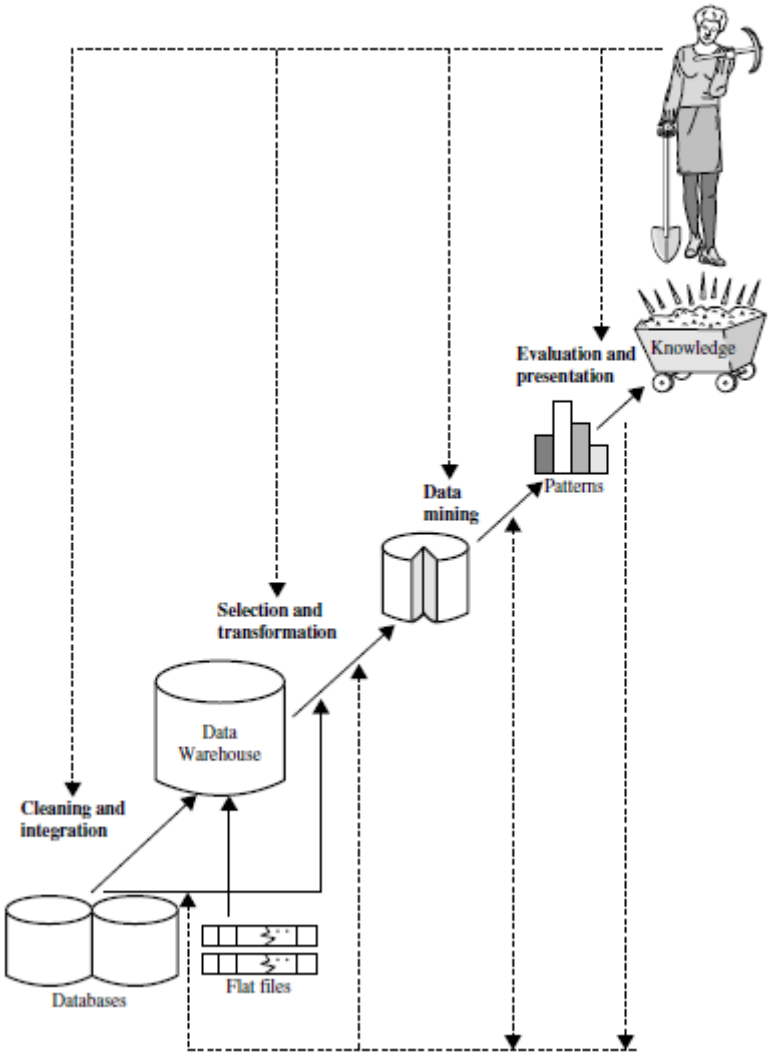

Figure 2. 2. Data mining as a step in the process of knowledge discovery [10]

The preceding view shows data mining as one step in the knowledge discovery process, albeit an essential one because it uncovers hidden patterns for evaluation. However, in industry, in media, and in the research milieu, the term data mining is often used to refer to the entire knowledge discovery process (perhaps because the term is shorter than knowledge discovery from data). Therefore, we adopt a broad view of data mining functionality: Data mining is the process of discovering interesting patterns and knowledge from large amounts of data. The data sources can include databases, data warehouses, the Web, other information repositories, or data that are streamed into the system dynamically.

2.2.1.1. What Kinds of Data Can Be Mined?

As a general technology, data mining can be applied to any kind of data as long as the data are meaningful for a target application. The most basic forms of data for mining applications are database data, data warehouse data, and transactional data (Section 2.2.1.1.1). Data mining can also be applied to other forms of data (e.g., data streams, ordered/sequence data, graph or networked data, spatial data, text data, multimedia data, and the WWW). Transactional data is focused on this thesis. Experiments are made on this type of datasets.

2.2.1.1.1. Transactional Data

In general, each record in a transactional database captures a transaction, such as a customer's purchase, a flight booking, or a user's clicks on a web page. A transaction typically includes a unique transaction identity number (trans ID) and a list of the items making up the transaction, such as the items purchased in the transaction. A transactional database may have additional tables, which contain other information related to the transactions.

trans_ID	*list_of_item_IDs*
T100	I1, I3, I8, I16
T200	I2, I8
...	...

Figure 2. 3. Fragment of an example transactional database for sales.

Example 2.1. Transactions can be stored in a table, with one record per transaction. A fragment of an example transactional database is shown in Figure 2. 3. From the relational database point of view, the table in the figure is a nested relation because the attribute list of item IDs contains a set of items. Because most relational database systems do not support nested relational structures, the transactional database is usually either stored in a flat file in a format similar to the table in Figure 2. 3.

As an analyst of the example transactional database, you may ask,“Which items sold well together?” This kind of market basket data analysis would enable you to bundle groups of items together as a strategy for boosting sales. For example, given the knowledge that printers are commonly purchased together with computers, you could offer certain printers at a steep discount (or even for free) to customers buying selected computers, in the hopes of selling more computers (which are often more expensive than printers). A traditional database system is not able to perform market basket data analysis. Fortunately, data mining on transactional data can do so by mining frequent itemsets, that is, sets of items that are frequently sold together.

2.2.1.2. Mining Frequent Patterns and Associations

Frequent patterns, as the name suggests, are patterns that occur frequently in data. There are many kinds of frequent patterns, including frequent itemsets, frequent subsequences (also known as sequential patterns), and frequent substructures. A frequent itemset typically refers to a set of items that often appear together in a transactional data set—for example, milk and bread, which are frequently bought together in grocery stores by many customers. A frequently occurring subsequence, such as the pattern that customers, tend to purchase first a laptop, followed by a digital camera, and then a memory card, is a (frequent) sequential pattern. A substructure can refer to different structural forms (e.g., graphs, trees, or lattices) that may be combined with itemsets or subsequences. If a substructure occurs frequently, it is called a (frequent) structured pattern. Mining frequent patterns leads to the discovery of interesting associations within data.

Example 2.2 - Association analysis. Suppose that, as a marketing manager, you want to know which items are frequently purchased together (i.e., within the same transaction). An example of such a rule, mined from the example transactional database, is buys (X, “computer”) → buys (X, “software”) [support = 1%], where X is a variable representing a customer. A 1% support means that 1% of all the transactions under analysis show that computer and software are purchased together. Typically, association rules are discarded as uninteresting if they do not satisfy both a minimum support threshold.

To gain association rules of transactional database systems, Apriori, which is a kind of frequent itemsets finding algorithm, can be used to mine frequent itemsets.

2.2.1.2.1. Apriori Algorithm

Apriori is a seminal algorithm proposed by R. Agrawal and R. Srikant [11] in 1994 for mining frequent itemsets for Boolean association rules. Apriori employs an iterative approach known as a level-wise search, where k-itemsets are used to explore (k+1)-itemsets. First, the set of frequent 1-itemsets is found by scanning the database to accumulate the count for each item, and collecting those items that satisfy minimum support. The resulting set is denoted by L_1. Next, L_1 is used to find L_2, the set of frequent 2-itemsets, which is used to find L_3, and so on, until no more frequent k-itemsets can be found. The finding of each L_k requires one full scan of the database.

To improve the efficiency of the level-wise generation of frequent itemsets, an important property called the Apriori property is used to reduce the search space.

Apriori property: *All nonempty subsets of a frequent itemset must also be frequent* [10].

The Apriori property is based on the following observation. By definition, if an itemset *I* does not satisfy the minimum support threshold, *min_sup*, then *I* is not frequent, that is, $P(I) < min\_sup$. If an item *A* is added to the itemset *I*, then the resulting itemset (i.e., $I \cup A$) cannot occur more frequently than *I*. Therefore, $I \cup A$ is not frequent either, that is, $P(I \cup A) < min\_sup$.

This property belongs to a special category of properties called antimonotonicity in the sense that if a set cannot pass a test, all of its supersets will fail the same test as well. It is called antimonotonicity because the property is monotonic in the context of failing a test.

Let's look at a concrete example, based on the *Apriori Training Database*, *D*, of Table 2. 3. There are nine transactions in this database, that is, $|D| = 9$. We use Figure 2. 3 to illustrate the Apriori algorithm for finding frequent itemsets in D.

1. In the first iteration of the algorithm, each item is a member of the set of candidate 1-itemsets, C_1. The algorithm simply scans all of the transactions to count the number of occurrences of each item. candidate 1-itemsets, C_1. The algorithm simply scans all of the transactions to count the number of occurrences of each item.

Table 2. 3. Apriori Training Database

Id	Items
$T100$	I1, I2, I5
$T200$	I2, I4
$T300$	I2, I3
$T400$	I1, I2, I4
$T500$	I1, I3
$T600$	I2, I3
$T700$	I1, I3
$T800$	I1, I2, I3, I5
$T900$	I1, I2, I3

2. Suppose that the minimum support count required is 2, that is, *min_sup* = 2. (Here, we are referring to absolute support because we are using a support count. The corresponding relative support is 2/9 = 22%.) The set of frequent 1-itemsets, L_1, can then be determined. It consists of the candidate 1-itemsets satisfying minimum support.

