DYNAMIC ITEMSET HIDING UNDER MULTIPLE SUPPORT THRESHOLDS

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

DOCTOR OF PHILOSOPHY

in Computer Engineering

by
Ahmet Cumhur ÖZTÜRK

July 2018
İZMİR

We approve the thesis of **Ahmet Cumhur ÖZTÜRK**

Examining Committee Members:

Assoc. Prof. Dr. Belgin ERGENÇ BOSTANOĞLU
Computer Engineering, Izmir Institute of Technology

Assoc. Prof. Dr. Adil ALPKOÇAK
Computer Engineering, Dokuz Eylül University

Assoc. Prof. Dr. Orhan DAĞDEVİREN
International Computer Institute, Ege University

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

Dr. Serap ŞAHİN
Computer Engineering, Izmir Institute of Technology

2 July 2018

Assoc. Prof. Dr. Belgin ERGENÇ BOSTANOĞLU
Supervisor, Computer Engineering, Izmir Institute of Technology

Assoc. Prof. Dr. Murat ERTEN
Head of the Department of Computer Engineering

Prof. Dr. Aysun SOFUOĞLU
Dean of the Graduate School of Engineering and Science

ACKNOWLEDGEMENTS

Firstly, I would like to thank my advisor Assoc. Prof. Belgin ERGENÇ BOSTANOĞLU and express my sincere gratitude to her for giving me the opportunity to work with her and for her endless support. I have learned many things from her.

My gratitude and appreciation to my examiners Assoc. Prof. Adil ALPKOÇAK and Dr. Serap ŞAHİN for their time and effort spent on my thesis.

I would also like to thank The Scientific and Technological Research Council of Turkey (TÜBİTAK) for supporting my thesis, under ARDEB 3501 Project No: 114E779.

I would like to thank all my family, starting with my mother Safiye ÖZTÜRK, my father Mustafa ÖZTÜRK, my sister Çiçek ÖZTÜRK and my brother Murat ÖZTÜRK for supporting me throughout my whole life and encouraging me. I am very lucky to have a family like them.

I especially thank my wife and love, Gözde ÖZTÜRK for her patience, encouragement and support. Writing this thesis would be very difficult without her.

Finally, I would like to thank to my mother in love Gülten TIĞA and father in love Osman TIĞA for their supports.

ABSTRACT

DYNAMIC ITEMSET HIDING UNDER MULTIPLE SUPPORT THRESOLDS

Data sharing is commonly performed between organizations for mutual benefits. However, if confidential knowledge is not hidden before the data is published it may pose threat to security and privacy. The privacy preserving frequent itemset mining is the process of hiding sensitive itemsets from being discovered with any frequent itemset mining algorithm. The privacy constraint of sensitive itemset hiding is sensitive threshold. If support of a given sensitive itemset is under the sensitive threshold, then this sensitive itemset is considered as non-interesting and hidden. One possible way of decreasing support of sensitive itemsets under predefined sensitive threshold is deleting items from a set of transaction. This type of frequent itemset sanitization is called distortion based frequent itemset hiding.

The main focus of this thesis is to preserve sensitive itemsets with considering the multiple sensitive thresholds on both static and dynamic environments. Three different distortion based frequent itemset hiding algorithms proposed; Pseodo Graph Based Sanitization (PGBS), Itemset Oriented Pseudo Graph Based Sanitization (IPGBS) and DynamicPGBS are proposed. Both PGBS and IPGBS algorithms are designed for static environment and the DynamicPGBS algorithm is designed for the dynamic environment. The main objective of these three algorithms is to hide all sensitive itemsets with giving minimum distortion on non-sensitive knowledge and data in the resulting sanitized database.

.

ÖZET

ÇOKLU DESTEK EŞİKLERİNDE DİNAMİK SIK KÜMELER GİZLEMESİ

Veri paylaşımı, ortak yararlar için kuruluşlar arasında yaygın olarak yapılmaktadır. Ancak, gizli bilgi, veriler yayınlanmadan önce gizlenmez ise güvenlik ve gizlilik için tehdit oluşturabilir. Gizliliği koruyan sık kümeler madenciliği hassas kümelerin herhangi bir sık küme madencilik algoritması ile ortaya çıkarılmasını önleme işlemidir. Sık kümelerin gizlenmesindeki kısıtlama hassas eşiktir. Belirli bir hassas kümenin desteği hassas eşiğin altında ise bu hassas küme ilgi çekmez ve gizli olarak kabul edilir. Önceden tanımlanmış hassas eşik altındaki hassas kümelerin desteğini azaltmanın olası bir yolu, bir dizi kayıttan öğeleri silmektir. Bu tür temizleme işlemi bozma esaslı sık küme gizlemesi olarak adlandırılır.

Bu tezin ana odak noktası, hassas kümeleri hem statik hem de dinamik ortamlarda çoklu hassas destek eşiklerini dikkate alarak korumaktır. Üç farklı bozma esaslı sık küme gizleme algoritması; Pseodo Graph Based Sanitization (PGBS), Itemset Oriented Pseudo Graph Based Sanitization (IPGBS) ve DynamicPGBS önerilmiştir. Hem PGBS hem de IPGBS algoritmaları statik ortam için tasarlanmıştır ve DynamicPGBS algoritması dinamik ortam için tasarlanmıştır. Bu üç algoritmanın temel amacı, temizlenmiş veri tabanında tüm hassas kümelerin saklanması, hassas olmayan bilgi ve verilerde ise en az bozulma oluşturmaktır.

TABLE OF CONTENTS

LIST OF FIGURES

Table **Page**

LIST OF TABLES

Table **Page**

CHAPTER 1

INTRODUCTION

With the rapid developments in storage devices and CPU technologies every day more and more transactional data is being kept by companies and organizations. This huge amount of data mostly contains unforeseen information beneficial for the data owner. The data mining is the process of extracting such information from the data with the help of artificial intelligence, machine learning, statistics and database systems. One of the tasks of data mining is association rule mining. The association rule mining task identifies frequent patterns in the transactional data in the form of dependencies among attributes. It was first proposed by [42] and used for analyzing market basket data. The association rule mining mainly has two steps where the first is called frequent itemset generation and the second is called association rule generation. The frequent itemsets are the set of items that are occurring together above a predefined support threshold and the association rules are the meaningful rules that can generated from the set of frequent itemsets. The second step of association rule mining is straightforward; it consists of uncovering all possible combinations of items of a given frequent itemset [56]. Besides the frequent itemset mining is a much more sophisticated task compared to association rule generation and as a result association rule mining algorithms focus on the first step.

Many organizations implement itemset mining to their transactional data for short or long term planning and strategically decision making. They also share data with each other or with third parties for their mutual benefit. However, if the data being shared contains private or strategically important information this may pose threat to security and privacy of the organization. The protection of sensitive knowledge from being discovered by any data mining technique is called privacy preserving data mining (PPDM). Privacy preserving frequent itemset mining (PPFIM) is a subtask of PPDM Similarly the protection of frequent itemsets from being discovered by any frequent itemset mining technique is called privacy preserving frequent itemsets mining.

1.1. The Problem Statement

In this thesis the privacy preserving frequent itemset mining is investigated in two different database environments; dynamic and static. The following assumptions are made in the problem of protecting sensitive frequent itemsets. First the database owner has to know the sensitive itemsets in advance and the second the database owner has to define the sensitive threshold of each sensitive itemset before the sanitization process where the sensitive threshold is the support value for considering each sensitive itemset as being restricted.

The protection of sensitive frequent itemsets in a given transactional database D can be handled by changing sufficient amount of transactions till every sensitive itemset becomes uninteresting in the modified database D'. But this brings out the side effects as loss of non-sensitive frequent itemsets and the dissimilarity between original database D and D'. The most challenging problem of frequent itemset hiding is protecting the non-sensitive knowledge while hiding all given predefined sensitive itemsets.

Based on the environment they are designed for, the privacy preserving frequent itemset mining algorithms can be divided into two where the first is PPFIM algorithms for static environment and the second is PPFIM algorithms for dynamic environment. In static environment the state of the itemsets never change, on the other hand in dynamic environment the state of itemsets continuously change with the arriving batch of transactions. After a batch of transaction arrives to the database the state of a frequent itemset may become infrequent or remain frequent also the state of an infrequent itemset may become frequent or remain infrequent. This state imbalance of itemsets should be considered while designing the frequent itemset hiding algorithm in dynamic environment. Although any static environment frequent itemset hiding algorithm can be adapted to dynamic environment with waiting all increments to arrive and then perform the hiding process on the whole updated database, this will be inefficient. The inefficiency is due to running the hiding process from beginning whenever a database publish process is needed.

In the literature most of the frequent itemset hiding algorithms are designed with considering a single sensitive threshold for all sensitive itemset but this does not reflect the reality. Because as stated in [46] some itemsets appear frequently while others appear rarely and unique support threshold for all itemsets does not reflect their importance. For

illustration in a super market the purchase amount of cooking pan and oven is most probably smaller than purchase amount of coffee and milk. The sales of cooking pan and oven because the profit obtained from them may carry more importance. Two problems arise if a unique sensitive threshold is defined for every sensitive itemset in the database; the first is this gives limitation to the database owner and the second is support of high frequent sensitive itemset is going to be decreased more if low frequent itemsets needed to be hidden. This will increase the loss in non-sensitive knowledge if items in the given transactional database are highly correlated with each other.

1.2. Contributions of the Thesis

The aim of this thesis is to give a contribution to the field of privacy preserving frequent itemset mining (PPFIM). First existing PPFIM techniques in the literature are surveyed and they are categorized according to hiding methodology they propose. Next three different frequent itemset hiding algorithms are proposed. The first algorithm is Pseudo Graph Based Sanitization (PGBS), the second algorithm is Itemset Oriented Pseudo Graph (IPGBS) and the third algorithm is Dynamic Pseudo Graph based Sanitization (DynamicPGBS). The PGBS and the IPGBS algorithms are developed for static database environment and the DynamicPGBS is proposed for dynamic database environment. Unlike most of the existing frequent itemset hiding solutions, these three algorithms allow database owner to define different sensitive threshold for each sensitive itemset. The objective of these proposed itemset hiding algorithms is to hide all sensitive itemsets with minimizing the side effects such as execution time, loss of non-sensitive knowledge and amount of modification made on transactions. The PGBS, IPGBS and DynamicPGBS algorithms employ different graph based internal data structures for increasing the efficiency of the sanitization process.

The contributions of this research are listed as follows:

- Pseudo Graph Based Sanitization (PGBS) Algorithm [14]: This algorithm uses the Pseudo Graph (PG) data structure to represent each different transaction of the transactional database D. It directly converts all transactions of D into PG without filtering or eliminating any transaction, provides advantage on execution time but compromise disadvantage on memory consumption. The main methodology behind the PGBS algorithm is to group sensitive itemsets

sharing common item and then modifying transactions containing each different sensitive itemset group. This make possible to reduce support of more than one sensitive itemset simultaneously with single item removal by deleting the item that is common in the uncovered transactions.

- Itemset Oriented Pseudo Graph Based Sanitization (IPGBS) Algorithm: This algorithm uses the Itemset Oriented Pseudo Graph (IPG) data structure to represent each different sensitive itemset of the transactional database D. The IPGBS algorithm uncovers each sensitive itemset from D and represents relation between transactions and sensitive itemsets with IPG. Representing only sensitive itemsets with the graph based data structure provides advantage on memory requirement but compromise is more execution time for creating the graph based data structure. This is because every sensitive itemset in each transaction is filtered out before they are converted into the IPG. The main methodology behind IPGBS algorithm is to modify transactions containing maximum number of non-sanitized sensitive itemsets. As in PGBS algorithm the IPGBS decreases support of more than one sensitive itemset at once with removing item that is common in more than one sensitive itemset. Also the IPGBS tries to prevent decreasing support of already sanitized sensitive itemsets overmuch with modifying transactions only containing sanitizing sensitive itemsets.
- Dynamic Pseudo Graph Based Sanitization (DynamicPGBS) Algorithm [36]: This algorithm is designed for the dynamic environment and uses the Transaction Oriented Pseudo Graph (TPG) data structure for representing each sensitive transaction. The only difference between the PG and TPG is that PG keeps all transactions while TPG keeps only sensitive transactions of a given transaction database. Representing only sensitive transactions with the graph based data structure provides advantage on memory requirement but compromise is more execution time as in IPGBS algorithm. The main methodology behind DynamicPGBS algorithm is to modify transactions containing maximum number of sensitive itemsets. As in PGBS algorithm the IPGBS decreases support of more than one sensitive itemset at once with removing item that is common in more than one sensitive itemset.

- The performance of PGBS and IPGBS algorithms are evaluated together with one different similar counterpart. The results indicate that the PGBS algorithm is advantageous in terms of execution time and total number of item removals compared to IPGBS algorithms. On the other hand, the IPGBS algorithm is advantageous in terms of non-sensitive knowledge loss and data validity compared to PGBS algorithm. The performance of DynamicPGBS is evaluated with two different dynamic frequent itemset hiding algorithms. The first counterpart SPITF is similar to DynamicPGBS as both algorithms modify transactions in the whole updated database. On the other hand, the second counterpart RHID algorithm only modifies transactions in the incremental part of the database. The evaluation results indicate that the DynamicPGBS is capable of hiding all given sensitive itemsets while the SPITF algorithm fails to hide some of the sensitive itemsets. The DynamicPGBS is advantageous in terms of non-sensitive knowledge loss, execution time and data validity compared to SPITF [26] and RHID [24] algorithms.

1.3. Organization of the Thesis

In chapter 2 the fundamentals of knowledge discovery, association rule mining and frequent itemset mining are given. Then the process of protecting sensitive frequent itemsets in transactional database for both static and dynamic environment is described. In addition, the metrics for measuring the effectiveness of frequent itemset hiding algorithms is provided.

In chapter 3 the state of art in frequent itemset hiding research is reviewed. The main methodologies behind existing frequent itemset hiding algorithms in the literature are introduced and described.

In chapter 4 a set of frequent itemset hiding algorithms are introduced. Two of the hiding methods namely PGBS and IPGBS are designed for static database environment and one is namely DynamicPGBS is designed for dynamic environment.

In chapter 5 the algorithms that are proposed are validated by using a set of experiments. These experiments are for measuring the execution time, the information loss, the distance, the accuracy and the total memory allocation. Also the databases and the set of itemsets defined as sensitive which are adopted in the experiments are described.

In chapter 6 conclusion of the work is given with a brief summary and future research planning with the direction of this work is discussed.

CHAPTER 2

BASIC CONCEPTS

The process of discovering previously unknown knowledge from huge amount of data with the help of database systems, artificial intelligence, machine learning, and statistics is called data mining. Data mining is different from analyzing the data with query processing tools. Query processing tools enable editing, finding, reporting and summarizing the data while data mining enables extracting previously unknown knowledge. Although the terms knowledge discovery and data mining are commonly used interchangeably they are different. Knowledge discovery (KDD) refers to overall process of discovering useful knowledge from data, it includes data selection, data cleaning, data transformation, data mining, interpretation and evaluation [55]. Data mining refers to application of algorithms for extracting patterns from data. Data mining is the key step in KDD process. As can be seen from Figure 1, data mining is a sub step in knowledge discovery.

Figure 1. Steps in knowledge discovery.

The data often contains noisy and missing values so the data from multiple data sources should be cleaned before integration. The cleaning phase includes filling the missing values, removing outliers and maintaining consistency of different data types. Next the data from heterogeneous data sources are combined into compatible data source called integrated database. After all data are cleaned and integrated the data relevant to analysis is selected and transformed into appropriate form for the data mining [57]. The transformation phase includes operations such as normalization, generalization or aggregation. Then the data mining is applied to the transformed data for gathering previously unknown, useful, interesting and hidden patterns. After the patterns are

generated, the knowledge is gathered by filtering out patterns by some measures such as the interestingness threshold.

2.1. Data Mining Techniques

There are various data mining tasks for extracting different kinds of pattern from the database and these data mining tasks can be can be categorized into two; descriptive data mining tasks and predictive data mining tasks. The descriptive data mining tasks derive patterns that summarize the underlying relationships between data and describe the general properties of the existing data such as web pages that are accessed together. The predictive data mining tasks predict the value of a specific attribute based on the value of other attributes such as deciding whether a patient has specific disease based on the medical test results. The classification, regression, time series analysis are examples for the prediction tasks while clustering, summarization and association rules are some of the examples for descriptive tasks [47-49].

The outstanding tasks of data mining can be can be categorized as classification, clustering and association rule mining.

Classification: The task of mapping data into predefined classes is called the classification [50]. This task first creates a predictive learning function that distinguishes the data according to its features and then it assigns the class of a newly presented data to its corresponding class with the help of this function. The classification of data consists of training phase and labelling phase. The training phase creates the predictive learning function (mapping function) with using the training set where the training set consists of previously labeled data. Then with using mapping function the label of the new data can estimated.

Figure 2 illustrates a classification algorithm used for predicting whether a new arrival email is spam or not. The training data consists of categorized emails and a classification algorithm is applied to these data for generating rules to predict the status of a given email.

Clustering: The clustering task divides data objects in to groups called clusters where each cluster consists of objects with similar characteristics [48]. The clustering task is sometimes referred as unsupervised classification because as in classification method the

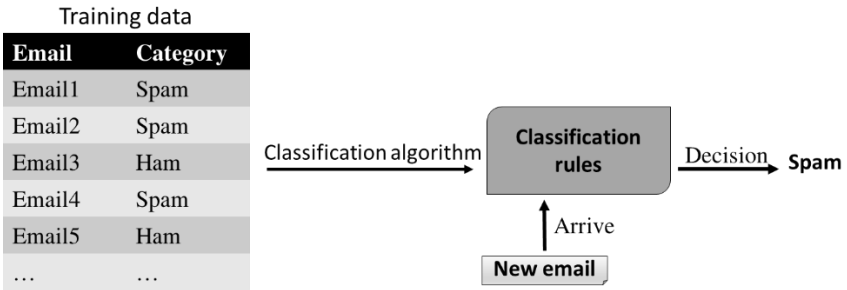


Figure 2. Classification algorithm for predicting the status of a received email.

clustering groups data into different groups however unlike classification the number of clusters are not predefined, they are derived from the given data. Cluster analysis is being used in many applications such as market segmentation, social network analysis, image processing medical segmentation and anomaly detection.

Figure 3 illustrates a clustering algorithm applied to two dimensions of data objects where x and y axis represents two different features and the points represents each data objects. After a clustering algorithm is applied three different groups is discovered according to the features of data objects.

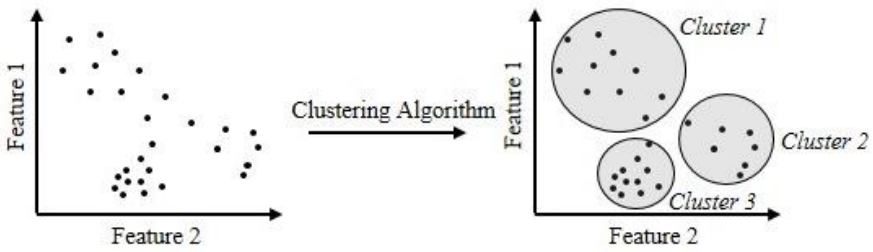


Figure 3. Clustering data objects according to their features.

Association rules: The task of uncovering interesting relations between items is called association rule mining. The association rule mining can be applied to transactional data and it was first used in analyzing shopping behaviors of customers [42]. Association rules are generated with two basic steps where the first step is frequent itemset (co-occurring itemsets) generation and the second step is meaningful rule generation from these discovered frequent itemsets. Association rules are considered interesting if they satisfy both minimum support and confidence thresholds. In a given associating rule a→b c, support is the percentage of transactions containing "a", "b" and "c", confidence is the percentage of transactions containing "a" also contains "b" and "c". The relations between

items is expressed with rules such as 80% of transactions containing item “a”, “b” and “c” at the same time and 60% of all transactions containing “a” also contains “b” and “c”.

2.2. Frequent Itemset Mining

Let I={i_1,…i_n} be a set of items, an itemset X is a non-empty subset of I. A transaction is a an ordered pair of items denoted as <TID,X> where TID is the unique identifier. The set of all TIDs may denote the set of all customers visiting the e-commerce web site, all customers making phone calls of a GSM company, all credit card users of a bank. A transactional database is a set of transactions. The set of items I may denote the customer purchases from an e-commerce web site, log files of telephone calls carried out by a specific GSM company, credit card purchase information of customers and so on. The number of items located in an itemset is denoted as k and and the size of the itemset is denoted as k-itemset. There are $2^k - 1$ number of proper subset of a k-itemset. Figure 4 illustrates the itemset lattice for the items I = {a,b,c,d}. In this figure there are $2^4 - 1 = 15$ number of possible itemsets that can be generated from the set I [53].

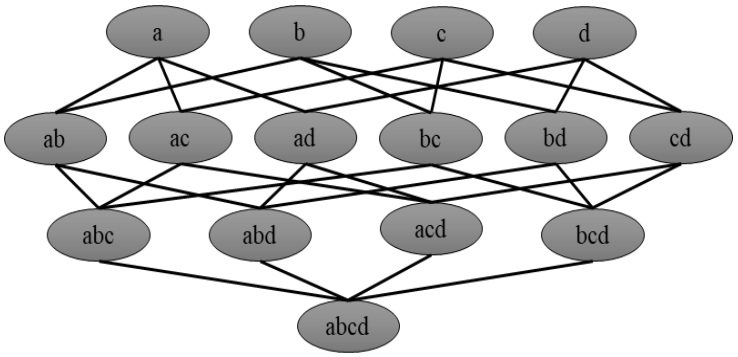


Figure 4. Itemset lattice of four different items.

A transactional database is a set of transactions and it consist of tuples where each tuple has a unique identifier and corresponding transaction. It is possible to represent a transactional database as table. A binary transactional database represents the relation between TIDs and items, a given transaction with transaction id n contains an itemset X={x_1,x_2,x_3,…,x_k} if {for all i=1 to k| x_i=1}. Figure 5 (a) shows an example of transactional database and Figure 5 (b) shows its binary representation. In the binary

database the set of all items are I={a,b,c,d,e,f,g} and the set of transaction ids are tids={1,2,3,4,5,6,7,8,9,10}.

TID	Transactions
1	cf
2	abe
3	de
4	afg
5	ade
6	ae
7	acd
8	abcde
9	adfg
10	acd

(a) Transactional database

TID	a	b	c	d	e	f	g
1	0	0	1	0	0	1	0
2	1	1	0	0	1	0	0
3	0	0	0	1	1	0	0
4	1	0	0	0	0	1	1
5	1	0	0	1	1	0	0
6	1	0	0	0	1	0	0
7	1	0	1	1	0	0	0
8	1	1	1	1	1	0	0
9	1	0	0	0	0	1	1
10	1	0	1	1	0	0	0

(b) Binary database

Figure 5. Sample databases.

The support count of an itemset X is the number of transactions containing X in D and it is denoted as scount(X). The fraction of transactions containing X in D is called support of X and it is denoted by supp(X). The support is an estimation of joint probability of items generating X. The support of an itemset X is calculated by equation (2.1) where |D| is the total number of transactions in D.

$$supp(X) = \frac{scount(X)}{|D|} \tag{2.1}$$

An itemset X is frequent if supp(X) ≥ σ, where σ is the user specified minimum support threshold and the set of all frequent itemsets in D is denoted as FI. For illustration suppose the database in Figure 5 (a) is given and let σ =10%, then all frequent itemsets generated from this database is shown in Table 1, where itemset column gives the itemsets and the support column shows the support of the corresponding itemset.

2.3. Association Rule Mining

Association rules represent affinities among itemsets. An association rule is represented as Y→Z where X and Y are different itemsets in transactional database D and Y∩Z=∅. An association rule relies on the support and confidence measures [42].

Table 1. Frequent itemsets when σ =20%.

itemset	support	itemset	support	itemset	support
A	0.2	fg	0.2	de	0.2
C	0.4	ab	0.2	ad	0.5
D	0.6	be	0.2	ae	0.4
F	0.3	af	0.2	afg	0.2
G	0.2	ac	0.3	abe	0.2
E	0.5	cd	0.3	acd	0.3
B	0.2	de	0.3	ade	0.2

The support is the measure of frequency of a rule in the given database D, the confidence is the measure of strength between itemsets in the given rule. The confidence of a rule Y→Z is calculated with the following equation [58]:

$$confidence(YZ) = \frac{supp(YZ)}{supp(Y)} \tag{2.2}$$

Given a database D and user defined minimum support and confidence thresholds, the association rules are generated with following steps;

Step1: Enumerate all itemsets that have support greater than the minimum support threshold.

Step2: Generate all rules having confidence greater than minimum confidence threshold with using the frequent itemsets generated at step 1.

Table 2 illustrates strong rules generated from the database given in Figure 5 (a) with 0.3 minimum support and 0.6 minimum confidence thresholds.

Table 2. Association rules with minimum support=0.3 and minimum confidence=0.6.

Association rule	Support	Confidence
d→a	0.5	0.83
a→d	0.5	0.625
e→a	0.4	0.8

2.4. Privacy Preserving Frequent Itemset Mining

In modern business, organizations make their database public or share it with other organizations or third parties for providing extraction of knowledge. However frequent itemset mining may lead to malicious usage of itemsets and pose security threat

on strategic or private information of data owners when the database is shared without any precautions. These set of itemsets are called sensitive frequent itemsets and they are defined as follows:

Sensitive Itemset: Let IL be a set of all itemsets in the itemset lattice of transactional database D, and the SI be a set of itemsets that should be hidden and defined by owner of the database where SI $\subset$ IL. A set of itemsets that are able to infer any itemset in SI are called sensitive itemsets.

The privacy preserving frequent itemset mining is the problem of concealing sensitive itemsets from the shared or published database from being discovered with any frequent itemset mining algorithm. This problem can achieved by generating a new database D' from the original database D in such a way that when frequent itemset mining is applied to the D' none of the sensitive itemsets are revealed [59]. Two problems arise when the database is sanitized in such a way. The first one is protection of sensitive knowledge and the solution is to prevent sensitive itemsets from being discovered with any frequent itemset mining algorithm. The second problem is protecting non-sensitive knowledge and the solution is to find optimum hiding solution that distorts minimum amount of non-sensitive frequent itemsets.

One possible way to hide sensitive itemsets from database D is to decrease their supports till the sensitive itemsets become infrequent. The support threshold used for hiding a given sensitive itemset X is the sensitive support threshold and it is defined by the database owner. This process of modifying the transactions to the point where no sensitive itemset can be discovered is called the sanitization process [9]. Decreasing the support of sensitive itemsets can be achieved by deleting items called victim items (selected for deletion) from a sufficient amount of transactions. The set of transactions that contain at least one sensitive itemset is called sensitive transaction and is defined as follows:

Sensitive Transaction: If a transaction supports any itemset in SI then it is called sensitive transaction. Formally if T is a set of all transactions in database D and SI is the set of all sensitive itemsets, for any si$\in$SI if si $\subseteq$ t where t $\in$ T, t is called sensitive transaction.

The objective of itemset hiding is to produce a sanitized database D' from original database D. It is mainly done by reducing the supports of sensitive itemsets under their predefined sensitive thresholds. The sensitive threshold is defined as follows:

Sensitive Threshold: The support threshold used for considering a sensitive itemset as non-interesting or hidden is called sensitive threshold and sensitive threshold of an itemset Y is denoted as st (Y). Formally if SI is the set of sensitive itemsets in D and ST is the set of all sensitive thresholds of these sensitive itemsets, then for all si $\in$ SI iff supp(si) < st(si) then all sensitive itemsets in D are hidden.

One of the techniques for reducing the support of a sensitive itemset is deleting items called victims from sensitive transactions. This technique is called distortion based frequent itemset hiding. Identifying the transactions that will be modified and deciding the items that will be removed from these transactions are two main challenges of distortion based sanitization technique.

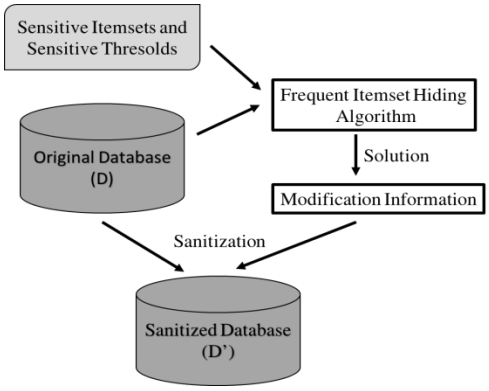


Figure 6. Distortion based frequent itemset hiding process.

The process of distortion based sensitive itemset hiding is illustrated in Figure 6. First the user defines the sanitization information which consist of sensitive itemsets and their sensitive threshold next the original database D and the sanitization information is given as input to the frequent itemsets hiding algorithm. The frequent itemsets hiding algorithm finds out the modification information which is the solution of the sanitization problem. Then with using the modification information the original database D is updated and the sanitized database D' is generated. Each different sensitive transaction and victim set for solving the sanitization problem has different side effects to the database. These side effects are defined as follows:

Hiding Failure (HF): The amount of sensitive itemsets failed to be hidden is represented with the hiding failure metric. It is calculated as follows;

$$HF = \frac{|SI|}{|SI'|} \tag{2.3}$$

where |SI'| is the number of sensitive itemsets in the sanitized database D' and |SI| is the number of sensitive itemsets in the original database.

Information Loss (IL): The amount of non-sensitive knowledge unintentionally removed from the original database during the sanitization process is represented with the Information Loss metric. It is calculated as follows;

$$IL = \frac{(|FI| - |SI|) - (|FI'| - |SI'|)}{|FI'| - |SI'|} \quad (2.4)$$

where |FI| is the number of frequent itemsets in the original database D and |FI'| is the number of frequent itemsets in the sanitized database.