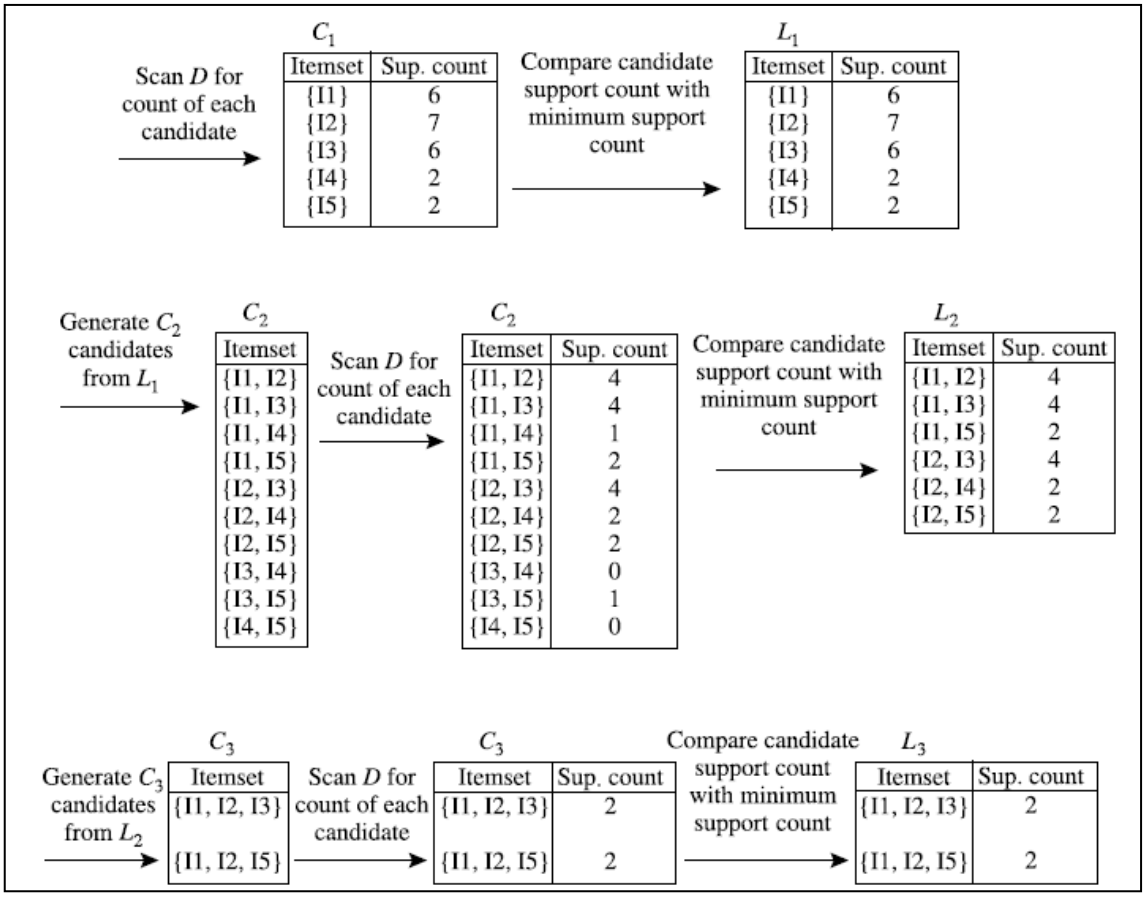

Figure 2. 4. Generation of the candidate itemsets and frequent itemsets [10].

In our example, all of the candidates in C_1 satisfy minimum support.

3. To discover the set of frequent 2-itemsets, L_2, the algorithm uses the join $L_1 \bowtie L_1$ to generate a candidate set of 2-itemsets, C_2. C_2 consists of $\binom{|L_1|}{2}$ 2-itemsets. Note that no candidates are removed from C_2 during the prune step because each subset of the candidates is also frequent.

4. Next, the transactions in D are scanned and the support count of each candidate itemset in C_2 is accumulated, as shown in the middle table of the second row in Figure 2. 4.

5. The set of frequent 2-itemsets, L_2, is then determined, consisting of those candidate 2-itemsets in C_2 having minimum support.

6. The generation of the set of the candidate 3-itemsets, C_3, is detailed in Figure 2.5. From the join step, we first get $C_3 = L_2 \bowtie L_2 = \{\{I1, I2, I3\}, \{I1, I2, I5\}, \{I1, I3, I5\}, \{I2, I3, I4\}, \{I2, I3, I5\}, \{I2, I4, I5\}\}$. Based on the Apriori property that all subsets of a frequent itemset must also be frequent, we can determine that the four latter candidates cannot possibly be frequent. We therefore remove them from C_3, thereby saving the effort of unnecessarily obtaining their counts during the subsequent scan of D to determine L_3.

(a) Join: $C_3 = L_2 \bowtie L_2 = \{\{I1, I2\}, \{I1, I3\}, \{I1, I5\}, \{I2, I3\}, \{I2, I4\}, \{I2, I5\}\}$
$\bowtie \{\{I1, I2\}, \{I1, I3\}, \{I1, I5\}, \{I2, I3\}, \{I2, I4\}, \{I2, I5\}\}$
$= \{\{I1, I2, I3\}, \{I1, I2, I5\}, \{I1, I3, I5\}, \{I2, I3, I4\}, \{I2, I3, I5\}, \{I2, I4, I5\}\}$.

(b) Prune using the Apriori property: All nonempty subsets of a frequent itemset must also be frequent. Do any of the candidates have a subset that is not frequent?

- The 2-item subsets of {I1, I2, I3} are {I1, I2}, {I1, I3}, and {I2, I3}. All 2-item subsets of {I1, I2, I3} are members of L_2. Therefore, keep {I1, I2, I3} in C_3.
- The 2-item subsets of {I1, I2, I5} are {I1, I2}, {I1, I5}, and {I2, I5}. All 2-item subsets of {I1, I2, I5} are members of L_2. Therefore, keep {I1, I2, I5} in C_3.
- The 2-item subsets of {I1, I3, I5} are {I1, I3}, {I1, I5}, and {I3, I5}. {I3, I5} is not a member of L_2, and so it is not frequent. Therefore, remove {I1, I3, I5} from C_3.
- The 2-item subsets of {I2, I3, I4} are {I2, I3}, {I2, I4}, and {I3, I4}. {I3, I4} is not a member of L_2, and so it is not frequent. Therefore, remove {I2, I3, I4} from C_3.
- The 2-item subsets of {I2, I3, I5} are {I2, I3}, {I2, I5}, and {I3, I5}. {I3, I5} is not a member of L_2, and so it is not frequent. Therefore, remove {I2, I3, I5} from C_3.
- The 2-item subsets of {I2, I4, I5} are {I2, I4}, {I2, I5}, and {I4, I5}. {I4, I5} is not a member of L_2, and so it is not frequent. Therefore, remove {I2, I4, I5} from C_3.

(c) Therefore, $C_3 = \{\{I1, I2, I3\}, \{I1, I2, I5\}\}$ after pruning.

Figure 2.5. Generation and pruning of candidate 3-itemsets, C3, from L2 [10].

Note that when given a candidate k-itemset, we only need to check if its (k-1)-subsets are frequent since the Apriori algorithm uses a level-wise search strategy. The resulting pruned version of C_3 is shown in the first table of the bottom row of Figure 2.4.

7. The transactions in D are scanned to determine L_3, consisting of those candidate 3-itemsets in C_3 having minimum support (Figure 2.4).

8. The algorithm uses $L_3 \bowtie L_3$ to generate a candidate set of 4-itemsets, C_4. Although the join results in {{I1, I2, I3, I5}}, itemset {I1, I2, I3, I5} is pruned because its subset {I2, I3, I5} is not frequent. Thus, $C_4 = \emptyset$, and the algorithm terminates, having found all of the frequent itemsets.

2.2.2. Data Utility

Publishing database enables researchers and policy-makers to analyze the data and learn important information benefiting the society as a whole, such as the factors causing certain diseases, effectiveness of a medicine or treatment, and social-economic patterns that can guide the formulation of effective public policies. In other words, publishing database results in utility gain for the society as a whole. However, as the database contains specific information about individuals, a household, or an organization, publishing database could also results in privacy loss for them. Hence before the database can be made public, one must ensure that the privacy loss is limited to an acceptable level. This is typically done via anonymization, which modifies the database to improve the privacy. Because anonymization makes data in the database modified, it also causes losses in potential utility gain, when compared with the case of disclosing the original database.

The challenges include [12]:

Utility measure: One key issue in the utility-based privacy preservation is how to model the data utility in different applications. A good utility measure should capture the intrinsic factors that affect the quality of data for the specific application.

Balance between utility and privacy: In some situation, preserving utility and privacy are not conflicting. But more often than not, hiding the privacy information may have to sacrifice some utility. How do we trade off between the two goals?

Efficiency and scalability: The traditional privacy preservation is already computational challenging. For example, even simple restriction of optimized k-

anonymity is NP-hard [13]. How do we develop efficient algorithms if utility is involved? Moreover, real databases often contains millions of high dimensional tuples, highly scalable algorithms are needed.

Ability to deal with different types of attributes: Real life data often involve different types of attributes, such as numerical, categorical, binary or mixtures of these data types. The utility-based privacy preserving methods should be able to deal with attributes of different types.

The beginning of the research in utility-based privacy preservation is not so old. Loukides and Shao [14] proposed a method that attempts to optimise the trade-off between utility and protection. They introduce a measure that captures both utility and protection, and an algorithm that exploits this measure using a combination of clustering and partitioning techniques. In a paper that appeared in KDD 2008, Brickell and Shmatikov [15] applied another solution to the tradeoff between privacy and utility. They define privacy loss of the published data as certain kinds of information learned by the adversary from the dataset and utility gain of the published data is defined as the same kinds of information learned by the researchers. Since both the adversary and the researchers see the same dataset and try to learn the same kinds of information, Brickell and Shmatikov claim that their knowledge gains are the same. Hence any utility gain by the anonymized data must be offset by the same amount of privacy loss. Finally, they reached an intriguing conclusion "even modest privacy gains require almost complete destruction of the data-mining utility." However, Tiancheng Li and Ninghui Li [1] criticized this study that it is inappropriate to directly compare privacy with utility, because of several reasons, including both technical and philosophical ones. The most important reason of this idea is presented that privacy is an individual concept, and utility is an aggregate concept. The anonymized dataset is safe to be published only when privacy for each individual is protected; on the other hand, utility gain adds up when multiple pieces of knowledge are learned. Secondly, they claims that even if the adversary and the researcher learn exactly the same information, one cannot conclude that privacy loss equals utility gain.