Distance: The total number of items removed during the sanitization process is represented by the Distance metric and it is calculated as follows;

$$Distance = |I| - |I'| \quad (2.5)$$

where |I| is the total number of items in the original database and |I'| is the total number of items in the sanitized database.

Accuracy Loss: The total number of transactions modified during the sanitization process is represented with the Accuracy Loss metric and it is calculated as follows;

$$Accuracy\ Los = (|D - total\ number\ of\ transactions\ modified\ in\ D)/|D| \quad (2.6)$$

where D is the original database and |D| is the total number of transactions in D.

Table 3. Sensitive itemsets and their sensitive thresholds.

Sensitive Itemset	Sensitive Threshold
ab	0.3
cd	0.15
bd	0.1

For illustrating the given definitions and distortion based frequent itemset hiding suppose the transactional database is given in Figure 5 (a) and sensitive itemsets with their sensitive thresholds are given in Table 3. The transaction ids of sensitive transactions in D are 2,7,8 and 10. The support of "ad" is 0.5, "bd" is 0.1 and "cd" is 0.3 where support of each sensitive itemset is not smaller than their sensitive thresholds. One possible sanitization solution may remove the item "d" from transactions 8, 9 and 10 and as a result support of "ad", "bd" and "cd" reduce to 0.2,0.0,0.1 respectively which means all sensitive itemsets are hidden from the database D. The Hiding Failure in this sanitization

solution is zero because all sensitive itemsets are hidden; the Information Loss is calculated as ;((44 - 4) - (28 - 0)) / (28 - 0) = 0.4285 if the minimum support threshold for mining itemsets in D and D’ is set to 0.1. The Distance is 3 because at total 3 items are removed from D, the accuracy is 0.7, in different word database D and D’ are 70% similar to each other.

CHAPTER 3

RELATED WORK

In this chapter a literature review on existing frequent itemset and association rule hiding approaches are presented. These techniques are classified into two main categories depending on the environment they are designed for; dynamic environment and static environment as illustrated in Figure 7. In dynamic environment transactional database is being continuously updated by the arrival of increments whereas in static databases the state of transactions never changes. In the literature only a few approaches are proposed for dynamic environment. As can be seen in the figure the sanitization approaches for static database environment can classified as; border based, exact, reconstruction based and heuristic based. The border based approaches separate the itemset lattice with a border and revise this border to sanitize the given database. The exact approaches convert the sanitization problem into constraint satisfaction problem (CSP) and then apply integer programming to solve the CSP. Reconstruction based approaches first eliminate sensitive itemsets and their supersets from the set of frequent itemsets of the original database and then try to generate a sanitized database from non-sensitive itemsets. Heuristic based sanitization algorithms rely on different heuristics in victim item and victim transaction selection.

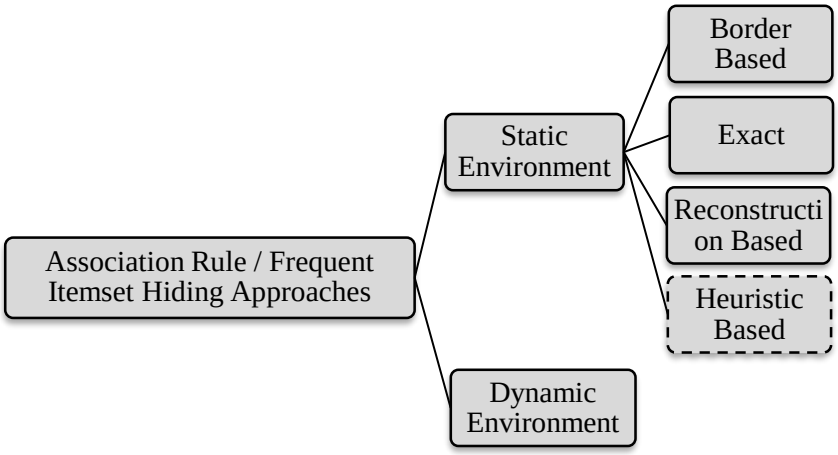


Figure 7. Rule/Itemset hiding approaches.

3.1. Sanitization Algorithms for Static Environment

In static environment the status of transactions never change more precisely it is assumed that there is not update and delete operations performed on the database. The main categorizations of rule or itemset hiding approaches on static environment can be done as border based approaches, exact approaches, reconstruction based approaches and heuristic based approaches as shown in Figure 7.

3.1.1. Border Based Sanitization Algorithms

Borders enable to separate frequent itemsets from non-frequent itemsets in the itemset lattice [1]. The border based hiding solution is to revise the border so that all sensitive itemsets and their supersets are separated from non-sensitive frequent itemsets. The positive border denoted as $Bd^{+}$and it consists of all frequent itemsets whose proposer supersets are non-frequent whereas the negative border is denoted as $Bd^{-}$and it consists of all non-frequent itemsets whose proper subsets are frequent. For illustration suppose the itemset lattice generated from a database is given in Figure 8. In this figure consider the frequent itemsets based on the predefined minimum support minsup are separated from non-frequent itemsets with the border as shown in straight line. The itemsets on the right hand side of the border are frequent and itemsets on the left side of the border are infrequent. Also suppose that sensitive itemsets are defined as “AB” and “AC”. The optimal revised border line moves minimal set of non-sensitive frequent itemsets on the right hand side of the border while moves all sensitive itemsets and their supersets on the right hand side of the revised border. The dashed line in Figure 8. shows the revised border after the sanitization process where all sensitive itemsets are left on the right hand side of the revised border.

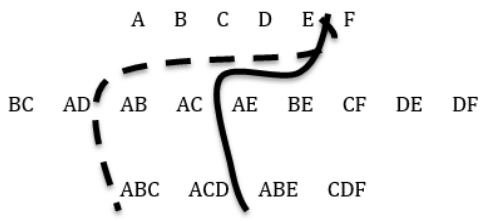


Figure 8. Itemset lattice for illustrating border revision.

The revision of the border for hiding sensitive frequent itemsets was first proposed by [2]. To improve the quality of the resulting sanitized database they focus on obtaining the quality of the border. The basic idea of their proposed algorithm BBA is to dynamically assigning weight to each itemset on the positive border, based on the support of itemsets. The algorithm calculates all possible sum of weights for each possible item deletion to hide a given itemset and then it chooses the item to delete among items that gives minimum impact on sum of weights.

The Max-Min1 and Max-Min2 [3] are two frequent itemset hiding approaches based on border revision. The main focus of these two algorithms is to modify the itemsets on the negative border while maintaining all itemsets on the revised positive border. For each item of a sensitive itemset the algorithm uncovers positive border itemsets containing the item and then among these itemsets the algorithms select the itemset that has the highest support in the database which is called max-min itemset. The algorithms try to modify the border in such a way that the support of the max-min itemset is minimally affected. The Max-Min1 and Max-Min2 algorithms employ the basic properties; they differ in selecting the victim item to delete from transactions. While the Max-Min1 algorithm selects the victim item randomly the Max-Min2 algorithm selects the victim item that gives minimum knowledge loss in case of deletion.

3.1.2. Exact Sanitization Algorithms

Exact frequent itemset hiding approaches use both border based and integer programming methodologies to find a hiding solution. These approaches formulate the frequent itemset hiding problem into constraint satisfaction problem (CSP) and then apply integer programming to solve the CSP. A CSP consists of a set of variables and a set of constraints [4]. Each variable has non-empty set of potential values and each constraint represents non empty set of possible combinations of variables. The CSP is solved by choosing value for each variable that satisfies the constraints. The integer programming (IP) and binary integer programming (BIP) are linear programming models where in IP all variables are integers and in BIP each variable can only take 0 or 1. For illustration Figure 9. shows a constraint matrix of a database. The columns t_1 to t_5 indicates the sensitive transactions and rows r_1 to r_4 indicates the sensitive itemsets and also the

constraint matrix shows if the given itemset is contained in a sensitive transaction e.g. the sensitive itemset r_1 is contained in only t_4. If the integer program aims minimizing the number of transaction modification, then it can be formulated as to minimize $x_1+x_2+x_3+x_4+x_5$ where the variable x_i is 1 if the ith transaction is modified and 0 otherwise.

	t_1	t_2	t_3	t_4	t_5
r_1	0	0	0	1	0
r_2	0	1	0	0	1
r_3	0	1	0	0	1
r_4	1	0	1	1	0

Figure 9. Constraint matrix with 4 different sensitive itemsets

The exact frequent itemset was first proposed in [5]. In this work minimum number of transactions for modification is identified with formulating the frequent itemset hiding problem into CSP and this CSP is solved with using linear programming. Then two heuristics namely the blanket approach and intelligent approach are employed for modifying these previously identified transactions. The blanket approach deletes all items except one from transactions whereas the intelligence approach deletes as small as possible number of items from transactions. The blanket approach hides more non-sensitive itemsets than intelligence approach because the intelligence approach tries to group sensitive itemsets and remove the item having maximum degree of conflict. The aim of the CSP they formulate is to find out minimum number of transaction modification for hiding all given sensitive itemsets. Also they make it possible to decompose the CSP into different parts for solving in a parallel manner to decrease the execution time.

Inline algorithm [6] does not rely on any heuristic during the sanitization operation. This algorithm formulates the hiding problem into CSP and solves it by using Binary Integer Programming (BIP). The CSP employed in the Inline algorithm tries to find the hiding solution that has the minimum distance metric.

Two Phase Iterative Algorithm [7] is an extension of the Inline algorithm. The Two Phase Iterative algorithm iterates till a solution of the given CSP is found and if not it iterates till a predefined number of iterations have taken place.

Hybrid [8] algorithm combines the border revision, CSP and BIP for hiding sensitive itemsets. Instead of modifying transactions in the original database the Hybrid

algorithm adds database extension to the database. This extension consists of a set of transactions where attaching these transactions into the database lowers the importance of sensitive itemsets while minimally effects the non-sensitive itemsets.

The full exact approach [37] uses integer programming for optimizing the information loss and distance side effects of the sanitization process. This algorithm is entirely exact because it does not employ any heuristics during the sanitization process. Constraints defined in the integer programming of this approach try to ensure that the frequency of sensitive itemsets are below support threshold while frequency of non-sensitive itemsets are still above the threshold.

3.1.3. Reconstruction Based Sanitization Algorithms

The main idea in reconstruction based sanitization approaches is to release a new database that is constructed from sanitized knowledge. Rather than sanitizing the actual database these approaches first sanitize the sensitive rules or itemsets and then create a new database from this sanitized knowledge. These approaches are called knowledge based because they start the sanitization from the knowledge. The computational complexity of reconstruction based approaches are first analyzed in [32] and showed that for most of the cases the problem is NP-Complete for finding a compatible dataset from frequent itemsets. Figure 10 illustrates the reconstruction based frequent itemset approach proposed in [30]. First all frequent itemsets of the original database D is generated then the set of sensitive frequent itemsets (FI') is removed from frequent itemsets (FI) and a new sanitized database (D') is created from sanitized frequent itemsets (FI').

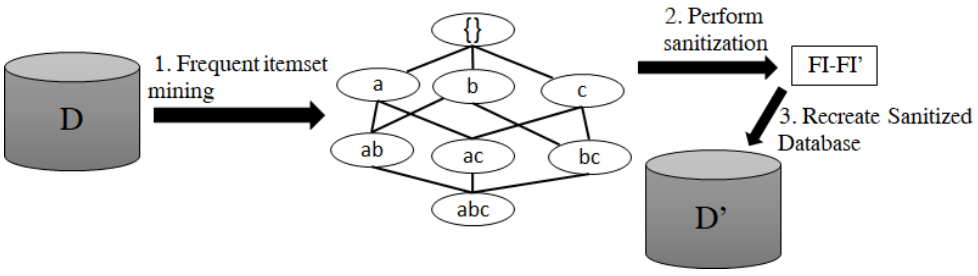


Figure 10. Inverse frequent itemset mining

The reconstruction based sanitization idea is first proposed in [30] called Constraint based Inverse Itemset Lattice Mining (CIILM). This algorithm is designed for hiding sensitive itemsets. The CIILM algorithm first sanitizes the itemset lattice and then recreates the database from the sanitized itemset lattice with performing inverse frequent itemset mining.

Fp-Tree based approach is presented in [31]. This approach has three phases where the first phase generates all frequent itemsets with their support from the database. The second phase performs sanitization operation over this generated set. The third phase creates a new database from sanitized frequent itemsets with using an FP-Tree based inverse frequent itemset mining algorithm.

Randomization of the transactions is proposed in [33]. This approach does not remove any item from the database instead it adds items to each transaction. In this approach first all possible frequent itemsets from the randomized transactions is generated with their support. Then the support of each possible frequent itemset is reconstructed to find out frequent itemsets.

3.1.4. Heuristic Based Sanitization Algorithms

A large amount of research has been conducted based on Heuristic sanitization approaches for static environment. These approaches perform the sanitization either with decreasing the support of sensitive itemsets or giving uncertainty to the support of sensitive itemsets. The heuristic based frequent itemset algorithms using the first technique are called distortion based frequent itemset hiding algorithms whereas the algorithms using the former technique are called blocking based frequent itemset hiding algorithms.

3.1.4.1. Blocking Based Heuristic Approaches

Blocking based sanitization approaches replaces some items with unknowns for hiding rules or itemsets. The aim of replacing unknowns is to give uncertainty to the support of sensitive itemsets or confidence of sensitive rules. Figure 11 illustrates a blocking based sanitization algorithm where Figure 11 (a) is the original database and

Figure 11 (b) is the sanitized database. For the itemset AC the support is 75% in the original database. The item A in the first transaction and the item C in the second transaction are replaced by unknown (?) in the sanitized database. As a result, the support of itemset AC becomes uncertain in the sanitized database and it can be expressed in the range between 0% and 50%.

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

(a) Original database

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

(b) Sanitized database

Figure 11. Blocking based sanitization.

Saygin et al. [23] proposed three blocking based sanitization approaches; CR, CR2 and GIH where CR and CR2 designed for hiding sensitive association rules with decreasing their confidence and GIH hides association rules by decreasing support of their generating itemsets. In this study a safety margin is used to define how much the MST of sensitive frequent itemsets or MCT of sensitive rules should be smaller to prevent recovery of hidden pattern by adversary. The CR and GIH algorithm choose transactions containing minimum number of items while the CR2 algorithm choose transactions containing maximum number of items on the left hand side of the sensitive rule. The GIH algorithm selects the items having maximum support as victim items and then replaces unknown instead of these victim items.

The ISL and DSR [41] algorithms are two blocking based sanitization algorithms designed for hiding informative association rules. The informative association rule set is the smallest set of association rules that makes the prediction of the entire association rule set of the database. The ISL algorithm increase support of the left hand side itemset of a given sensitive rule whereas the DSR algorithm decrease support of the right hand side itemset of a given sensitive rule. Both ISL and DSR algorithms choose the transactions for modification according to their size and they replace binary items with unknowns (?).