2.2.2. Privacy Preserving Data Mining

Privacy preserving data mining is a relatively new research area in the data mining community, counting approximately a decade of existence. It investigates the side-effects of data mining methods that originate from the penetration into the privacy of individuals and organizations. Since the pioneering work of Agrawal & Srikant [16] and Lindell & Pinkas [17] in the beginning of 2000s, several approaches have been proposed in the research literature for the offering of privacy in data mining. The majority of the proposed approaches can be classified along two principal research directions: (i) data hiding approaches and (ii) knowledge hiding approaches.

The first direction collects methodologies that investigate how the privacy of raw data, or information, can be maintained before the course of mining the data. The approaches of this category aim at the removal of confidential or private information from the original data prior to its disclosure and operate by applying techniques such as perturbation, sampling, generalization or suppression, transformation, etc. to generate a sanitized counterpart of the original dataset. Their ultimate goal is to enable the data holder receive accurate data mining results when is not provided with the real data or adhere to specific regulations pertaining to microdata publication (e.g., as is the case of publishing patient-specific data).

The second direction of approaches involves methodologies that aim to protect the sensitive data mining results (i.e., the extracted knowledge patterns) rather than the raw data itself, which were produced by the application of data mining tools on the original database. This direction of approaches mainly deals with distortion and blocking techniques that prohibit the leakage of sensitive knowledge patterns in the disclosed data, as well as with techniques for downgrading the effectiveness of classifiers in classification tasks, such that the produced classifiers do not reveal any sensitive knowledge [18].

2.2.4. A Motivating Example

The following example scenario, borrowed from the work of Verykios et al. [19], complements the real world examples we presented earlier and motivates the necessity of applying association rule hiding algorithms to protect sensitive association

rules against disclosure. 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, they find 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 find 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.

2.2.5. Association Rule Hiding

We focus on the knowledge hiding thread of privacy preserving data mining and study a specific class of approaches which are collectively known as *frequent itemset* and *association rule hiding* approaches. Other classes of approaches under the knowledge hiding thread include classification rule hiding, clustering model hiding, sequence hiding, and so on and so forth.

Association rules are implications that hold in a transactional database under certain user-specified parameters that account for their significance. Significant association rules provide knowledge to the data miner as they effectively summarize the data, while uncovering any hidden relations (among items) that hold in the data. The term “association rule hiding” has been mentioned for the first time in 1999 by Atallah et al. [20] in a workshop paper on knowledge and data engineering. The authors of this work studied the problem of modifying the original database in a way that certain (termed as “sensitive”) association rules disappear without, however, seriously affecting the original data and the nonsensitive rules. They proposed a number of solutions like fuzification of the original database, limiting access to the original data, as well as

releasing samples instead of the entire database. Due to the combinatorial nature of the problem, all of the solutions that the authors proposed, as well as the vast majority of solutions that have been proposed since then, were based on heuristics. Heuristic solutions, although computationally efficient and scalable, suffer from local optima issues as they require optimizing a specific function in each step of the algorithm, which however does not guarantee finding an optimal hiding solution to the whole problem. More recently, a new direction of exact approaches to association rule hiding has been proposed. These approaches have increased time and memory requirements but in return they offer quality guarantees on the identified solution [18].

2.2.5.1. Classes of Association Rule Hiding Algorithms

Association rule hiding algorithms can be divided into three classes, namely heuristic approaches, border-based approaches and exact approaches.

2.2.5.1.1. Heuristic Approaches

The first class of approaches involves efficient, fast algorithms that selectively sanitize a set of transactions from the original database to hide the sensitive association rules. Due to their efficiency and scalability, heuristic approaches have been the focus of attention for the majority of researchers in the data mining area. However, there are several circumstances in which these algorithms suffer from undesirable side-effects that lead them to identify approximate hiding solutions. This is due to fact that heuristics always aim at taking locally best decisions with respect to the hiding of the sensitive knowledge which, however, are not necessarily also globally best. As a result, heuristics fail to provide guarantees with respect to the quality of the identified hiding solution. Some of the most interesting heuristic methodologies for association rule hiding are presented in the next part of the book.

One of the earlier studies, which propose an algorithm for the hiding of sensitive association rules through the reduction in the support of their generating itemsets, belongs to Atallah et al. [20]. One of the most significant contributions of this work is the proof that the authors provide regarding the NP-hardness of finding an optimal

sanitization of a dataset. On the negative side, the proposed approach does not take into consideration the extend of loss in support for the large itemsets, as long as they remain frequent in the sanitized database.

Dasseni et al. [2] generalize the hiding problem in the sense that they consider the hiding of both sensitive frequent itemsets and sensitive association rules. The authors propose three single rule heuristic hiding algorithms that are based on the reduction of either the support or the confidence of the sensitive rules, but not both. In all three approaches, the goal is to hide the sensitive rules while minimally affecting the support of the nonsensitive itemsets. In order to achieve this, transactions are modified by removing some items, or inserting new items depending on the hiding strategy. A basic drawback of the proposed methodologies is the strong assumption that all the items appearing in a sensitive association rule do not appear in any other sensitive rule. Under this assumption, hiding of the sensitive rules one at a time or altogether makes no difference. Moreover, the proposed methodologies fail to avoid undesired side-effects, such as lost rules and ghost rules.

Verykios et al. [19] extend the previous work of Dasseni et al. [2] by improving and evaluating the association rule hiding algorithms of [2] for their performance under different sizes of input datasets and different sets of sensitive rules. In addition to that, the authors propose two novel heuristic algorithms that incorporate the third strategy presented above. The first of these algorithms protects the sensitive knowledge by hiding the item having the maximum support from the minimum length transaction (i.e., the one with the least supporting items). The hiding of the generating itemsets of the sensitive rules is performed in a decreasing order of size and support and in a one-by-one fashion. Similarly to the first algorithm, the second algorithm first sorts the generating itemsets with respect to their size and support, and then hides them in a round-robin fashion as follows. First, for each generating itemset, a random ordering of its items and of its supporting transactions is attained. Then, the algorithm proceeds to remove the items from the corresponding transactions in a round-robin fashion, until the support of the sensitive itemset drops below the minimum support threshold, thus the itemset is hidden. The intuition behind hiding in a round-robin fashion is fairness and the proposed algorithm, although rather naïve, serves as a baseline for conducting a series of experiments.

Oliveira and Zaïane [21] contribute to this area with four heuristic algorithms. They identify transactions to sanitize based on the number of sensitive item sets

supported by a transaction. Their naïve algorithm removes all the items in a sensitive item set supported by the transaction, retaining only the item with the highest frequency in cases where all the items in the transaction are removed. Their minimum frequency item algorithm removes the item with the smallest support in a sensitive item set supported by the transaction, while their maximum frequency item algorithm removes the item with the greatest support. Their item grouping algorithm (IGA) relies on grouping sensitive patterns based on item sets sharing the same set of items. All four approaches allow for some sensitive item sets not to be hidden, and this may not be appropriate in many situations.

Amiri [22] presents three data sanitization heuristics that demonstrate high data utility at the expense of computational speed. The first heuristic reduces the support of the sensitive item sets by deleting a set of supporting transactions. The second approach modifies, instead of deleting, the supporting transactions by removing some items until the sensitive item sets are protected. The third algorithm aggregates the previous two by using the first approach to identify the sensitive transactions and the second one to remove items from these transactions, until the sensitive knowledge is hidden. Although the algorithms by Amiri are similar in philosophy to the previous approaches, the three proposed methodologies do a better job in modeling the overall objective of a rule hiding algorithm.

We examine Item Grouping Algorithm (IGA) algorithm, which the heuristic part of our new approach is derived from, in detail to compare it with our approach.

2.2.5.1.1.1. The Item Grouping Algorithm (IGA)

The main idea behind the Item Grouping Algorithm, denoted by IGA, is to group sensitive patterns in groups of itemsets sharing the same itemsets. If two sensitive patterns intersect, by sanitizing the conflicting sensitive transactions containing both sensitive patterns, one would take care of hiding these two sensitive patterns at once and consequently reduce the impact on the released database. However, clustering the sensitive patterns based on the intersections between patterns leads to groups that overlap since the intersection of itemsets is not transitive. By solving the overlap between clusters and thus isolating the groups, a representative of the itemset linking related to the patterns in the group, all sensitive patterns in the group will be hidden in one step. This again will minimize the impact on the database and reduce the potential

accidental hiding of legitimate patterns. The sketch of the Item Algorithm is given as follows in Figure 2.6.