3.1.4.2. Distortion Based Heuristic Approaches

The distortion based sanitization approaches conceal sensitive itemsets from the database by decreasing support of them with modifying or completely removing sensitive transactions from the database. Figure 12shows an example of distortion based frequent itemset hiding approach for the database given in Figure 12 (a). In Figure 12 (b) some itemsets are concealed from the database by deleting some items from a set of sensitive transactions. Distortion based sanitization algorithms differ in techniques that they employ during selection of victim item and selection of transactions for modification. Existing distortion based heuristic algorithms select the victim item according to support, degree of conflict or by trial and error and they select the sensitive transactions according to number of sensitive or non-sensitive itemsets they contain, transaction length or by trial and error.

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

(a) Original database

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

(b) Sanitized database

Figure 12. Distortion based sanitization.

Determining the transaction for modification and item to be removed from them can be handled in an iterative way by trying each different possible solution and then choosing the optimal solution among these solutions. This type of sanitization algorithms use the trial and error methodology. Atallah et al. [9] first proposed the privacy preserving association rule hiding problem and they proposed an association rule hiding algorithm that conceals the sensitive association rules by concealing the sensitive itemsets generating these rules. An itemset graph is presented by the authors and with using this graph they iteratively uncover itemsets and transaction set for modification that gives least distortion to rest of itemsets. Then they remove predefined victim items from

sensitive transactions. The Aggregate [10] algorithm selects sensitive transactions whose removal gives impact on minimum number of non-sensitive itemsets and at the same time gives impact on maximum number of sensitive itemset, then it removes the selected transactions from database. Disaggregate [10] approach removes items from sensitive transactions whose removal effect least number of non-sensitive itemsets and at same time effect maximum number of sensitive itemsets. The Hybrid [10] approach chooses the sensitive transactions for modification with using Aggregate approach then modifies selected transactions with using the Disaggregate approach. The Aggregate algorithm greedy selects sensitive transactions whose removal gives impact on minimum number of non-sensitive itemsets and at the same time gives impact on maximum number of sensitive itemset, then removes the selected transactions from database. Disaggregate approach removes items from sensitive transactions whose removal effects least number of non-sensitive itemsets and at same time effects maximum number of sensitive itemsets. The Hybrid approach chooses the sensitive transactions for modification with using Aggregate approach then modifies selected transactions with using the Disaggregate approach.

Another type of sanitization methodology is grouping sensitive itemsets and then modifying transactions containing each different groups of sensitive itemset. This type of sanitization may conceal multiple sensitive itemsets at once. The sensitive itemsets can grouped according to the item common at each sensitive itemset. Oliveria et al. [11] proposed four sanitization algorithms among these algorithms the IGA algorithm introduced the multiple itemset hiding concept. Template Table Sanitization Algorithm (TTBS) [13] overcomes the overlapping groups problem faced in IGA with using a table called Template Table. The PGBS [14] algorithm employs a graph based data structure to speed up the hiding process. As in IGA this algorithm groups sensitive itemsets sharing common item and selects the victim item among sensitive itemset that has the maximum degree of conflict and then deletes victim item from transactions containing maximum number of sensitive itemsets.

The number of association rule or frequent itemset combinations may decrease with the size of the transaction where size is the number of items a transaction contains. As a result, the distortion given to the knowledge can be reduced by modifying small sized transactions. The Sliding Window Algorithm (SWA) [12] removes victim item from sensitive transactions with the shortest size. Verykios et al. [15] proposed five algorithms

based on two approaches, first approach prevents rules from being generated by hiding the frequent sets from which they are derived whereas the second approach reduces the importance of the rules by setting their confidence below a user-specified threshold. Algorithm 2b [15] hides sensitive itemsets by removing the item having maximum support from smallest length sensitive transactions and the Algorithm 2c [15] hides sensitive itemsets by removing sensitive itemsets from smallest length sensitive transactions.

Modifying transactions according to number of itemsets or rules they contain may decrease the distortion on non-sensitive knowledge. Maximum Item Conflict First (MICF) [16] algorithm first sorts sensitive transactions in increasing order according to how many sensitive itemset a sensitive transaction contains and deletes the victim item from sensitive transactions. The victim item is selected among items in a sensitive itemsets which has the maximum degree of conflict where the degree of conflict is the number of sensitive itemsets containing the victim item. The algorithm SIF-IDF[17] is inspired from TF-IDF(Term Frequency-Inverse Document Frequency). The SIF-IDF gives the relation degree of a transaction with sensitive itemsets. First the algorithm calculates each transactions SIF-IDF value, then sorts them in decreasing order of SIF-IDF values and starts modification from the transaction having maximum SIF-IDF value. Relevance Sorting [18] deletes the victim item among sensitive transactions containing less number of non-sensitive itemsets. The algorithm calculates the number of non-sensitive itemsets in each sensitive transaction and then assigns a priority value called relevance to each sensitive transaction and according to the relevance value it selects the transactions for modification.

Some algorithms in previous studies assign weight to each transaction and item to decide the solution for the modification. Priority-based Distortion Algorithm (PDA) [19] first uncovers all sensitive transactions then for each sensitive transaction it calculates the priority of each sensitive transaction by calculating how many non-sensitive rules will be affected for all possible item removal of a sensitive rule contained in a transaction. Weight-based Distortion Algorithm (WDA) [19] first assigns weights to sensitive rules where the weight is the measure how close the confidence of a sensitive rule to minimum confidence threshold value (MCT), the value of the weight is high if confidence of a sensitive rule is close to the MCT. Then using the weights WDA calculates each sensitive transactions priority. Because of the algorithm's high complexity, it is almost impossible

to use it on datasets having so many distinct items or databases having so many transactions. The Fast Hiding Sensitive Association Rules (FHSAR) [20] algorithm first assigns weight to each sensitive transaction and items in sensitive rules where the transaction weight is proportional to the number of sensitive rules a transaction contains and inverse proportional to the length of the transaction, also the weights assigned to items are proportional to the degree of conflict. Then the algorithm sorts the sensitive transactions in descending order of their weights and from sufficient amount of sensitive transactions deletes the items whose weight is maximum in the given sensitive transaction. The HSARWI [21] algorithm assigns weight to each sensitive transaction and items in sensitive rules as in FHSAR algorithm. The sensitive transactions are selected with the same procedures used in FHSAR algorithm; two algorithms differ in victim item selection. The HSARWI algorithm considers whether an item is on the right or left hand side of the sensitive rule and increases the weight of the item if it is on the right hand side. The MDSRRC [25] algorithm assigns weight to each transaction and item in a given sensitive rule where the weight of an item is the number of sensitive association rules containing the item and weight of a transaction is the sum of weights of all items contained in a transaction. The MDSRRC first finds the victim item having maximum sensitivity where the sensitivity of an item is the number of sensitive association rules containing the item in the right hand side. Next the algorithm deletes the item having maximum sensitivity from the transactions containing the victim item and having maximum weight.

Representative association rules are the smallest set of rules that covers all association rules in a given database with respect to the predefined support and confidence thresholds. The study in [27] first uncovers sensitive representative rules with using the GSEE algorithm and then selects all sensitive representative rules that have sensitive item(s) on the right hand side of the rule. Next the EDSR algorithm hides sensitive itemsets with deleting sensitive item(s) from transactions supporting these previously selected rules. The EDSR algorithm starts modifying the transactions according to the number of items they contain (size). The HRR algorithm which is combination of ISL (Increase Support of LHS) and DSR (Decrease Support of RHS) [29] is proposed in [28]. The HRR first uncovers representative rules containing the sensitive itemset on the left hand side of representative rules and then deletes sensitive itemsets from transactions supporting these uncovered representative rules. Next the algorithm uncovers transactions

containing the sensitive itemset on the right hand side of representative rules and then deletes sensitive itemset from transactions supporting these uncovered representative rules.

Particle swarm optimization is first proposed by [34] and its inspired from bird flocking to find food sources. The particles represent the problem solutions and each particle has velocity representing the flying directions according to other solutions. PSO2DT [35] is a particle swarm optimization based frequent itemset hiding approach. In this approach the maximum number of transactions to be removed is equal to the particle size. The particle size is calculated with the difference between the highest support count of frequent itemset among all frequent itemsets and minimum support count threshold. PSO2DT sanitizes frequent itemsets with deleting sufficient amount of transactions from the database.

The lattice theory was first proposed by [40] and the lattice of frequent itemsets can be used for selecting the victim item. Intersection lattice based frequent itemset hiding algorithms first generates the intersection lattice of all frequent itemsets, then it calculates the number of supersets of each itemset where an itemsets p_i is a superset of itemset i if I $\subseteq sp_i$. If I is the set of all itemsets in database D then I $\cap sp_i \in$ I so it can be inferred that I is a intersection lattice. The generating set of I is denoted as GS (I) and it is composed of smallest set of itemsets of I where every itemset in I can be generated by intersecting some itemsets in GS (I). The HCSRIL (Heuristic for Confidence and Support Reduction based on Intersection Lattice) [38] algorithm selects the victim item and minimum number of transactions modification of this victim item that causes the minimum impact on itemsets in GS (FI) where FI is the frequent itemsets in the original database. The AARHIL (Algorithm of association rule hiding based on intersection lattice) [39] algorithm tries to give minimum distortion on non-sensitive association rules and intersection lattice of frequent itemsets. This algorithm selects the transactions for modification according to their weights where the calculation of the weight is based on the number of sensitive and non-sensitive rules a transaction contains. Also the AARHIL algorithm selects the victim item that gives least impact on GS (FI).

All these distortion based heuristic algorithms explained above try to reduce the side effects and execution time with employing different methodologies and with different data structures. Also most of them allow database owner to assign a single sensitive threshold for each sensitive itemset. However as indicated in [20] different

itemsets have different support values in the database and sensitive threshold of different itemsets should be set according to their importance such as sensitive threshold of cheaper itemsets in a supermarket transactional database should be set higher then expensive itemsets because the cheap itemsets have high support.

3.2. Sanitization Algorithms for Dynamic Environment

In this section literature review of existing distortion based frequent itemset hiding algorithms for dynamic environment is presented. These algorithms are called dynamic sanitization algorithms to ease the complicity of the explanation. In dynamic environment the transactional database is being continuously updated by increments. The main challenge of dynamic sanitization mechanisms is to maintain large number of sensitive transaction search space. As illustrated in Figure 13 the dynamic transactional database consists of original part and incremental part. In dynamic environment the sensitive itemsets can hidden with two different approaches. The first approach modifies transactions in only incremental part and the second approach modifies transactions in the whole updated database. For the first approach if the original database D has already been sanitized then concealing sensitive itemsets in only incremental part d and then unifying d with D is going to produce a sanitized updated database. The first approach sanitizes the incremental part and then unifies the original part D with the sanitized incremental part d'. For the second approach the first the incremental part d is unified with the original part D and then the sanitization process is performed on the whole updated database ($D \cup d$).

TID	Itemset	
1	xyz	Original part (D)
2	xy	
3	yz	
4	wxyz	
5	wx	
6	xyz	Incremental part (d)
7	wxyz	

Figure 13. Dynamic transactional database.

3.2.1. Sanitization on Incremental Database

Sanitizing only the incremental part is more efficient in terms of execution time and resource allocation because the transaction search space for the hiding solution is smaller than the whole transactions in the updated database. But this strategy causes high rate of corruption on non-sensitive knowledge and data because its search space only includes transactions of the incremental part.

The RHID [24] algorithm is designed for hiding sensitive rules in dynamic environment with decreasing support of itemsets in the right hand side of a given rule. It uses the same methodologies and techniques as in MDSRRC [25] algorithm, the only difference is that, the RHID algorithm employs a table to keep actual number of necessary support decrease need for each sensitive rule. This table is updated whenever a new batch of transactions is inserted to the original part of the database. The RHID algorithm just modifies transactions of the incremental part so it does not guarantee to keep distortion on the database and loss of non-sensitive information at minimum level.

3.2.2. Sanitization on Updated Database

As the number of potential transactions for modification includes all transactions of the updated database ($D \cup d$), sanitizing the updated database is more efficient in terms of information loss, distance and accuracy. However, this strategy is disadvantageous in terms of execution time and memory consumption because of huge transaction search space. If necessary precautions are taken such as appropriate data structure usage then these disadvantages can be tolerated.

A dynamic frequent itemset hiding algorithm SPITF is proposed in [26]. This algorithm uses same concepts of TTBS [13] algorithm for sanitizing the sensitive itemsets. The SPITF algorithm groups sensitive itemsets sharing common item and then uncovers transactions containing these itemsets. The SPITF selects the item having maximum degree of conflict as victim item and removes it from sufficient amount of sensitive transactions. Also a tree like data structure called Sensitive Pattern Indexed Transaction Forest (SPITF) was designed to increase the efficiency of execution time. This data structure stores all transaction in the database and allows modifying transactions

without accessing the database. This approach reduces side effects such as distance and information loss since it performs the sanitization process on the whole updated database.

Although most of the transactional databases are dynamic in real life, there is only a two sanitization algorithms; SPITF and RHID designed for dynamic database environment. The RHID algorithms is incapable of finding the optimal hiding solution because the optimal hiding solution can only be found if the transaction search space for the modification encapsulates all transactions in the database. The SPITF on the other hand may fail to hide all given sensitive itemsets because uncovering all sensitive transactions from the tree like data structure that it employs may fail for some cases.

3.3. Summary

In this chapter we reviewed existing frequent itemset and rule hiding approaches in the literature. These approaches are classified into two major categories according to the environment they designed for; static and dynamic. In dynamic environment there is only two heuristic itemset/rule sanitization algorithms whereas for the static environment, there are many algorithms grouped as the border based, exact, reconstruction based and heuristic itemset/rule hiding approaches. The border based approaches separate the non-frequent and frequent itemsets with a border in the itemset lattice and then revises this border to find out an optimal hiding solution. Border based approaches give minimum distortion to the original database but they may unable to find optimum hiding solution for some cases although there exists [22]. The exact approaches have very high computational time as they employ integer programming so this makes them impracticable. The reconstruct based approaches may have problem in generating the resulting database with the same size as the original database and also putting non-frequent itemsets into transactions of the resulting sanitized database is a difficult task. Heuristic based approaches are not designed for finding an overall optimum solution to the sanitization problem but they usually find solution close to the best one with small response time and in the literature there are many studies based on heuristic approaches [38].

Basic properties of the existing distortion based heuristic sanitization approaches are given in Table 4. In this table the “Algorithm” column indicates the algorithm name,

the “Hiding” column shows if the algorithm is designed for hiding sensitive itemset or hiding sensitive association rules. The “Victim Item Selection” column gives the victim item selection criteria of the algorithm where “degree” shows the selection of the item is done according to degree of conflict which is the number of occurrences of the item in different sensitive itemsets, “support” shows the item selection is done according to its support in the database, “iterative” shows the item is selected in a trial and error way, “weight” shows the item selection is done according to weight of the item where the weight is calculated depending on some heuristics, “lattice” shows the item is selected according to intersection lattice and “none” shows the algorithm does not select any victim item instead it completely removes some set of sensitive transactions from the database. The “Transaction Selection” column indicates selection criteria of the transactions for modification where “size” shows that the transactions are selected according to number of items contained by transactions, “Degree” shows that the selection is done according to number of sensitive rules or itemsets contained by transactions, “Greedy” shows that the selection is done in a trial and error way and “Weight” shows that the selection is done with using previously calculated weights of transactions.

The “Multiple rule/itemset Hiding” column shows whether the algorithm is designed for hiding more than one sensitive rules or itemsets at each iteration of the hiding process. The column “Multiple Support Threshold” indicates whether the algorithm enable assigning different sensitive thresholds for each sensitive itemset or not. The “Environment” column indicates if the algorithm is designed for the incremental or static database environment.

According to the Table 4, most of the algorithms are designed for hiding itemsets rather than association rules. Some of the algorithms employ different heuristics for modifying transactions besides some of the algorithms completely remove predefined set of transactions from the database. The modification set of transactions are mostly selected according to their weights and the weight of transactions are calculated based on some heuristics such as the number of sensitive itemset or rule a transaction supports. If intersection of sensitive itemsets or rules is not empty in other words if they share common item or items then they are called overlapping itemsets or rules. Most of the proposed approaches are designed with assuming that sensitive itemsets or rules are overlapping.

After the heuristic based itemset/association rule hiding algorithms are surveyed it can be discovered that most of these hiding algorithms allow database owner to define a unique sensitive threshold for all sensitive itemsets. However unique sensitive threshold gives limitation to the database owner and also prevents to consider the significance of different characteristics of sensitive itemsets. Also during the sanitization process if supports of all sensitive itemsets are decreased under a unique sensitive threshold then an adversary may infer that some itemsets are made uninteresting before the database is shared.

The second problem of existing solutions is, there is only two algorithms designed for dynamic database environment. It is possible to use any static environment sanitization algorithm to the dynamic database environment with waiting all updates and performing the sanitization operation on the whole updated database. Applying this type of solution is going to bring an overhead the resource allocation and execution time because dynamic databases are updated to continuously and whenever a database publish operation is performed the sanitization process is going to start from beginning.

Table 4. Classification of distortion based heuristic frequent itemset hiding algorithms

Algorithm	Hiding	Victim Item Selection	Transaction Selection	Multiple rule/itemset Hiding	Multiple Support Threshold	Environment
RHID [24]	Rule	Weight	Weight		✓	Dynamic
SPITF [26]	Itemset	Degree	Degree		✓	
TTBS [13]	Itemset	Degree	Degree		✓	Static
SWA [12]	Itemset	Support	Size	✓	✓	
MDSRRC [25]	Rule	Weight	Weight		✓	
HSARWI[21]	Rule	Weight	Weight	✓		
FHSAR [20]	Rule	Degree	Weight	✓		
MICF [16]	Itemset	Degree	Weight	✓		
Aggregate [10]	Itemset	None	Greedy	✓		
Disaggregate [10]	Itemset	Greedy	Greedy	✓		
Hybrid [10]	Itemset	Greedy	Greedy	✓		
IGA[11]	Itemset	Degree	Degree	✓		
RelevanceSorting[18]	Rule	Support	Weight			
EDSR[27]	Itemset	None	Size			
HRR[28]	Rule	Support	All			
SIF-IDF[17]	Itemset	Support	Weight			
Algorithm 2b [15]	Itemset	Support	Size			
Algorithm 2.c [15]	Itemset	None	Size			
PDA[19]	Rule	Greedy	Weight			
WDA [19]	Rule	None	Weight			
Naïve[11]	Rule	All	Degree			
MaxFIA [11]	Itemset	Support	Degree			
MinFIA [11]	Itemset	Support	Degree			
PSO2DT [35]	Itemset	All	Weight	✓		
PGBS [14]	Itemset	Degree	Degree	✓	✓	
HCSRIL[38]	Rule	Lattice	Size			
AARHIL[39]	Rule	Lattice	Weight			

CHAPTER 4

PRIVACY PRESERVING FREQUENT ITEMSET HIDING

In this thesis three different distortion based frequent itemset hiding algorithms; PGBS [14], IPGBS and DynamicPGBS [36] are proposed. Both PGBS and IPGBS are designed for static database environment whereas the DynamicPGBS is designed for dynamic database environment. These sanitization approaches hide given sensitive itemsets by decreasing their support under predefined sensitive thresholds with deleting victim items from some sensitive transactions. In order to ease the complexity of the hiding solution some heuristic are employed in the design phase of these three algorithms. These heuristics are used for determining the transactions for modification and item to be deleted from them. In order to increase the efficiency of sensitive transaction and victim item identification all PGBS, IPGBS and DynamicPGBS use Pseudo Graph based data structure.

4.1. The Pseudo Graph Data Structure

The essential steps of distortion based frequent itemset hiding process includes identifying sensitive transactions, counting supports of sensitive itemsets and counting support of candidate victim items. All these steps can be acquired with multiple database scan operations however performing sequential search operation on the actual database for once has O (|D|) worst case time complexity. To decrease the time complexity of the search operation and reduce the number of database scan operations to one, the database is presented as Pseudo Graph data structure. Performing scan operations on Pseudo Graph rather than the actual database or other data structures like matrix or inverted index provides significant improvement in terms of execution time.

Pseudo Graph. A graph G (V, E) is a set vertices V and a set edges E where some vertices are connected by edges. If a graph consists of ordered pair of vertices, then it is called directed graph. A loop is an edge that connects a vertex to itself. A pseudo graph is a directed graph which allows multiple edges and loops.

In PGBS all transactions in the given database D are represented as Pseudo Graph (PG) without checking their contents. The PGBS utilizes the advantage of PG for efficiently identifying the sensitive transactions and calculating support counts of items and sensitive itemsets. In IPGBS all sensitive itemsets are represented as Pseudo Graph. The IPGBS utilizes the advantage of PG for identifying sensitive transactions that contain maximum number of sensitive itemsets. In DynamicPGBS all sensitive transactions are represented as Pseudo Graph. The DynamicPGBS utilizes the advantage of PG for efficiently calculating support counts of items in sensitive transactions and sensitive itemsets.

4.2. The Pseudo Graph based Sanitization Algorithm (PGBS)

The PGBS algorithm sanitizes the given transactional database D by reducing support of sensitive itemsets under their predefined sensitive thresholds. The support reduction is done by removing sufficient amount of items from sufficient amount of sensitive transactions. The PGBS algorithm converts the given database D into sanitized database D' with zero Hiding Failure in other words no sensitive itemsets can be extracted from D'. The aim of this algorithm is keeping maximum number of non-sensitive itemsets present in D and cause minimum item removal on D. The two main sub problem of this algorithm are determining the set of sensitive transactions for modification and selecting the victim item to be removed from these sensitive transaction set. The PGBS algorithm is based on two different heuristics for determining the set of transactions for modification and victim item selection. The number of sensitive itemsets that are uncovered from a single transaction is referred as cover degree. For illustration in Figure 14 (a) a sample transactional database and in Figure 14 (b) sensitive itemsets of this database are shown. The cover degree column of Figure 14 (a) indicates the cover degree of each transaction. For many cases it is possible to reduce the number of transaction modification by selecting high cover degree. Because this enables to sanitize more than one sensitive itemset at once by modifying a single transaction. Also modifying less number of transactions reduce the number of iterations of the sanitization algorithm. The first heuristic is based on this fact and selects transactions according to their cover degrees.

Transaction ID	Transaction	Cover Degree
1	abcdef	3
2	bc	1
3	bcdefg	2
4	abcefg	2
5	abceg	2
6	abcdf	3
7	df	1
8	abf	1

(a) Transactional database

Sensitive Itemset	Sensitive Thresold
ab	30%
bc	60%
df	20%

(b) Sensitive itemsets

Figure 14. Sample database and sensitive itemsets with their sensitive thresholds.

If intersection of two or more sensitive itemsets is not empty with another meaning they share a common item then removing this item from sensitive transaction reduces support of more than one sensitive itemset at the same time.

The flowchart of PGBS algorithm is given in Figure 15. The algorithm takes the database D, the set of sensitive itemsets (SI) and their sensitive thresholds (SI) as input. This algorithm consists of four main processes where the first process is converting the database D into PG. The second process is creating the Sensitive Count Table (SCT) where the SCT keeps the number of necessary support count decrease of each sensitive itemset. The third process creates the Sanitization Table (ST), this table keeps the information related to modification operation that is going to applied to D. The last process is the sanitization process; this process creates a copy of the database and applies the sanitization information stored in the ST to the copy of the database.

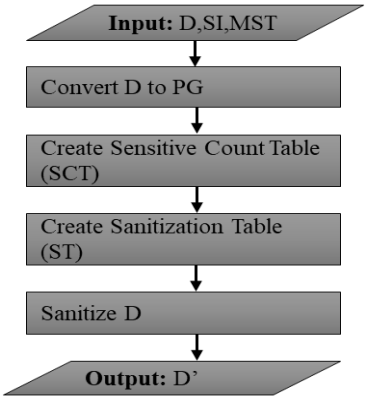


Figure 15. The flowchart of PGBS algorithm.

4.2.1. Pseudo Graph (PG)

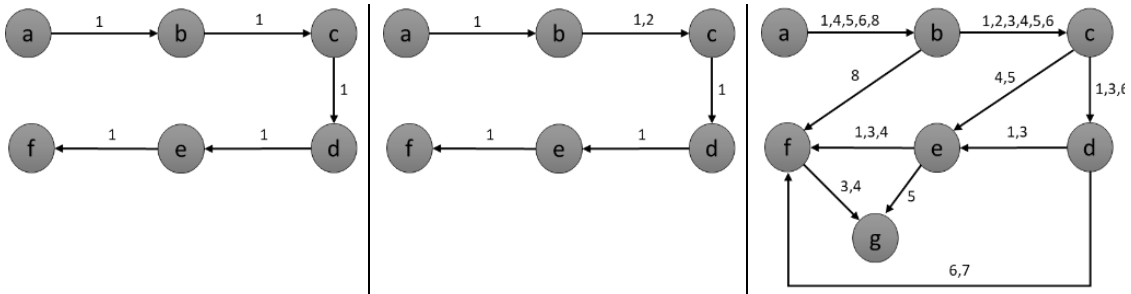
The Pseudo Graph used in PGBS algorithm represents all transactions in the given database D with Pseudo Graph. The vertices represent each different item in D and they are connected to each other by edges where edges are labelled with transaction ids. As an example if item i_1 is connected with item i_2 with a directed edge labeled with n then this means that i_1 appears with i_2 in the nth transaction. The vertices having reflective edges represent the transactions containing a single item.

To create the PG initially all transactions in D are sorted in lexicographic or alphabetic order and then each transaction in D is inserted one by one to the PG. The insertion process first checks whether there exists a vertex for each item i of the given transaction tr and if not vertex labelled with i is created. Then each item of transaction tr is connected to each other in sequential order and the edges connecting these vertices are labelled with the id of tr. If there exists any transaction tr containing only one item i then a self-loop labelled with the id of tr is created for the vertex i. The algorithm for creating the Pseudo Graph of a given database D is depicted in Algorithm 1. First transactions are read by one by from the database and then for each item of the transaction vertices are created if it is not present. Next vertices representing each item of a given transaction are connected sequentially labelled with the transaction ids.

Algorithm 1: Creating Pseudo Graph

```
Input: Transactional Database D
Output: Pseudo Graph representation of transactions in D
foreach transaction tr ∈ D do
    foreach item j ∈ tr do
        if PG not contains any vertex labeled with j then
            create a new vertex v_j
        end
    end
    if tr contains only one item j then
        create a self loop for vertex v_j labbelled with transaction id of tr
    else
        foreach item j ∈ tr do
            if there is a successor item i of the item j then
                put an outgoing edge from vertex v_j to vertex v_i labeled
                  with transaction id of tr
            end
        end
    end
end
```

The conversion process of the database D into PG is illustrated in Figure 16. Suppose the database D in Figure 14 (a) is given and all transactions in D are alphabetically sorted before the conversion process. The Pseudo Graph given in Figure 16 (a) and (b) shows transactions “abcdef” and “bc” are inserted into PG respectively and the Figure 16 (c) shows PG after all remaining transactions in D are inserted PG.



(a) PG after transaction “abcdef” inserted

(b) PG after transaction “bc” inserted

(c) PG after all transactions inserted

Figure 16. Inserting transactions into PG.

Transactions containing a given 2-itemset is uncovered from PG with using the intersection operation. Intersecting the transactions ids on the outgoing edge of one vertex with transaction ids on the incoming edge of the other vertex gives transactions containing both items at the same time. Suppose XY be a 2-itemset and prefix(X) detonate transaction ids on outgoing edges of vertex X and postfix denote transaction ids on incoming edges of vertex(Y). The support count of XY in database is calculated by prefix(X) $\cap$ postfix(Y). Similarly the support count of a k-itemset is calculated by prefix($item_1$) $\cap$ postfix($item_2$) $\cap$ … $\cap$ postfix ($item_k$) where the sequence of an item $item_N$ in k-itemset is denoted as N. As an example transactions containing the itemset “ade” in Figure 16 (c) is calculated by ({1,4,5,6,8} $\cap$ {1,3} $\cap$ {1,3,4,5}) = {1} so the support count of itemset “ade” is 1. Also the support count of an item is equal to the total number of distinct transaction ids on the incoming and outgoing edges. As an example support count of “c” in Figure 16 (c) is {1,2,3,4,5,6} $\cup$ {1,3,4,5} = {1,2,3,4,5,6} which means the support count of item “c” is 6.

4.2.2. Creating the Sensitive Count Table

Sensitive Count Table (SCT) represents the minimum number of support count decrease required to hide a sensitive itemset. The SCT has three attributes, SID, SI and N_{Modify}. The SID is the unique identifier of records in SCT, SI is the sensitive itemset and N_{Moidfy} is the minimum number of support count decrease to hide a given sensitive itemset. The N_{modify} is calculated with the following equation;

$$N_{Modify} = \lfloor scount(X) - st(X) * |D| + 1 \rfloor \quad (4.1)$$

where X is the sensitive itemset, the scount (X) is the number of transactions supporting X in D, st (X) is the sensitive threshold of X and |D| is the total number of transactions in database D. After the SCT is created it is sorted in descending order of N_{Modify} attribute. For illustration, in Table 5 the N_{Modify} of the sensitive itemset "ab" is 3; it means after item "a" or item "b" is deleted from 3 transactions of the database then the sensitive itemset "ab" will become sanitized.

Table 5. Sensitive Count Table (SCT) of PGBS and IPGBS algorithms.