Step 1. For each restrictive pattern $rp_i \in R_P$ do
1. $T[rp_i] \leftarrow$Find_Sensitive_Transactions(rp_i, D);
Step 2.
1.Group restrictive patterns in a set of groups GP such that $\forall\ G \in GP, \forall\ rp_i, rp_j \in G$, rp_i and rp_j share the same itemset I. Give the class label α to G such that $\alpha \in I$ and $\forall \beta \in I$, sup(α, D) $\leq$ sup(β, D).
2. Order the groups in GP by size in terms of number of restrictive patterns in the group.
3. Compare groups pairwise G_i and G_j starting with the largest. For all $rp_k \in G_i \cap G_j$ do
3.1. if size(G_i) $\neq$ size(G_j) then remove rp_k from smallest(G_i, G_j)
3.2. else remove rp_k from group with class label α such that sup(α, D) $\leq$ sup(β, D) and α, β are class labels of either G_i or G_j
4. For each restrictive pattern $rp_i \in R_P$ do
4.1. $Victim_{rp_i} \leftarrow \alpha$ such that α is the class label of G and $rp_i \in G$
Step 3. For each restrictive pattern $rp_i \in R_P$ do
1. $NumTrans_{rp_i} \leftarrow |T[rp_i]| \times (1 - \psi)$ // $|T[rp_i]|$ is the number of sensitive transactions for rp_i
Step 4. $D' \leftarrow D$
For each restrictive pattern $rp_i \in R_P$ do
1. Sort_Transactions($T[rp_i]$); //in descending order of degree of conflict
2. $TransToSanitize \leftarrow$ Select first $NumTrans_{rp_i}$ transactions from $T[rp_i]$
3. in D' foreach transaction $t \in TransToSanitize$ do
3.1. $t \leftarrow (t - Victim_{rp_i})$
End

Figure 2.6. Item Grouping Algorithm [21].

We implement IGA on the database whose transactions and items are shown in Table 2. 1. Initially, we eliminate transactions that does not have any sensitive itemsets and give them ids in a way of beginning with ‘0’ (‘a’ part in Figure 2.7). Then, we map sensitive itemsets with sensitive transactions which support related sensitive itemsets (‘b’ part in Figure 2.7). Degree-of-conflict, which is the number of restrictive patterns that can be mined from the sensitive transaction, of these transactions are calculated and sensitive transactions that have relevant sensitive transaction are sorted descendingly (‘a’ and ‘c’ part in Figure 2.7). Then, based on the support value of sensitive itemsets in the dataset, we find the number of transactions to sanitize and get the support value of sensitive itemsets decreased below minimum support threshold (‘d’ part in Figure 2.7). Transactions that will be sanitized are identified from sorted list based on this founded number.

a)

Example Database *D*		
Id	**Transactions**	**Degree-of-conflict**
0	1 2 3 7 8 10	2
	3 9 10	
	4 5 6	
1	1 2 3 6 7 8 9	4
2	1 2 3 6 7	2
	10	
	4	
3	3 6 7 8 9	3
4	3 8 9	2
5	5 6 7	1

b)

{8 9} -> 1, 3, 4
{8 3} -> 0, 1, 3, 4
{6 7} -> 1, 2, 3, 5
{1 2 3} -> 0, 1, 2

c)

{8 9} -> 1, 3, 4
-> 1, 3, 4
{8 3} -> 0, 1, 3, 4
-> 1, 3, 0, 4
{6 7} -> 1, 2, 3, 5
-> 1, 3, 2, 5
{1 2 3} -> 0, 1, 2
-> 1, 0, 2

d)

Sensitive Itemset	**Transaction ids**	**Reqired number of transactions**
{8 9}	1, 3	2
{8 3}	1, 3, 0	3
{6 7}	1, 3, 2	3
{1 2 3}	1, 0	2

Figure 2.7. Finding victim transactions of our example in IGA.

After some preprocess to find transactions that will sanitize to hide each sensitive itemsets, we group sensitive itemsets based on shared itemsets that sensitive itemsets have. We give a label, that is the least supported item of shared itemset by all transactions in the dataset, to each group. Sensitive itemsets, that does not have any shared itemsets, are accepted as a singleton group. The least supported item of this kind of sensitive itemset is chosen as a label to the singleton group (Table 2. 4). Then, groups are sorted descendingly based on the size of groups. In our example, there is no change, since groups are already lined up descendingly. Then, we try to solve overlapped itemsets problems. If there is a sensitive itemset that is in more than one group, there are two considerations. If the number of elements in groups of the sensitive itemset are different, overlapped sensitive itemsets are removed from the group that has less

number of elements. If the size of groups is the same, the sensitive itemset is removed from the group which has the least support value of label – item (Table 2. 5).

Table 2. 4. Grouping sensitive itemsets of our example in IGA.

Group Id	Group Label	Sensitive Itemsets
0	8	{8 9}, {8 3}
1	3	{8 3}, {1 2 3}
2	6	{6 7}

Table 2. 5. Solving overlapped itemset problem of our example in IGA.

Group Label	Support Value of Label (%)	Group Elements
8	40	{8 9}, {8 3}
3	60	{8 3}, {1 2 3}
6		{6 7}

After finding transactions that will be sanitized and grouping sensitive itemsets based on shared itemsets, sanitization is made based on these information (Table 2. 6). By removing the victim item (label) from the sensitive transactions related to the sensitive itemsets in the group, all sensitive itemsets in the group are hidden in one step. When all transaction are put in sanitization process, the sanitized dataset is generated with the number of modifications on the dataset "D" is 8. 4 sensitive transactions are sanitized after all process and 7 nonsensitive itemsets were previously frequent still are frequent whereas 23 nonsensitive itemsets were frequent are no longer frequent.

Table 2. 6. All required information is prepared to use in sanitization for our example in IGA.

Sensitive Itemset	Victim Transactions	Group Label
{8 9}	1, 3	8
{8 3}	1, 3, 0	3
{6 7}	1, 3, 2	6
{1 2 3}	1, 0	3

2.2.5.1.2. Border-Based Approaches

The second class of approaches considers association rule hiding through the modification of the borders in the lattice of the frequent and the infrequent itemsets of the original database. Borders [23] capture those itemsets of a lattice that control the position of the borderline separating the frequent itemsets from their infrequent counterparts. Data modification, applied to the original database to facilitate sensitive knowledge hiding, has an immediate effect on the support of the various itemsets and, subsequently, on the borders of the sanitized database. Border-based algorithms achieve to hide the sensitive association rules by tracking the border of the nonsensitive frequent itemsets and greedily applying the data modifications that have minimal impact on the quality of the border to accommodate the hiding of the sensitive rules. The algorithms in this class differ in the methodology they follow to enforce the new, revised borders, in the sanitized database.

Sun & Yu [3 and 4] proposed the first frequent itemset hiding methodology in 2005. Their methodology is based on the notion of the border constructed by the non-sensitive frequent itemsets in an attempt to track the impact of altering transactions. Instead of considering each non-sensitive itemset individually, their algorithm focuses on preserving the quality of the resulting border.

Gkoulalas-Divanis and Verykios used this border concept in their works [5, 6 and 7] to minimize overconcealing nonsensitive itemsets. They capture the itemsets hiding process as a border revision operation and they presented a set of algorithms which enable the computation of the revised borders that pertain to an exact hiding solution.

2.2.5.1.3. Exact Approaches

The third class of approaches contains non-heuristic algorithms which conceive the hiding process as a constraints satisfaction problem (CSP) that is a kind of optimization problem and is solved by using integer programming. The main difference of these approaches, when compared to those of the two previous classes, is the fact that the sanitization process is capable of offering quality guarantees for the computed

hiding solution. The modeling of the hiding problem as an optimization problem enables the algorithms of this category to identify optimal hiding solutions that minimally distort the original database as well as introduce no side-effects to the hiding process. Exact hiding approaches can be considered as the descendant of borderbased methodologies. They operate by first applying border revision to compute a small portion of itemsets from the original database whose status (i.e., frequent vs. infrequent) in the sanitized database plays a crucial role to the quality of the hiding solution. Having computed those itemsets, the exact methodologies incorporate unknowns to the original database and generate inequalities that control the status of selected itemsets of the border. These inequalities along with an optimization criterion that requires minimal modification of the original database to facilitate sensitive knowledge hiding, formulate an optimization problem whose solution (if exists) is guaranteed to lead to optimal hiding. It is important to mention that unlike the two previous classes of approaches that operate in a heuristic manner, exact hiding methodologies achieve to model the hiding problem in a way that allows them to simultaneously hide all the sensitive knowledge as an atomic operation (from the viewpoint of the hiding process). On the negative side, these these approaches are usually several orders of magnitude slower than the heuristic ones, especially due to the time that is taken by the integer programming solver to solve the optimization problem.

The paper by Menon et al. [8] propose an interesting approach to the problem of privacy preserving data mining. They were the first to present an integer programming optimization method that consisted of an exact and a heuristic part to hide frequent itemsets. The exact part of the method uses the database to formulate an integer program trying to obtain the minimum number of transactions that have to be sanitized. The optimization process of solving the CSP is driven by a criterion function that is inspired from the measure of accuracy [24], essentially penalizing the hiding algorithm based on the number of transactions that are sanitized from the original database to accommodate the hiding of the sensitive itemsets. The constraints imposed in the integer programming formulation aim to capture the transactions of the database that need to be sanitized for the hiding of each sensitive itemset. An integer programming solver is then used to find the solution of the CSP that optimizes the objective function and to derive the value of the objective. In turn, the heuristic algorithm uses this information to perform the actual sanitization of the database. Using integer programming and heuristic together gives evaluated impact on the result data than

border-based approaches give. However, it causes failing to notice side-effect of information loss.

The study of Gkoulalas-Divanis and Verykios [5] includes the inline algorithm that has been proposed which does not rely on any heuristics to secure the sensitive knowledge derived by association rule mining. Similarly to the algorithm of Menon et al. [8], the algorithm in [5] aims to hide the sensitive frequent itemsets of the original database that can lead to the production of sensitive association rules. They introduce the notion of distance between two databases along with a measure for quantifying it. The quantification of distance provides us with important knowledge regarding the minimum data modification that is necessary to be induced to the original database to facilitate the hiding of the sensitive itemsets, while minimally affecting the nonsensitive ones. By trying to minimize the distance between the original database and its sanitized counterpart, the inline algorithm formulates the hiding process as an optimization problem in which distance is the optimization criterion. The attained solution is such that the distance measure is minimized.

Gkoulalas-Divanis and Verykios present a two–phase iterative algorithm in the study [6] that extends the functionality of the inline algorithm of [5] to allow for the identification of exact hiding solutions for a wider spectrum of problem instances. In their another study [7], they introduce very interesting exact methodology that includes hybrid database generation. This approach extends the regular process of data sanitization by applying an extension to the original database instead of either modifying existing transactions. Although a framework for decomposition and parallelization of exact hiding approaches has been recently proposed in [25], lack of solutions that can be used in all circumstances for the complexity problems of exact methods and lack of practical approaches that have proposed up to present in this area are the reasons of researching for better solution in privacy preserving data mining.

CHAPTER 3

COEFFICIENT-BASED ITEMSET HIDING SOLUTION FOR ASSOCIATION RULE HIDING

In this section, we introduce a novel method for the itemset hiding problem. We first define the problem using integer programming and then simply augment the objective function with coefficients in order to reduce the information loss. To clarify this new method and the approach of Menon et al. which is the starting point of our study, we implement both on the database whose transactions and items are shown in Table 2. 1.

Modeling the itemset hiding problem with integer programming can be done in several ways. We follow the way of Menon et al. [8] such that the objective achieves the maximum accuracy. Our method alters the objective function such that the binary variables indicating whether a transaction is chosen or not are multiplied by some pre-computed coefficients that reflect the amount of information loss. We compute the coefficients of transactions that support sensitive patterns only.

Based on the example database presented on Table 2. 1 and Table 2. 2 in Section 2.1, we start with creating the constraint matrix by eliminating transactions, which do not support sensitive itemsets, from consideration as shown below:

$$\begin{pmatrix} & t_1 & t_2 & t_3 & t_4 & t_5 & t_6 \\ r_1 & 0 & 1 & 0 & 1 & 1 & 0 \\ r_2 & 1 & 1 & 0 & 1 & 1 & 0 \\ r_3 & 0 & 1 & 1 & 1 & 0 & 1 \\ r_4 & 1 & 1 & 1 & 0 & 0 & 0 \end{pmatrix} \quad (1)$$

6 columns represents respectively: $t_1 \rightarrow T_1, t_2 \rightarrow T_4, t_3 \rightarrow T_5, t_4 \rightarrow T_8, t_5 \rightarrow T_9, t_6 \rightarrow T_{10}$. While t_1 supports sensitive itemsets r_2 and r_4, t_6 contains only one sensitive itemset that is r_3.

3.1. The Approach of Menon et al. Implementation

In this section, we implement the approach that is presented in the study of Menon et al. [8] which models the itemset hiding problem with integer programming as we do in our new method. Initally, give a_{ij} a binary value. Be 1 if transaction $i \in D$ supports itemset $j \in F^R(\sigma_{min})$. Otherwise, the value of a_{ij} is 0. For the variable x_i, it will be set to 1 if transaction $i \in D$ is sanitized. Otherwise, the value of x_i is 0. σ_j represents the current support for itemset $j \in F(\sigma_{min})$. For the approach of Menon et al., the frequent item set formulation is :

$$min \sum_{i \in D} x_i , \tag{2}$$

$$s.t. \sum_{i \in D} a_{ij} x_i \geq \sigma_j - \sigma_{min} + 1 \qquad \forall j \in F^R(\sigma_{min}), \tag{3}$$

$$x_i \in \{0,1\} \quad \forall i \in D. \tag{4}$$

When the formulation is arranged based on the constraint matrix that is created in previous section, the integer programming formulation is generated as:

$$\min x_1 + x_2 + x_3 + x_4 + x_5 + x_6,$$
$$s.t \quad x_2 + x_4 + x_5 \geq 2, \qquad (r_1)$$
$$x_1 + x_2 + x_4 + x_5 \geq 3, \qquad (r_2)$$
$$x_2 + x_3 + x_4 + x_6 \geq 3, \qquad (r_3)$$
$$x_1 + x_2 + x_3 \geq 2, \qquad (r_4)$$
$$x_1, x_2, x_3, x_4, x_5, x_6 \in \{0,1\}.$$

When this integer program is solved, the result in the optimal solution shows $x_1 = x_2 = x_3 = x_4 = 1$ with the other variables being 0. The accuracy of the resulting sanitized database is 0.60.

We choose transaction T_1 that is represented in the constraint matrix by t_1 to explain the intelligent sanitization approach [8]. T_1 supports two sensitive itemsets - r_2 and r_4. First, the item appearing in the most number of sensitive patterns supported by that transaction is selected from all items in $r_2 \cup r_4$ (Items 1, 2, 3, 8). For the example, "3" is selected, because it appears twice while each one of the others appears once. Delete "3" and remove all sensitive itemsets that contain selected item. This action

eliminates r_2 and r_4 at the same time. If sensitive item sets remain supported by T_1, repeat the procedure. For our example, there is no sensitive item set left. The sanitized transaction is {1, 2, 7, 8, 10}. When all transaction are put in process, the sanitized database is generated with the number of modifications on the database D is 8. After the sanitization, 8 nonsensitive itemsets were previously frequent still are frequent whereas 22 nonsensitive itemsets were frequent are no longer frequent.

3.2. Coefficient-Based Itemset Hiding

Our new method consists of three essential parts: Coefficient Computation, Integer Programming Solution and Heuristic Sanitization.

3.2.1. Coefficient Computation

To minimize the impact on nonsensitive frequent itemsets, coefficient computation is made for each transaction which supports sensitive patterns as the first step. A coefficient of the transaction gives the information about a risk of overconcealing nonsensitive frequent itemsets on this transaction. If the value of coefficient is high, it means that the number of concealed nonsensitive itemsets included in the transaction would be high after the sanitization.

1: **for** transactions $i \in D$ such that i is to be sanitized
2: **identify** all sensitive frequent item sets $F_i^R \in F^R(\sigma_{min})$ supported by i
3: **identify** all nonsensitive frequent item sets $F_i^n \in F^n(\sigma_{min})$ supported by i
4: **while** $F_i^R \neq \emptyset$
5: **calculate** $f_j = |\{k \in F_i^R \mid j \in k\}|$, ∀ items j in i
6: **calculate** $j^* = argmax_j\{f_j\}$
7: **calculate** $g_j = |\{k \in F_i^n \mid j^* \in k\}|$, ∀ items j in i
8: **update** coefficient $c_i = c_i + g_j$
9: **update** $F_i^R = F_i^R \backslash \{k \in F_i^R | j^* \in k\}$
10: **end while**
11: **end for**

Figure 3.1. Coefficient Computation Algorithm.

This coefficient computation is typically organized by taking into account that initial utility worth of each nonsensitive itemsets on studied database is the same. Calculating a risk of overconcealing nonsensitive frequent itemsets is made based on this assumption. If some information on the database has different worth of utility based on the area where the shared database is utilized, relevant coefficients can be computed independently of Coefficient Computation Algorithm thereby paying regard to requirements of the area.

A coefficient, which is represented by the Coefficient Computation Algorithm in Figure 3.1 as "c_i", is calculated for each transaction that is included in the constraint matrix. For example, we may choose transaction T_1 to explain this calculation (on line 1). First, sensitive frequent itemsets and nonsensitive frequent itemsets are identified for transaction T_1 (on line 2 and 3). The item appearing in the most number of sensitive patterns supported by that transaction is selected from all items in $r_2 \cup r_4$ (Items 1, 2, 3, 8) (on line 5 and 6). The item "3" is selected. Record the number of appearances of the item "3" in the non-sensitive frequent itemsets supported by the transaction (on line 7). For our example, there are 6 appearances of item "3" in the non-sensitive frequent itemsets such as {1, 3}, {2, 3}, {10, 3}, {7, 3}, {1, 7, 3}, {2, 7, 3}. Remove all sensitive itemsets supported by the transaction contain the selected item "3" (on line 9). If sensitive itemsets remain supported by T_1, repeat the procedure (on line 4) and sum the appearances of new selected item in the non-sensitive frequent itemsets supported by the transaction with recorded value (on line 8). There is no sensitive itemset left in our example. Hence, the total summation for transaction T_1 is 6. After, all transactions which are included in the constraint matrix are taken in consideration based on this procedure, a coefficient for each transaction is found such as $T_1 \rightarrow 6$, $T_4 \rightarrow 29$, $T_5 \rightarrow 14$, $T_8 \rightarrow 6$, $T_9 \rightarrow 0$, $T_{10} \rightarrow 1$.

3.2.2. Integer Programming Solution

In this section we describe the integer programming formulation which is similar to the solution of Menon et al. [8] to solve Coefficient-Based Itemset Hiding problem. Initally, give a_{ij} a binary value. Be 1 if transaction $i \in D$ supports itemset $j \in F^R(\sigma_{min})$. Otherwise, the value of a_{ij} is 0. For the variable x_i, it will be set to 1 if transaction $i \in D$ is sanitized. Otherwise, the value of x_i is 0. σ_j represents the current

support for itemset $j \in F(\sigma_{min})$. Recall that c_i is the coefficient that is calculated in Coefficient Computation for transaction $i \in D$, which contains at least one sensitive itemset.