SID	SI	N_{Modify}
0	ab	3
1	df	3
2	bc	2

4.2.3. Creating the Sanitization Table

The process of creating the Sanitization Table consists of creating the sanitization Table (ST) and updating PG. The Sanitization Table (ST) keeps the final modification information that will be applied to the database before it is published. The ST has two attributes, Victim and Transactions. The victim attribute keeps the victim item that is selected to be deleted and the transactions attribute keeps the list of transactions that this victim item will be deleted. After each victim and corresponding transaction set for modification is determined it is put into the Sanitization Table and consequently the Pseudo Graph is updated by deleting this victim item from the transactions determined.

The algorithm of creating the Sanitization Table (ST) is depicted in Algorithm 2. The algorithm starts the sanitization process with the sensitive itemset having greatest

N_{Modify} value. The victim item is selected among items in a given sensitive itemset having maximum conflict degree. If there is more than one victim item having the same conflict degree the victim item is selected with maximum support count value among candidate victim items. If there is still more than one candidate victim item, then it is selected randomly among them and then put into the variable *victim*. The *unifiedItemsets* variable in the algorithm unifies all sensitive itemsets in SCT sharing the *victim* variable. If the number of sensitive itemsets in the *unifiedItemsets* variable is equal to one, then this implies that the variable *victim* is not contained in any different sensitive itemset than the active sensitive itemset *si* and it is impossible to unify the *si* with any other sensitive itemset in SCT. Next the ids of sensitive transactions supporting *unifiedItemsets* are uncovered from PG with not exceeding the N_{Modify} of the *si* and put into variable *sensitiveTransactions*. Then the *victim* and the *sensitiveTransactions* are added to the sanitization Table (ST). The PG is updated by deleting the *victim* from transactions stored in *sensitiveTransactions* and the N_{Modify} of each sensitive itemset in *unifiedItemsets* is reduced by the number of transaction ids stored in *sensitiveTransactions.* If N_{Modify} of any record in SCT becomes less than or equal to zero, then it is removed from both SCT and *unifiedItemsets* to avoid decreasing the support of already sanitized sensitive itemset more than necessary. Also database D may not contain sufficient number of transactions supporting *unifiedItemsets*, in such a case the sensitive itemset having the least N_{Modify} is removed from *unifiedItemsets* at each iteration till only the active sensitive itemset *si* remains in *unifiedItemsets*. The creation of the Sanitization Table terminates when there is no remaining row left in SCT.

4.2.4. Illustrating Example

To illustrate how the algorithm depicted in Algorithm 2 works suppose the Sensitive Count Table (SCT) is given in Table 5 and Pseudo Graph (PG) is given in Figure 16 (c) as input. In this example there are three sensitive itemsets; “ab”, “df” and “bc” where the itemset “ab” has the maximum N_{Modify} value which is 3. So the sanitization process starts from the “ab” (step 1). The victim item is selected as “b” (line 2) among items in “ab” and put into variable *victim* because it has the maximum conflict degree (conflict degree= 3). Then the sensitive itemsets containing “b” are unified (line 3) and

put into the variable *unfiedItemsets*. The *unifiedItemsets* is “abc” and according to the PG in Figure 16 (c) transactions containing “abc” are {1,4,5} (line 11). N_{Modify} of “ab” and “bc” are updated in SCT as 0 and -1 respectively (line 12) and the sensitive itemsets “ab” and “bc” are removed from SCT and *unifiedItemsets* (lines 12-18) because their N_{Modify} value become less than or equal to zero. The victim “b” and transaction ids {1,4,5} pair is put into the Sanitization Table (line 19) and item “b” is removed from transaction {1,4,5} in PG as shown in Figure 17 (a) (line 20). The next non-sanitized sensitive itemset in SCT is “df” and the victim is selected as “f” because it has the maximum support count value (support count f = 4) (lines 1-2). The *unifiedItemsets* variable is “df” because the only remaining sensitive itemset in SCT is “df” (line 3). Transactions containing “df” with not exiting the N_{Modify} of “df” are {1,3,6} and they are assigned to variable *sensitiveTransactions* (line 5). The PG is updated as in Figure 17 (b) where the item “f” is removed from transactions {1,3,6} (line 7) and the sensitive itemset “df” is removed from SCT (line 8). The creation of the Sanitization Table terminates because there is no record left in SCT.

4.2.5. Sanitizing the Database

The last process of PGBS is applying the sanitization solution stored in the Sanitization Table (ST) to the original database D and create a sanitized database D’. The whole sanitization process is illustrated in Figure 18. In this figure first the given database D is converted to the Pseudo Graph (PG) representation, then the Sensitive Count Table (SCT) is created with using the PG and the previously defined sensitive itemsets. The third step creates the Sanitization Table (ST) with using the PG and information stored in SCT. The final step creates a copy of the database D and applies the information stored in ST to this copy to produce the D’.

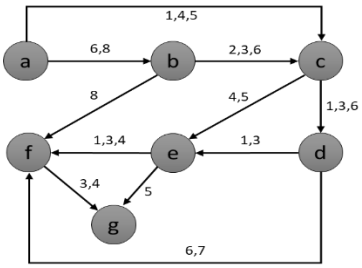
4.3. The Itemset Oriented Pseudo Graph based Sanitization Algorithm (IPGBS)

As the PGBS algorithm the Itemset Oriented Pseudo Graph (IPGBS) algorithm is designed for frequent itemset hiding in the static environment. The main similarities

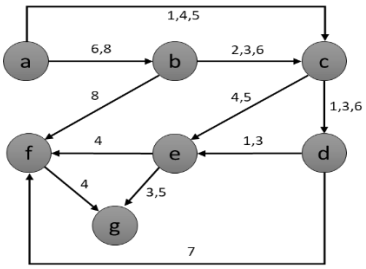
Algorithm 2: Creating Sanitization Table

Input: Sensitive Count Table SCT, Pseudo Graph PG
Output: Sensitive Transactions Table

1 **foreach** *sensitie itemset si* ∈ *SCT where* $si.N_{Modify}>0$ **do**
2 *victim* ← maximum cover degree item in si or maximum support or randomly selected and item being selected first time for si
3 *unifiedItemsets* ← unify all sensitive itemsets in SCT that contains the *victim*
4 **if** *number of sensitive itemsets in* unifiedItemsets == *1* **then**
5 sensitiveTransactions←uncover transactions containing *unifiedItemsets* from *PG* with not exceeding the $si.N_{Modify}$
6 Add *victim* and sensitiveTransactions to ST
7 Delete *victim* from sensitiveTransactions in PG
8 Remove *si* from SCT
9 **end**
10 **while** *number of sensitive itemsets in* unifiedItemsets>*1 and* $si.N_{Modify}>0$ **do**
11 sensitiveTransactions←uncover transactions containing *unifiedItemsets* from *PG* with not exceeding the $si.N_{Modify}$
12 **foreach** *sensitive itemset s in* unifiedItemsets **do**
13 $s.N_{Modify}$←$s.N_{Modify}$-|sensitiveTransactions|
14 **if** $s.N_{Modify} \leq 0$ **then**
15 Remove *s* from SCT
16 Remove s from *unifiedItemsets*
17 **end**
18 **end**
19 Add *victim* and *sensitiveTransactions* to ST
20 Delete *victim* from sensitiveTransactions in PG
21 **if** *non of the sensitive itemsets in* unifiedItemsets *sanitized* **then**
22 Remove the sensitive itemset having least N_{Modify} from *unifiedItemsets*
23 **end**
24 **end**
25 **if** $si.N_{Modify} >0$ **then**
26 goto step 2
27 **end**
28 **end**



(a) PG after item "b" is deleted from transactions {1,4,5}



(b) PG after item "f" is deleted from transactions {1,3,6}

Figure 17. Updating PG with deleting items.

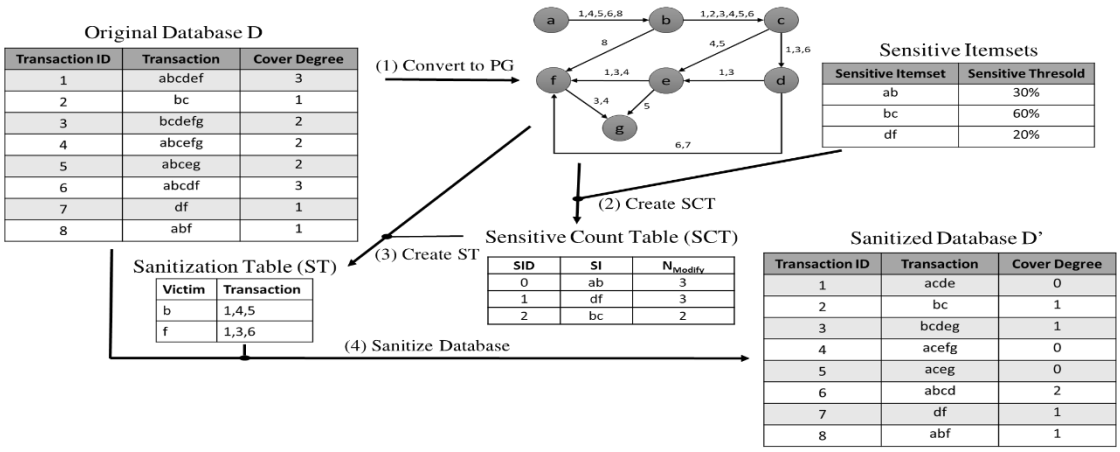


Figure 18. The flow of the processes in PGBS.

between these two algorithms are first both of them hide sensitive itemsets with decreasing their support under their sensitive thresholds and the second both employ the Pseudo Graph data structure during the sanitization process. However, the main focus of IPGBS algorithm is reducing the amount of information loss whereas the main focus of PGBS algorithm is to minimize the execution time. The Psudo Graph data structure used in IPGBS is called Itemset Oriented Pseudo Graph (IPG). The IPG keeps only the sensitive itemsets and the transactions containing them.

4.3.1. Itemset Oriented Pseudo Graph (IPG)

The Pseudo Graph used in IPGBS algorithm represents all sensitive itemsets in the given database D and it based on the Pseudo Graph proposed in PGBS algorithm. The vertices represent each different sensitive itemset and they are connected to each other by edges where edges are labelled with transaction ids. The transaction ids on the edges represent the set of transaction ids containing the sensitive itemsets on the path between starting vertex and its direct successor. As an example if itemset is_1 is connected with item is_2 with a directed edge labeled with n then this means that is_1 appears with is_2 in the nth transaction. The vertices having reflective edges represent the transactions containing a single itemset.

The procedures of manipulating IPG include construction of the graph, insertion of transactions and deleting transactions from the graph. Constructing the IPG and insertion of the transactions is depicted in Algorithm 3. First each transaction in database

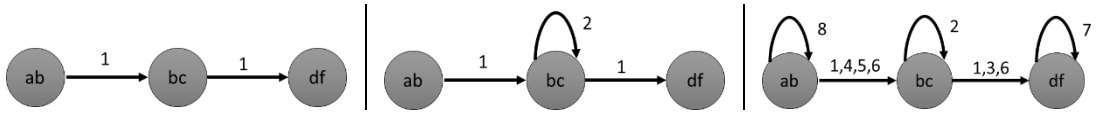
D is checked to find out whether it contains any sensitive itemset, and if so these sensitive itemsets are put into the variable *sItemsets.* Then if IPG does not contain any vertex labelled with any itemset *si* in *sItemsets,* a vertex labelled with *si* is created. Next, if the variable *sItemsets* contains more than one sensitive itemset then vertices in IPG labelled with these sensitive itemsets are connected with each. The label on the edge connecting these vertices is composed of transaction ids containing these sensitive itemsets. Also a reflective edge is created for the last sensitive itemset in *sItemsets* to indicate the finish vertex of the deepest path on the graph. If the transaction *tr* contains only one sensitive itemset then a loop is created in IPG for the vertex labelled with this sensitive itemset.

The construction of the IPG is illustrated in Figure 19. Suppose the transactional database and sensitive itemsets are given in Figure 14 (a) and (b) respectively. First each transaction is checked in sequential order to find out if it is sensitive and if so it is inserted into the IPG. The first sensitive transaction in D is "abcdef" with transaction id 1. Vertices labelled with "ab", "bc" and "df" are created and connected with each other with edge 1 as in Figure 19 (a). The next transaction in D is "bc", and it contains only one sensitive itemset "bc", because there is no any other sensitive itemset in this transaction a reflective edge labelled with 2 is created on vertex "bc" as shown in Figure 19 (b). After all remaining sensitive transactions in D are inserted into the IPG the resulting IPG is shown in Figure 19 (c).

Algorithm 3: Creating Pseudo Graph of IPGBS

Input: Transactional Database D
Output: Itemset Pseudo Graph (IPG) representation of sensitive itemsets in D

```
1  foreach sensitive transaction tr ∈ D do
2      SItemsets ←all sensitive itemsets in tr
3      foreach sensitive itemset si ∈ SItemsets do
4          if IPG does not contain any vertex labeled with si then
5              create a new vertex V_si
6          end
7      end
8      N←Number of sensitive itemsets in tr
9      for i←1 to N do
10         if i ≠ N then
11             put an outgoing edge from V_SItemsets[i] to V_SItemsets[i+1]
                 labeled with transaction id of tr
12         else
13             put an outgoing edge from V_SItemsets[i] to V_SItemsets[i]
                 labeled with transaction id of tr
14     end
15 end
```



(a) IPG after transaction "abcdef" inserted

(b) IPG after transaction "bc" inserted

(c) IPG after all transactions inserted

Figure 19. Inserting transactions into IPG.

Transactions supporting a given sensitive itemsets *si* is equal to the distinct transaction ids on the incoming and outgoing edges of vertex *si*. Similarly transactions supporting N number of sensitive itemsets is calculated by Prefix ($sensitiveItemset_1$) ∩ Postfix($sensitiveItemset_2$) ∩ … ∩ Postfix ($sensitiveItemset_N$). As an example the transactions supporting the sensitive itemsets "ab", "bc" and "df" are uncovered from IPG in Figure 19 (c) by {1,4,5,6,8} ∩ {1,2,4,5,6} ∩ {1,3,6,7} = {1,6}. The procedure deleting transaction from the graph is only includes deleting specified transaction from the outgoing edges of vertices.

4.3.2. IPGBS Algorithm

The IPGBS (Itemset Oriented Pseudo Graph Based Sanitization) algorithm is a distortion based frequent itemset hiding algorithm. The main objective of the IPGBS algorithm is to reduce the non-sensitive information loss during sanitization operation. In order to keep the difference between number of non-sensitive information in the original and sanitized database minimum number of transactions are modified. The IPGBS algorithm starts the modification operation from transactions containing maximum number of sensitive itemsets. To reduce the execution time complexity of uncovering transactions having maximum degree of conflict from the actual database D the Itemset Oriented Pseudo Graph (IPG) is employed.

The flowchart of this algorithm is illustrated in Figure 20. As in PGBS the IPGBS algorithm takes the database D, the set of sensitive itemsets (SI) and their sensitive thresholds (SI) as input. The IPGBS algorithm consists of four main processes where the first process is converting the database D into IPG. The second process is creating the Sensitive Count Table (SCT) and the third process creates the Sanitization Table (ST).

The final process is the sanitization process; this process creates a copy of the database and applies the sanitization information stored in the ST to the copy of the database.

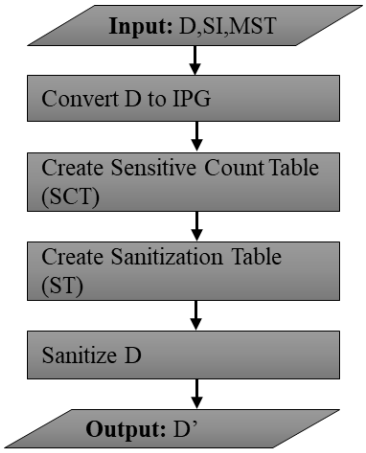


Figure 20. The flowchart of IPGBS algorithm.

Both Sensitive Count Table and Sanitization Table keep the same information as in PGBS. The detail information for creating the SCT is explained in previous sections and the same producers are followed in IPGBS. There is only one difference between the SCT created in IPGBS and PGBS. The SCT is sorted alphabetically or numerically according to SI values in IPGBS algorithm whereas SCT is sorted according to N_{Modify} in PGBS algorithm.

4.3.3. Creating the Sanitization Table

The IPGBS algorithm tries to find out transactions containing maximum number of sensitive non-sanitized sensitive itemsets and then deletes previously determined items (victims) from these transactions. It is possible to uncover transactions supporting maximum number of sensitive itemset by uncovering transaction ids of the deepest path in IPG. The deepest path of a sensitive itemset *si* is the longest path in IPG, starting from the *si.* The depth first search traversal method is the efficient way of finding the deepest path of a sensitive itemset.

The algorithm for creating the Sanitization Table is depicted in Algorithm 4. First each sensitive itemset*si* stored in the SI field of SCT is started to be selected by one by and if the *si* is not already sanitized the hiding operation for *si* starts. The variable

deepestPath stores the longest path in IPG starting from the given sensitive itemset*si*. The *deepestPath* does not contain any sensitive itemset that is already sanitized because in the opposite case support of sensitive itemset may decreased more than necessary and this may lead to effect more non-sensitive frequent itemsets in the resulting sanitized database. After the deepest path containing maximum number of strong sensitive itemsets is determined by using depth first search traversal technique, the transactions containing this path are extracted from IPG with not exceeding the N_{Modify} value of the corresponding sensitive itemset. Next the item having the maximum cover degree among items in *deepestPath* is selected as victim. If there exists more than one item having the same cover degree, the item with maximum support count is selected. If there is still more than one, then the victim item is selected randomly. After the sensitive transactions for modification and victim item are determined the IPG is updated by removing these transactions from outgoing edges of the active sensitive itemset*si*. This update operation is for avoiding selecting the same transactions more than once in the next iteration. Then these transactions and victim item pair is inserted into the Sanitization Table (ST). If any of the sensitive itemset stored in *deepestPath* contain the victim item, then N_{Modify} value of them are decreased by the number of transactions uncovered. Besides the selected victim item may not be common in all sensitive itemsets of the deepest path. In such a case these sensitive itemsets are inserted into the variable *NotContain* and a new victim item is selected for these sensitive itemsets. Then this victim item and sufficient number of transactions from previously uncovered transactions are inserted into the Sanitization Table (ST). The algorithm continues to generate deepest path and select victim item and uncover sensitive transactions till the given sensitive itemset *si* is sanitized or there does not exist any new deepest path for *si*. If there are insufficient number of transactions uncovered to hide sensitive itemset *si* and there is no more different deepest path left for the vertex *si* then insufficient number of transactions problem arises, which means there is an insufficient number of transactions containing the paths with length more than one starting from the vertex *si* in *IPG*. This problem appears due to the fact that given sensitive itemset may appear as the last element in most or all sensitive transactions when sensitive itemsets are sorted in alphabetically or numerically and as a result there would not be sufficient number of longest paths different from the loop pointing to the same vertex in IPG. The algorithm solves the insufficient number of transactions problem by selecting

the victim item among items in *si* having maximum support count value and uncovering sufficient number of transactions from the incoming edges of a given sensitive itemset.

4.3.4. Illustrating Example

To illustrate how the algorithm depicted in Algorithm 4 works, suppose the Sensitive Count Table (SCT) is given in Table 1 and Itemset Oriented Pseudo Graph (IPG) is given in Figure 19 (c) as input. In this example there are three sensitive itemsets; "ad", "cd" and "bd" where the itemset "ad" has the maximum N_{Modify} value which is 3. So the sanitization process starts from the "ab" (line 1). The deepest path starting from vertex "ab" is "abcdf" (line 2) and transactions {1,6} supports this path, so the transaction id {1,6} is inserted into the variable *transactions* (line 3). The item "b" is selected as victim item as it has the maximum conflict degree (conflict degree =2) and then put into the variable *victim* (line 4). The IPG is updated as in Figure 21 (a) by removing transaction {1,6} (line 5) and the victim and transactions pair is inserted into the Sanitization Table (ST) (line 6). N_{Modify} of "ab" and "bc" are updated as 1 and 0 respectively. The sensitive itemset "df" in the path "abcdf" does not contain the victim item "b" so the item "f" is selected as victim item among items "d" and "f" because it has the maximum support count (support count=6). The N_{Modify} of "df" in SCT is updated as 1 and then the victim item "f" with transaction ids {1,6} are inserted into the ST (lines 8-13). The next deepest path containing maximum number of non-sanitized sensitive itemsets starting from vertex "ab" is "ab", the deepest path "abc" is neglected because it contains the sanitized sensitive itemset "bc". The item "a" is selected randomly as victim among the items "ab" because both support count of "a" and "b" are the same (support count of a = 5, support count of b=5). The IPG is updated by removing transaction {8} from vertex "ab" as shown in Figure 21 (b), the victim "a" and transaction {8} is added to the Sanitization Table (lines 2-5). The next none sanitized sensitive itemset in SCT is "df" and the deepest path starting from the vertex "df" is "df" with transaction {7}. Finally, the victim "d" and transaction {7} is inserted into ST and the IPG is updated as in Figure 21 (c).

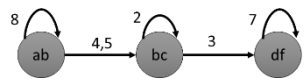
4.3.5. Sanitizing the Database

The sanitization process of IPGBS creates a copy of the original database D and then applies the sanitization solution stored in Sanitization Table (ST) to this copy. The whole sanitization process is illustrated in Figure 22. In this figure first the given database

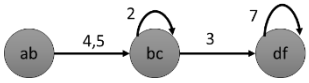
Algorithm 4: Creating Sanitization Table

Input: Itemset Pseudo Graph IPG, Sensitive Itemsets SI, Sensitive Thresholds
Output: Sanitization Table ST

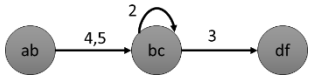
```
1  foreach sensitive itemseet si ∈ SCT where si.N_Modify > 0 do
2      foreach deepestPath starting from vertex si in IPG do
3          transactions← transactions containing deepestPath in IPG
           // number of transactions ≤ si.N_Modify
4          victim←maximum cover degree item in deepestPath
5          Remove transactions from outgoing edges of vertex si in IPG
6          Insert victim and transactions pair into ST
7          For all sensitive itemset s in deepestPath where victim ⊂ s
            reduce N_Modify of s by |transactions|
8          For all sensitive itemset s in deepestPath and victim ⊄ s add s
            to the varible NotContain
9          while NotContain is not empty do
10             victim←maximum cover degree item in NotContain
11             For all sensitive itemset s in NotContain where victim ⊂ s
                reduce N_Modify of s by |transactions| and Remove s from
                NotContain
12             Insert victim and transactions pair into ST // number of
                transactions ≤ maximum N_Modify in NotContain
13         end
14     end
15 end
```



(a) IPG after the path "abcdf" is deleted from transactions {1,6}



(b) IPG after path "ab" is deleted from transaction {8}



(c) IPG after path "df" is deleted from transaction {7}

Figure 21. Updating IPG with deleting paths.

D is converted to the Itemset Oriented Pseudo Graph (IPG) representation, then the Sensitive Count Table (SCT) is created with using the IPG and the previously defined sensitive itemsets. The third step creates the Sanitization Table (ST) with using the IPG and information stored in SCT. The final step creates a copy of the database D and applies the information stored in ST to this copy to produce the D'.

4.4. Dynamic Frequent Itemset Hiding Algorithm (DynamicPGBS)

The Dynamic Pseudo Graph based Sanitization algorithm [36] is designed for hiding frequent itemsets in the dynamic environment. In dynamic environment transactional databases are continuously being updated by receiving increments. A static

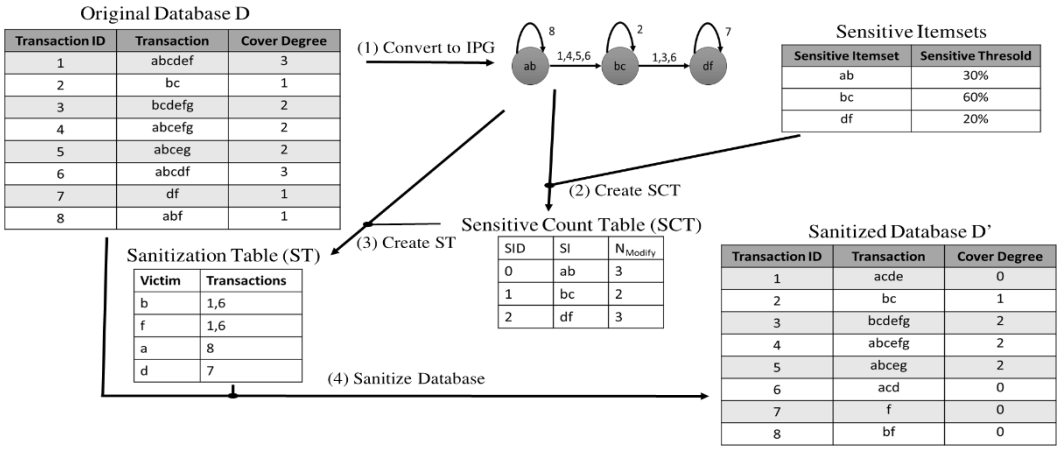


Figure 22. The flow of the processes in IPGBS algorithm.

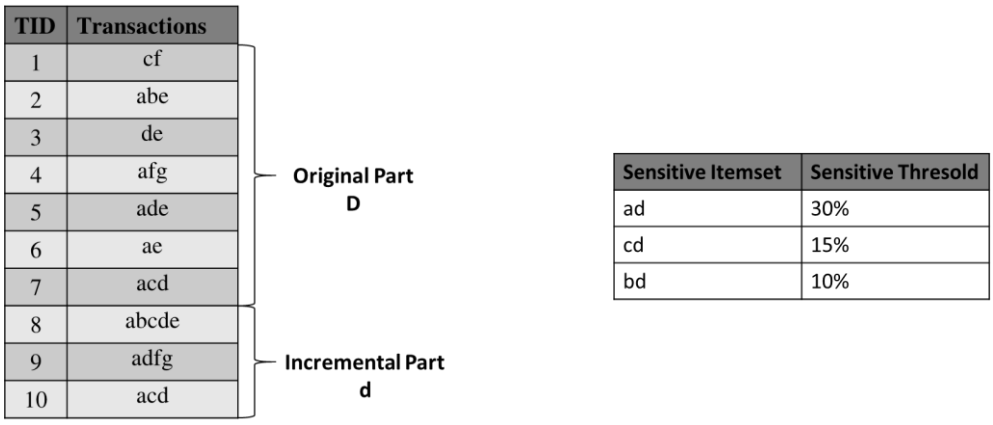
frequent itemset hiding algorithm can be applied to the dynamic database environment by waiting all increments and then applying the sanitization process to the whole updated database. However, this will be inefficient in terms of total resource allocation and execution time because whenever an incremental part arrives to the database the sanitization process should start from scratch to consider the updates.

4.4.1. DynamicPGBS Algorithm

The database D in a dynamic database environment is being continuously updated with receiving increments. Figure 23 (a) illustrates this environment where the original part is denoted as D and the incremental part is denoted as d and D∪d brings out the updated database. In order to hide sensitive itemsets in the updated database either the transactions in the incremental part d or transactions in the whole updated database D ∪d need to be modified. Modifying only the transactions in the incremental part of the database is dealing with small number of transactions. Suppose there are three sensitive itemsets as in Figure 23 (b). The original database D is already sanitized because support of all these three sensitive itemsets are below their sensitive threshold where support of

“ad”, “cd” and “bd” are 28.6%,14.3% and 0% respectively. After the database D is updated with the incremental part d the support of “ad”, “cd” and “bd” becomes 50%, 30%, and 10% respectively and this reveals the need for sanitization process before the updated database is published. One possible approach is to modify transactions in only incremental part d and the other possible approach is to modify transactions in the updated database. The first approach is adventurous in terms of execution time and resource allocation because there will be less transactions in the search space (transactions = {8,9,10}) compared to the second approach. However, the first approach may bring out more side effects such as loss of non-sensitive information and total number of item removal because all potential sensitive transactions will not be in the search space. On the other hand, the second approach is more advantageous in terms of side effects when compared to first approach but it will be inefficient in terms of execution time and resource allocation. However, it is possible to reduce these inefficiencies of the second approach by using an appropriate data structure.

The DynamicPGBS algorithm uses a Transaction Oriented Pseudo Graph (TPG) data structure based on the PG used in PGBS. Unlike the PGBS algorithm the DynamicPGBS algorithm only puts the sensitive transactions in to the Pseudo Graph data structure.



(a) Dynamic database (b) Sensitive itemsets

Figure 23. Sample database and sensitive itemsets with their sensitive thresholds.

4.4.2. Transaction Oriented Pseudo Graph (TPG)

The Transaction Oriented Pseudo Graph (TPG) data structure used in DynamicPGBS algorithm is very similar to the Pseudo Graph data structure used in PGBS. The only difference is the DynamicPGBS represents all sensitive transactions in the given database D rather than all transactions of D. The Pseudo Graph structure of DynamicPGBS is called TPG. It is assumed that items in each transaction are sorted in alphabetical or numerical order before they are converted to TPG. The algorithm for creating the TPG of a given database D is depicted in Algorithm 5. First each transaction in database D is checked to find out whether it is sensitive or not. Next vertices representing each item of a given sensitive transaction are connected sequentially labelled with the transaction ids.

The conversion process of the database updated database D ∪ d into TPG is illustrated in Figure 10. Suppose the updated dynamic database D ∪ dis given in Figure 23 (a) where all transactions in both D and d are alphabetically sorted. First each transaction is checked in sequential order to find out if it is sensitive and if so it is inserted into the TPG. The first sensitive transaction in D is “ade” with transaction id 5. Vertices labelled with “a”, ”d” and “e” are created and connected with each other with edge 5as in Figure 24 (a). The next sensitive transaction in D is “acd”, because there is no vertex created for item “c” a vertex labelled with “c” is created and then vertices “a”, “c” and “d” are connected sequentially with edge labelled with 5 as in Figure 24 (b). After all remaining sensitive transactions in D are inserted into the TPG the resulting TPG is shown in Figure 24 (c).