In the light of this information, the formulation is generated as below:

$$min \ \sum_{i \in D} c_i x_i \,, \tag{5}$$

$$s.t. \ \sum_{i \in D} a_{ij} \, x_i \geq \ \sigma_j - \ \sigma_{min} + \ 1 \qquad \forall j \ \in F^R(\sigma_{min}), \tag{6}$$

$$x_i \ \in \{0,1\} \ \ \forall i \ \in D. \tag{7}$$

Equation (5) represents the objective function minimizes the number of transactions sanitized. Equation (6) includes the constraint that more than $(\sigma_j - \sigma_{min})$ transactions supporting each sensitive itemset have to be sanitized, that's why this line is generated for each sensitive itemset. Equation (7) imposes that x_i has only binary value. The integer programming formulation is reorganized based on the constraint matrix (Eq.1) and coefficients that are gained by Coefficient Computation in the previous section as:

$$min \ 6x_1 + 29x_2 + 14x_3 + 6x_4 + 0x_5 + 1x_6,$$

$$\begin{aligned} s.t. \quad & x_2 + x_4 + x_5 \ \geq \ 2, && (r_1) \\ & x_1 + x_2 + x_4 + x_5 \ \geq \ 3, && (r_2) \\ & x_2 + x_3 + x_4 + x_6 \ \geq \ 3, && (r_3) \\ & x_1 + x_2 + x_3 \ \geq \ 2, && (r_4) \\ & x_1, x_2, x_3, x_4, x_5, x_6 \ \in \{0,1\}. \end{aligned}$$

Solving this integer program results in optimal solution $x_1 = x_3 = x_4 = x_5 = x_6 = 1$ with the other variables being 0. The accuracy of the resulting sanitized database is 0.50.

3.2.3. Heuristic Sanitization

Sanitization is a kind of process that includes removing items from a transaction; thereby the sanitized version of the transaction supports no itemset in $F^R(\sigma_{min})$. There are various sanitization approaches in the privacy preserving data mining literature. For

instance, Verykios et al. offered two sanitization techniques in their study [19]. These are generally based on hiding itemsets that are already sorted with respect to their size and support, in a different fashion such as one-by-one and round-robin. Amiri [22] presented the Aggregate Algorithm based on removing the most sensitive and the least nonsensitive itemsets in selected transaction. The process is repeated until all the sensitive itemsets are hidden. Furthermore, three item restriction-based algorithms [21] that are known as Minimum Frequency Item Algorithm (MinFIA), Maximum Frequency Item Algorithm (MaxFIA) and Item Grouping Algorithm (IGA) selectively remove items from transactions that support the sensitive sensitive itemsets. Intelligent sanitization in the paper [8] is the variant of their IGA.

We do not focus on the development of the sanitization techniques. Since we compare our new method with the study of Menon et al. [8], we prefer to use one of heuristics in their study. One is blanket sanitization where only one item is retained from the original transaction. The sanitization occurs by eliminating support for every nonsingleton itemset supported by the transaction. The other is intelligent sanitization, where an attempt is made to remove the fewest number of items from the transaction that would result in eliminating the support for every itemset in $F^R(\sigma_{min})$.

It is shown that the intelligent sanitization produces less distortion on nonsensitive itemsets thereby removing less number of items when this is compared with the blanket sanitization. So we prefer to use the intelligent sanitization to hide itemsets of transactions that are identified by the method described in the previous section. In order to be self-contained, intelligent sanitization algorithm is given in Figure 3.2.

1: **for** transactions $i \in D$ such that i is to be sanitized
2: **identify** all sensitive frequent item sets $F_i^R \in F^R(\sigma_{min})$ supported by i
3: **while** $F_i^R \neq \emptyset$
4: **calculate** $f_j = |\{k \in F_i^R \mid j \in k\}|$, $\forall$ items j in i
5: **remove** item $j^* = argmax_j\{f_j\} \in m$
6: **update** $F_i^R = F_i^R \backslash \{k \in F_i^R | j^* \in k\}$
7: **end while**
8: **end for**

Figure 3.2. Intelligent Sanitization [8].

Let us explain this, we choose transaction T_1 that is represented in the constraint matrix by t_1 to exemplify this approach (on line 1). T_1 supports two sensitive itemsets - r_2 and r_4 (on line 2). First, the item appearing in the most number of sensitive patterns supported by that transaction is selected from all items in $r_2 \cup r_4$ (Items 1, 2, 3, 8). For the example, "3" is selected, because it appears twice while each one of the others appears once. Delete "3" (on line 4 and 5) and remove all sensitive itemsets that contain selected item (on line 6). This action eliminates r_2 and r_4 at the same time. If sensitive itemsets remain supported by T_1, repeat the procedure (on line 3). For our example, there is no sensitive itemset left. The sanitized transaction is {1, 2, 7, 8, 10}. When all transaction are put in process, the sanitized database is generated with the number of modifications on the database *D* is 7. After the entire process, 17 nonsensitive itemsets were previously frequent still are frequent, although 13 nonsensitive itemsets that were frequent are no longer frequent.

CHAPTER 4

PERFORMANCE EVALUATION

We performed Coefficient-Based Itemset Hiding, the study of Menon et al. [8] and Item Grouping Algorithm [21] on real datasets using different parameters such as number of sensitive itemsets and minimum support value. Our code was implemented in Java on a Windows 7 - PC with Intel Core i5, 2.67 GHz processor. We performed exact parts of the experiments by using GNU GLPK [26]. In this section, features of datasets, selected parameters and results are explained in detail.

4.1. The Datasets

All datasets we use in our experiments are available through Frequent Itemset Mining Implementations Repository – FIMI. We chose the datasets which wu use in experiments in different level of on sparsity and density. Sparsity and density are terms used to describe the percentage of cells in a database table that are not populated and populated, respectively. The sum of the sparsity and density should equal 100%.

A table that is 10% dense has 10% of its cells populated with non-zero values. It is therefore 90% sparse – meaning that 90% of its cells are either not filled with data or are zeros.

Similarity of transactions in databases is an important speciality for our experiment. To measure this value, some different approach can be developed. We developed the algorithm in Figure 4.1.

The kosarak dataset which is provided by Ferenc Bodon [27] is a very large dataset containing 990,002 sequences of click-stream data from a Hungarian on-line news portal. The level of density for the kosarak is 0.02%. The retail dataset is a dataset that was reported in Brijs et al. 1999 [28]. It includes the retail market basket data from an anonymous Belgian retail store. The density for the retail is approximately 0.06% that signifies medium sparsity. The mushroom, which was generated by Roberto Bayardo from the UCI datasets and PUMSB [29], has transactions which are formed of similar items. It is a 19.33%. Furthermore, these datasets have different characteristics

such as the number of transactions, varieties of items and the number of frequent itemsets for different minimum support threshold.

```
for transactions i ∈ D
    for transactions j ∈ D such that i ≠ j
        for items m ∈ i
            increment counter if m is contained by j
        end for
    end for
    add counter / (number of transactions – 1) to the list l
end for
for values v ∈ l
    calculate result = result + v
end for
calculate output = ( result / size of l ) /average length of transactions
∈ D
```

Figure 4.1. Calculation of similarity of transactions in a database.

These variations contribute to our experiment and give a chance to measure the efficacy of our study. Table 4.1 includes summary information about these datasets.

Table 4.1. Characteristics of the Real Datasets

Database name	Number of transactions	Number of items	Avg. transansation. lenght	Number of nonsingleton frequent itemsets	Similarity of transactions
kosarak	990,002	41,270	8.10	1,462	0.05090
retail	88,162	16,470	10.30	5,472 15,316	0.03322
mushroom	8,124	119	23.00	53,540	0.46160

4.2. Evaluation Methodology

We compare our approach with the approach of Menon et al. [8] and Item Grouping Algorithm [21] in terms of the ratio of the number of transactions that are not sanitized and the total number of transactions in the database (accuracy), the percentage of items that are lost from the database (distance [5]), the percentage of of nonsensitive frequent itemsets that are lost (information loss) and the number of seconds (time) that is spent to complete whole process.

Execution times for coefficient computation (C), integer programming (IP) and heuristic sanitization (H) are separately recorded to maintain the total time for Coefficient-Based Itemset Hiding. Because the complexity is one of main problems for the privacy preserving data mining, the time is illustrated in detail in our experiments.

With our original sensitive itemsets, supersets of them are become hidden in the databases since any itemsets that contains sensitive itemsets should also be hidden. Original sensitive itemsets are specified with various lengths such as 10, 20 and 50. In addition, we use two different minimum support thresholds for the retail dataset to evaluate the impact of threshold.

4.3. Experimental Results

Results are categorized in terms of metric used in experiments such as accuracy, distance, time and information loss in tables 4.2, 4.3, 4.4 and 4.5 as well as in terms of approach such as Item Grouping Algorithm [21], Menon et al. [8] and Coefficient-Based Itemset Hiding in tables of Figure 4.2. Since our new approach have some characteristics, that are the same with the study of Menon et al. has, such as an integer programming modelling and intelligent sanitization, Table 4.6 is prepared for the close look of result comparison of these two solutions. For Table 4.6, negative values represent the sacrifices of our approach for gains on other metrics, while positive values show outclass performance of our new method over the approach of Menon et al.

When the tables are examined well, it is deduced that the performance of IGA falls behind the performance of exact algorithms in experiments. Although, there are some good performances on accuracy and distance, worst results on the information loss are belonged to IGA in every comparison.

Our new method – Coefficient-based Itemset Hiding makes progress with different fluctuations based on the characteristics of databases used in experiments. Firstly, it can be noticed that the proposed method decreased the number of lost nonsensitive frequent itemsets successfully for all kinds of databases in the experiments. It is obviously seen that our approach works more powerfully on databases whose the similarity of transactions are not so high such as Retail and Kosarak. Also, minimum support level used in experiments may be another critical point. Because decreasing the support value of itemsets below very low support level needs sacrificing more utility

gain, specifying low support level for itemset hiding reduces the coefficient benefit for information loss as Retail performance result at the support level – 44 (0.05%). On the other hand, the best performance in all experiments is attained for Retail database at the level of support - 88 (0.10%). Moreover, low minimum support threshold means high number of frequent itemsets. It causes higher time cost on our solution than caused on the results of other solutions in the experiments.

The performance result of Kosarak database shows that new approach works well with databases which have high number of transactions. It has up to 80% gain on information loss while there is not above 2% accuracy loss on Table 4.6. On other hand, since Kosarak is a very large database, total time cost is a general problem for integer programming solutions. Despite this, it is also remarkable that in case of 50 sensitive itemsets in Kosarak, our method is approximately six times better in execution time than the closest result that Menon approach performs. This is quite reasonable since coefficients help branch and cut algorithms of integer programming [30] by allowing more cuts.

When the result of Mushroom database in Table 4.5. is examined, it is deduced that although our new approach is the best solution in terms of the information loss, the performance is not very powerful as been in the results of experiments on other databases. As being similar with the experiment at the support level – 44 (0.05%) on Retail, low support level – 1,625 (20%) is used in the database whose the similarity of transactions is high. Lots of frequent itemsets are generated by the reason of this support level. Hence, our new approach sacrifices more utility gain and information loss to decrease the support value of itemsets below such a low support threshold. Also, the accuracy performance is dramatically decreased, but the distance value is not so high when we compare with the results of the study of Menon et al. It means that although the number of sanitized transaction is high, the dissimilarity between original and sanitized databases is not so high to be accepted as insufficient performance. This is the result of working on the database whose the similarity of transactions is very high. Not to lose nonsensitive frequent itemsets, the approach removes minimum number of items from each victim transaction. Hence, lots of sanitized transactions are used to hide all sensitive itemsets. High value of total time cost is another remarkable point of experiments on Mushroom. In consequence of Coffiecient-based Solution experiment results, it is obvious that completing coefficient calculation takes too long with specified minimum support threshold (20%), which generates large amounts of frequent itemsets,

on Mushroom. Hence, process on these frequent itemsets raises total time cost. As a result, our new approach has difficulty on some experiments, that is made with low minimum support threshold, and on some databases, whose the similarity of transactions is high. such as Mushroom. In spite of this, the approach gives successful results on most experiments.

Table 4. 2. Comparison between approaches in terms of accuracy

DB name ($б_{min}$)	Sensitive itemsets (with supersets)	Accuracy (%)		
		IGA	Menon	Coefficient-based
kosarak (4,950)	10 (18)	99.54	99.59	99.27
	20 (31)	99.05	99.23	98.5
	50 (65)	98.57	98.95	97
retail (88)	10 (10)	99.84	99.83	99.8
	20 (20)	99.58	99.6	99.42
	50 (65)	98.98	99.05	98.43
retail (44)	10 (15)	99.38	99.37	99.34
	20 (32)	98.8	98.77	98.54
	50 (97)	97.49	97.46	96.62
mushroom (1,625)	10 (2336)	93.4	93.4	93.4
	20 (2395)	88.57	93.16	84.94
	50 (5341)	90.65	92.32	79.47

Table 4. 3. Comparison between approaches in terms of total time cost.

DB name ($Б_{min}$)	Sensitive itemsets (with supersets)	Total Time (sec)		
		IGA	Menon	Coefficient-based
kosarak (4,950)	10 (18)	115	140.5	206.3
	20 (31)	146	153.7	324.7
	50 (65)	233	36,091	7,023
retail (88)	10 (10)	8	7	9.1
	20 (20)	9	8.1	9.1
	50 (65)	10	8.1	11.3
retail (44)	10 (15)	9	7.1	9.1
	20 (32)	9	8.1	12.1
	50 (97)	10	8.2	16.3
mushroom (1,625)	10 (2336)	1	1.1	90.1
	20 (2395)	1	1.2	115.5
	50 (5341)	8	1.3	142.8

Table 4.4. Comparison between approaches in terms of distance.

DB name ($Б_{min}$)	Sensitive itemsets (with supersets)	Distance (%)		
		IGA	Menon	Coefficient-based
kosarak (4,950)	10 (18)	0.077	0.069	0.093
	20 (31)	0.15	0.133	0.19
	50 (65)	0.3	0.221	0.391
retail (88)	10 (10)	0.016	0.017	0.019
	20 (20)	0.045	0.039	0.056
	50 (65)	0.154	0.109	0.154
retail (44)	10 (15)	0.062	0.062	0.065
	20 (32)	0.126	0.12	0.142
	50 (97)	0.346	0.265	0.335
mushroom (1,625)	10 (2336)	0.7	0.441	0.441
	20 (2395)	1.527	0.797	0.762
	50 (5341)	1.184	1.16	1.52

Table 4.5. Comparison between approaches in terms of information loss.

DB name (σ_{min})	Sensitive itemsets (with supersets)	Information loss (%)		
		IGA	Menon	Coefficient-based
kosarak (4,950)	10 (18)	13.504	6.787	1.316
	20 (31)	25.297	12.718	3.983
	50 (65)	34.789	22.19	4.152
retail (88)	10 (10)	0.732	0.183	0.037
	20 (20)	2.128	1.119	0.183
	50 (65)	5.437	1.813	0.481
retail (44)	10 (15)	2.163	0.556	0.281
	20 (32)	4.874	2.192	1.282
	50 (97)	11.197	4.363	2.392
mushroom (1,625)	10 (2336)	63.312	39.028	38.247
	20 (2395)	73.401	64.618	52.382
	50 (5341)	76.101	74.34	64.626

A)

DB name ($б_{min}$)	Sensitive itemsets (with supersets)	Solution by IGA			
		(%)	Time (sec)	(%)	(%)
		Accuracy	Total	Distance	Info. Loss
kosarak (4,950)	10 (18)	99.54	115	0.077	13.504
	20 (31)	99.05	146	0.15	25.297
	50 (65)	98.57	233	0.3	34.789
retail (88)	10 (10)	99.84	8	0.016	0.732
	20 (20)	99.58	9	0.045	2.128
	50 (65)	98.98	10	0.154	5.437
retail (44)	10 (15)	99.38	9	0.062	2.163
	20 (32)	98.8	9	0.126	4.874
	50 (97)	97.49	10	0.346	11.197
mushroom (1,625)	10 (2336)	93.4	1	0.7	63.312
	20 (2395)	88.57	1	1.527	73.401
	50 (5341)	90.65	8	1.184	76.101

B)

DB name ($б_{min}$)	Sensitive itemsets (with supersets)	Solution by Menon et al.					
		(%)	Time (sec)			(%)	(%)
		Accuracy	IP	H	Total	Distance	Info. Loss
kosarak (4,950)	10 (18)	99.59	11.5	129	140.5	0.069	6.787
	20 (31)	99.23	27.7	126	153.7	0.133	12.718
	50 (65)	98.95	35,976.1	115	36,091	0.221	22.19
retail (88)	10 (10)	99.83	0	7	7	0.017	0.183
	20 (20)	99.6	0.1	8	8.1	0.039	1.119
	50 (65)	99.05	0.1	8	8.1	0.109	1.813
retail (44)	10 (15)	99.37	0.1	7	7.1	0.062	0.556
	20 (32)	98.77	0.1	8	8.1	0.12	2.192
	50 (97)	97.46	0.2	8	8.2	0.265	4.363
mushroom (1,625)	10 (2336)	93.4	0.1	1	1.1	0.441	39.028
	20 (2395)	93.16	0.2	1	1.2	0.797	64.618
	50 (5341)	92.32	0.3	1	1.3	1.16	74.34

Figure 4.2. Results categorized in terms of type of approach used in experiments.

(cont. on next page)

C)

DB name (σ_{min})	Sensitive itemsets (with supersets)	Coefficient-Based Solution						
		(%)	Time (sec)				(%)	(%)
		Accuracy	C	IP	H	Total	Distance	Info. Loss
kosarak (4,950)	10 (18)	99.27	71	11.3	124	206.3	0.093	1.316
	20 (31)	98.5	141	67.7	116	324.7	0.19	3.983
	50 (65)	97	360	6,543	120	7,023	0.391	4.152
retail (88)	10 (10)	99.8	1	0.1	8	9.1	0.019	0.037
	20 (20)	99.42	1	0.1	8	9.1	0.056	0.183
	50 (65)	98.43	3	0.3	8	11.3	0.154	0.481
retail (44)	10 (15)	99.34	2	0.1	7	9.1	0.065	0.281
	20 (32)	98.54	4	0.1	8	12.1	0.142	1.282
	50 (97)	96.62	8	0.3	8	16.3	0.335	2.392
mushroom (1,625)	10 (2336)	93.4	89	0.1	1	90.1	0.441	38.247
	20 (2395)	84.94	114	0.5	1	115.5	0.762	52.382
	50 (5341)	79.47	141	0.8	1	142.8	1.52	64.626

Figure 4.2. Results categorized in terms of type of approach used in experiments.

Table 4.6. Comparison of Menon Approach and Coefficient-based Approach.

DB name (σ_{min})	Sensitive itemsets (with supersets)	Perc. (%)			
		Accuracy	Total Time	Distance	Info. Loss
kosarak (4,950)	10 (18)	-0.32%	-46.83%	-34.70%	80.61%
	20 (31)	-0.74%	-111.26%	-42.88%	68.68%
	50 (65)	-1.97%	80.54%	-76.64%	81.29%
retail (88)	10 (10)	-0.03%	-30.00%	-16.00%	80.00%
	20 (20)	-0.18%	-12.35%	-43.94%	83.61%
	50 (65)	-0.63%	-39.51%	-41.56%	73.47%
retail (44)	10 (15)	-0.03%	-28.17%	-4.83%	49.41%
	20 (32)	-0.23%	-49.38%	-18.53%	41.49%
	50 (97)	-0.86%	-98.78%	-26.24%	45.18%
mushroom (1,625)	10 (2336)	0.00%	-8090.91%	0.00%	2.00%
	20 (2395)	-8.82%	-9525.00%	4.43%	18.94%
	50 (5341)	-13.92%	-10884.62%	-30.54%	13.07%

CHAPTER 5

CONCLUSION

The serious privacy concerns that are raised due to the sharing of large transactional databases with untrusted third parties for association rule mining purposes, soon brought into existence the area of association rule hiding, a very popular subarea of privacy preserving data mining. Association rule hiding focuses on the privacy implications originating from the application of association rule mining to shared databases and aims to provide sophisticated techniques that effectively block access to sensitive association rules that would otherwise be revealed when mining the data. The research in this area has progressed mainly along three principal directions: (i) heuristic-based approaches, (ii) border-based approaches, and (iii) exact hiding approaches. The technique of approach that presented in the thesis consists in exact hiding approaches, since they comprise the most recent direction, offering increased quality guarantees in terms of side-effects and minimal alterations on a disclosed database introduced by the hiding process.

In this study, an efficient exact approach is presented to minimize side-effects in itemset hiding problem. The degree of side-effect is represented with coefficients that are placed into the objective function of integer programming. Experiments with real datasets show that our approach minimizes the number of concealed nonsensitive association rules successfully.

Association rule hiding is still an active research area, since there are several open problems that need to be addressed as well as a lot of room for improvement of the the hiding methodology. Although studies to date have presented various proposals in the area of exact frequent itemsets hiding, there is not a unifying approach to provide the best solution in all circumstances. The emergence of sophisticated exact hiding approaches of high complexity, especially for very large databases, causes the consideration of efficient parallel approaches to be employed for improving the runtime of these algorithms. Parallel approaches allow for decomposition of the constraints satisfaction problem into numerous components that can be solved independently. The overall solution is then attained as a function of the objectives of the individual

solutions. Although a framework for decomposition and parallelization of exact hiding approaches has been recently proposed in [25], there is not any solution to meet all requirements of each database in different circumstances.

In addition, Coefficient Computation Algorithm can be specialized based on the area where the published database is utilized. In this sense, coefficients in the objective function of the integer programming may be used in a more efficient way. Moreover, different optimization techniques can be achieved by exploiting the inherent characteristics of the constraints and objective function that are involved in the integer programming formulation, in a more advanced way.

REFERENCES

[1] Li, T., Li, N.: “On the Tradeoff between Privacy and Utility in Data Publishing”, Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 517-525, 2009.

[2] Dasseni, E., Verykios, V. S., Elmagarmid, A. K., Bertino, E.: “Hiding Association Rules by Using Confidence and Support”, Proceedings of the 4th International Workshop on Information Hiding, pp. 369–383, 2001.

[3] Sun, X., Yu, P. S.: “A Border-Based Approach for Hiding Sensitive Frequent Itemsets”, Proceedings of the Fifth IEEE International Conference on Data Mining (ICDM ’05), pp. 426–433, 2005.

[4] Sun, X., Yu, P. S.: “Hiding Sensitive Frequent Itemsets by a Border–Based Approach”, Computing Science and Engineering, Volume 1, Issue 1, pp. 74–94, September 2007.

[5] Gkoulalas-Divanis, A., Verykios, V. S.: “An Integer Programming Approach for Frequent Itemset Hiding.”, Proceedings of the ACM Conference on Information and Knowledge Management (CIKM ’06), pp. 748-757, November 2006.

[6] Gkoulalas-Divanis, A., Verykios, V. S.: “Hiding sensitive knowledge without side effects”, Knowledge and Information Systems, Volume 20, Issue 3, pp. 263-299, August 2009.

[7] Gkoulalas-Divanis, A., Verykios, V. S.: “Exact Knowledge Hiding through Database Extension”, IEEE Transactions on Knowledge and Data Engineering, Volume 21, Issue 5, pp. 699-713, May 2009.

[8] Menon, S., Sarkar, S., Mukherjee, S.: “Maximizing Accuracy of Shared Databases when Concealing Sensitive Patterns”, Information Systems Research, Volume 16, Issue 3, pp. 256–270, September 2005.

[9] Yeh, J. S., Hsu, P. C.: “HHUIF and MSICF: Novel Algorithms for Privacy Preserving Utility Mining”, Expert Systems with Applications, Volume 37, Issue 7, pp. 4779-4786, 2010.

[10] Han, J., Kamber M., Pei, J.: “Data Mining, Concepts and Techniques”, Amsterdam: Elsevier/Morgan Kaufmann, Internet resource, 2012.

[11] Agrawal, R., Srikant, R.: “Fast Algorithms for Mining Association Rules”, Proceedings of the 20th VLDB Conference Santiago, Chile.

[12] Hua, M., Pei, J.: “A Survey of Utility-based Privacy-Preserving Data Transformation Methods”, Privacy-Preserving Data Mining Advances in Database Systems, Volume 34, pp. 207-237, 2008.

[13] Bayardo, R. J., Agrawal R.: “Data Privacy Through Optimal K-Anonymization”, Proceedings of the 21st International Conference on Data Engineering (ICDE’05), pp. 217 – 228, 2005.

[14] Loukides, G., Shao, J.: “Data Utility and Privacy Protection Trade-Off in K-Anonymisation” ACM International Conference Proceeding Series, Volume 331, pp. 36-45, 2008.

[15] Brickell, J., Shmatikov, V.: “The Cost of Privacy: Destruction of Data-Mining Utility in Anonymized Data Publishing” Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 70-78, 2008.

[16] Agrawal, R., Srikant, R.: “Privacy Preserving Data Mining”, ACM SIGMOD Record Homepage, Volume 29, Issue 2, pp. 439-450, June 2000.

[17] Lindell, Y., Pinkas, B.: “Privacy Preserving Data Mining”, Journal of Cryptology, Volume 15, Issue 3, pp. 36–54, 2002.

[18] Gkoulalas-Divanis, A., Verykios, V. S.: “Association Rule Hiding for Data Mining”, New York: Springer, Internet resource, 2010.

[19] Verykios, V., Elmagarmid, A., Bertino, E., Saygin, Y., Dasseni, E.: “Association Rule Hiding”. IEEE Transactions on Knowledge and Data Engineering, Volume 16, Issue 4, pp. 434-447, April 2004.

[20] Atallah, M., Elmagarmid, A., Ibrahim, M., Bertino, E.,Verykios, V.: “Disclosure Limitation of Sensitive Rules”, Proceedings of the 1999 Workshop on Knowledge and Data Engineering Exchange (KDEX ’99), pp. 45-52, 1999.

[21] Oliveira, S. R. M., Zaïane, O. R.: “Privacy Preserving Frequent Itemset Mining”, Proceedings of the IEEE ICDM Workshop Privacy, Security Data Mining, pp. 43–54, Australian Computer Society, 2002.

[22] Amiri, A.: “Dare to Share: Protecting Sensitive Knowledge with Data Sanitization“, Decision Support Systems, Volume 43, Issue 1, pp. 181-191, 2007.

[23] Mannila, H., Toivonen, H.: “Levelwise Search and Borders of Theories in Knowledge Discovery”. Data Mining and Knowledge Discovery, Volume 1, Issue 3, pp. 241–258, 1997.

[24] Reddy, M., Wang, R. Y.: “Estimating Data Accuracy in a Federated Database Environment”, Proceedings of 6th International Conference on Information Systems and Management of Data (CISMOD), pp. 115–134, 1995.

[25] Gkoulalas-Divanis, A., Verykios, V. S.: “A Parallelization Framework for Exact Knowledge Hiding in Transactional Databases”, Proceedings of The IFIP TC-11 23rd International Information Security Conference, IFIP 20th World Computer Congress, IFIP SEC 2008, Volume 278, pp. 349-363, Springer, September 2008.

[26] GLPK, GNU GLPK 4.32 User’s Manual, Free Software Foundation Inc, Boston, MA, Internet resource, 2008.

[27] Bodon, F.: “A fast APRIORI implementation”, Proceedings of Workshop Frequent Itemset Mining Implementations (FIMI’03), Volume 90, CEURWS.org, CEURWorkshop Proceedings, 2003.

[28] Brijs, T., Swinnen, G., Vanhoof, K., Wets, G.:. “Using Association Rules for Product Assortment Decisions: A Case Study”, Proceeding of the 5th ACM SIGKDD Internat. Conf. Knowledge Discovery Data Mining, ACM Press, 1999.

[29] Bayardo, R.: “Efficiently Mining Long Patterns from Databases”, Proceedings of the ACM SIGMOD, 1998.

[30] Jünger, M., Liebling, T. M., Naddef, D., Nemhauser, G. L., Pulleyblank, W. R., Reinelt, G., Rinaldi, G., Wolsey, L. A.: “50 Years of Integer Programming 1958-2008 From the Early Years to the State-of-the-Art”, Springer, 2010.