Calculating support count of sensitive itemsets is carried with same way as in PGBS. However unlike PG the TPG only represents the sensitive transactions because of this fact the support count of items that are generated from the TPG only represents number of appearances of items in sensitive transactions.

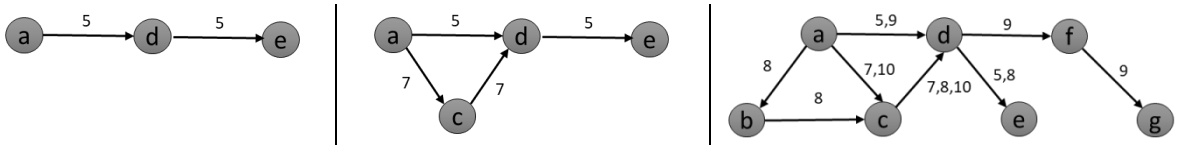
4.4.3. DynamicPGBS Algorithm

In dynamic database environment after each time the database is updated with a new batch of incoming transactions the state of a sensitive itemsets may change i.e. support of sensitive itemsets may exceed their sensitive thresholds, support of sensitive

Algorithm 5: Creating Pseudo Graph

Input: Transactional Database D
Output: Pseudo Graph PG

```
1  foreach sensitive transaction tr ∈ D do
2      foreach item j ∈ tr do
3          if PG does not contain v_j then
4              create v_j
5          end
6      end
7      foreach item i ∈ tr do
           // if there is more than one item in transaction tr
           and there exist a successor of the item at position i
8          if tr[i+1]≠null then
9              put an outgoing edge from v_tr[i] to v_tr[i+1] labeled with
                transaction id of tr
10         else
               // create a self loop if there is only one item in
               transaction tr
11             put an outgoing edge from v_tr[i] to v_tr[i] labeled with
                transaction id of tr
12         end
13     end
14 end
```



(a) TPG after transaction "ade" is inserted

(b) TPG after transaction acd is inserted

(c) TPG after all transactions in D ∪ d are inserted

Figure 24. Inserting transactions into TPG.

itemsets may fall behind their sensitive thresholds. The biggest problem in dynamic frequent itemset hiding problem is, while state of each sensitive itemset varies they should be hidden from the database with minimum execution time and side effects. These side effects include amount of non-sensitive knowledge loss, amount of data modified and total memory requirement. In order to speed up the execution time and minimize the resource allocation the Transaction oriented Pseudo Graph (TPG) data structure is employed. But unlike the PG as in PGBS, only sensitive transactions are put into the TPG structure. This is because first of all modification of non-sensitive transactions does not affect the support of any sensitive itemset and the second is the total number of vertices and edges in the graph can be reduced. Reducing the number of vertices and edges also reduces the total memory allocation of the TPG.

Before the database is published the DynamicPGBS algorithm deletes items called victims from predefined set of sensitive transactions for decreasing support of each sensitive itemset smaller than its sensitive threshold. None of the scan and transaction modification operations are performed on the actual database besides they are performed on the TPG. The information related to the sanitization operation is stored in Sanitization Table and whenever the database is published the sanitized database is generated from the copy of actual database by performing necessary modification.

The flowchart of DynamicPGBS algorithm is illustrated in Figure 25. Similar to the PGBS algorithm the DynamicPGBS algorithm takes database D, sensitive itemsets (SI) and their sensitive thresholds as input. First the database is converted to the PG, next if a new batch of transactions (d) arrives then the PG and database (D) are updated. Either after the conversion of D to the PG or d arrives if the database owner wants to release the database, the hiding process starts. The hiding process composes of creating the Sensitive Count Table (SCT) and creating the Sanitization Table (ST). After the hiding process is finished the sanitized database (D') is generated as output and all delete operations performed on PG are restored prior the sanitization process. The restore operation is performed by putting all victim and transaction pairs stored in ST into PG again and it is for utilizing all possible sensitive transaction modifications in the next sanitization process. Then the algorithm becomes ready to accept a new increment.

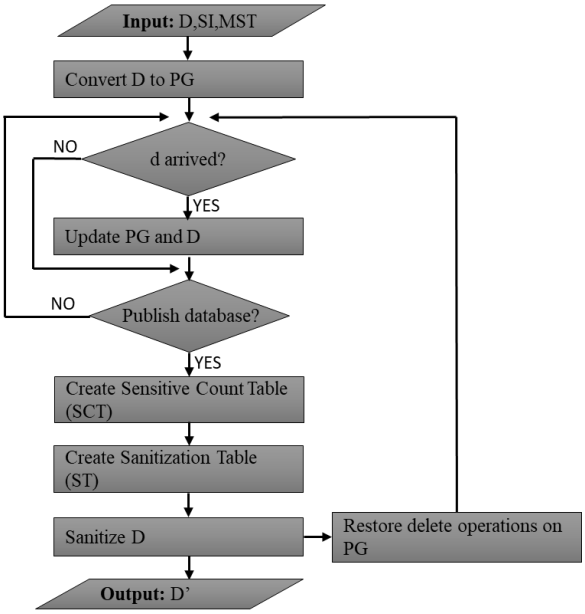


Figure 25. The flowchart of DynamicPGBS algorithm.

4.4.4. Creating Sensitive Count Table

As in PGBS algorithm Sensitive Count Table (SCT) represent the minimum number of support count decrease required to hide each sensitive itemset. The SCT has three attributes, SID, SI and N_{Modify}. The SID is the unique identifier of records in SCT, SI is the sensitive itemset and N_{Moidfy} is the minimum number of support count decrease to hide a given sensitive itemset. The N_{modify} is calculated with equation (1).

For illustration in Table 6 the Sensitive Count Table is shown for the Transaction Oriented Pseudo Graph (TPG) given in Figure 24 (c) and corresponding sensitive itemsets with their sensitive thresholds given in Figure 23 (b). After the SCT is created it is sorted according to the N_{Modify} column, this is for starting the sanitization process from sensitive itemset having maximum support decrease need.

Table 6. Sensitive Count Table (SCT) of DynamicPGBS algorithm.

SID	**SI**	**N_{Modify}**
0	ad	3
1	cd	2
2	bd	1

4.4.5. Creating the Sanitization Table

The process of creating the Sanitization Table consists of creating the sanitization Table (ST) and updating TPG. The Sanitization Table (ST) keeps the final modification information that will be applied to the database before it is published. As in PGBS the ST stores the victim and transaction pairs that represents the final modification information. The algorithm for creating the Sanitization Table is depicted in Algorithm 6. The sanitization process starts from the SI having maximum N_{Modify} value and also if N_{Modify} of a sensitive itemset is less than or equal to zero it indicates that the sensitive itemset is already hidden. In step 1 the first row of the SCT is assigned to the variable r_1 because it has the maximum N_{Modify}. Then the victim item is selected among items of the r_1.SI that has the maximum conflict degree, if there is more than one item having the same conflict degree then the victim item is selected with the highest support in TPG and if there is still more than one victim item then a random item is selected. The *unifiedItemsets* variable

stores sensitive itemsets of SCT and contain the victim item. The *sensitiveTransactions* is the set of sensitive transactions containing the *unifiedItemsets* and the number of transaction ids stored in *sensitiveTransactions* does not exceed the $r_1.N_{Modify}$. The TPG is updated by removing victim item from sensitiveTransactions of PG and then the victim and *sensitiveTransactions*pair is inserted into the Sanitization Table (ST). Then if any sensitive itemset in SCT is a subset of *unifiedItemsets*its N_{Modify} value is decreased by the number of transaction ids stored in *sensitiveTransactions*.If any of the N_{Modify} value of a sensitive itemset becomes less than or equal to zero it is removed from *unifiedItemsets.* If the sensitive itemset stored in r_1 is still having N_{Modify} greater than zero, then the algorithm tries to find out different transactions by changing the *unifiedItemsets* with removing the sensitive itemset having least N_{Modify} value.

Algorithm 6: Creating Sanitization Table

Input: Itemset Oriented Pseudo Graph IPG, Sensitive Itemsets SI, Sensitive Thresholds
Output: Sanitization Table ST, Support Count Table SCT

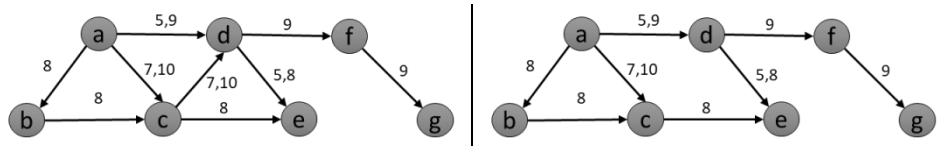
1 **while** N_{Modify} *value of the first row* $r_1 > 0$ *in SCT* **do**
2 **victim**←item that has the maximum cover degree in r1.SI or has the maximum support or selected randomly
3 **unifiedItemsets**← UnifiedItemsets ∪ SI for all SI ∈ SCT where $N_{Modify} > 0$ and victim ⊆ SI
4 **sensitiveTransactions**←transaction ids containing USI in PG
// number of transactions < r1.N_{Modify}
5 Update PG by removing *victim* from **sensitiveTransactions**
6 Insert **victim** and **transactions** pairs into ST
7 **foreach** *row* $r \in SCT$ **do**
8 **if** $r.SI \subseteq USI$ **then**
9 $r.N_{Modify}$←$r.N_{Modify}$ - |sensitiveTransactions|
10 **end**
11 **if** $r.N_{Modify} \leq 0$ **then**
12 USI ← USI - r.SI
13 Recalculate cover degree of items
14 **end**
15 **end**
16 **if** $r_1.N_{Modify} > 0$ **then**
17 Remove the sensitive itemset from unifiedItemsets which has the least N_{Modify} and go back to step 5
18 **else**
19 Sort SCT in decreasing order of N_{Modify}
20 **end**
21 **end**

4.4.6. Illustrating Example

To illustrate how the algorithm depicted in Algorithm 6 works suppose the Sensitive Count Table (SCT) is given in Table 6 and TPG is given in Figure 24 (c) as input. The first row stored in SCT is “ad” and the sanitization process starts from this sensitive itemset (line 1). The victim item is selected as “d” (line 2) among items in “ad” and put into variable *victim* because it has the maximum conflict degree (conflict degree item “d” = 3). Then all sensitive itemsets in SCT are unified and assigned to the variable *unfiedItemsets* because each of them contain the item “d”. The *unfiedItemsets* (line 3) becomes “abcd” and according to TPG in Figure 24 (c) only the transaction with id {8} contains this itemsets. The transaction id {8} is assigned to the *sensitiveTransactions* variable (line 4) and the TPG is updated by removing item “d” from transaction {8} (line 5) as shown in Figure 26 (a). The victim “d” and transaction {8} pair is added to the Sanitization Table (ST) (line 6) and N_{Modify} of each sensitive itemset in SCT is decreased by 1 (lines 7-10). After SCT is updated N_{Modify} of “bd” becomes zero so this means it is sanitized and will be neglected while calculating the conflict degrees of items and *unfiedItemsets* variable in the next iteration. The algorithm again selects the victim item as “d” (line 2) because it has the maximum conflict degree (conflict degree “d” = 2). The *unifiedItemsets* becomes “acd” (line 3) and according to the TPG in Figure 26 (a) transactions {7} and {10} contain it (line 4). The victim is removed from transactions {7} and {10} as shown in Figure 26 (b) (line 5) and the victim “d” and transactions {7,10} pair is inserted to the ST(line 6). The new N_{Modify} of “ad” and “cd” becomes 0 and -1 respectively so the hiding process terminates. After the ST is created a new copy of the original database D is created and the modification information stored in ST is applied to this copy. Then the TPG is recovered by using the ST and all records stored in ST are deleted and the algorithm becomes ready to accept new increments.

4.4.7. Sanitizing the Database

The sanitization process of DynamicPGBS creates a copy of the original updated database (D ∪ d)and then applies the sanitization solution stored in Sanitization Table (ST) to this copy. The whole sanitization process after an incremental part d is arrived to



(a) TPG after item “d” is deleted from transaction {8}

(b) TPG after item “d” is deleted from transactions {7,10}

Figure 26. Updating TPG with deleting items.

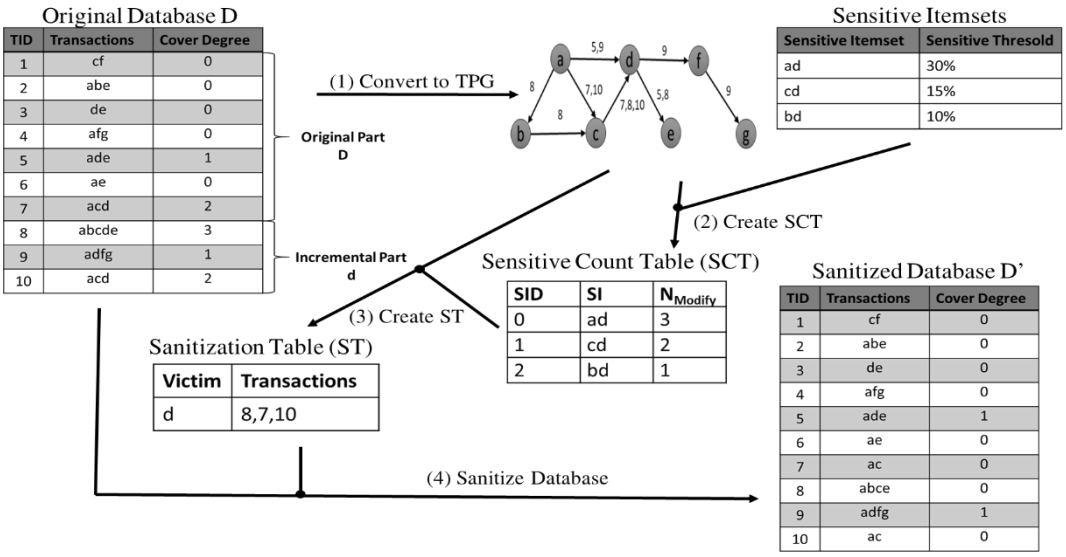


Figure 27. The flow of the processes in DynamicPGBS algorithm.

the original database D is illustrated in Figure 27. In this figure first the given updated database D ∪ d is converted to the Transaction oriented Pseudo Graph (TPG) representation, second the Sensitive Count Table (SCT) is created with using the TPG and the previously defined sensitive itemsets. Third the Sanitization Table (ST) is created with using the TPG and information stored in SCT. Finally a copy of the database D ∪ d is created and the information stored in STis applied to this copy for producing the sanitized database D’.

CHAPTER 5

PERFORMANCE EVALUATION

This chapter presents experimental results undertaken to empirically validate proposed frequent itemsets hiding algorithms; PGBS, IPGBS and DynamicPGBS. The experiments are performed on both real and synthetic databases by varying different attributes of the databases. The evaluation results of PGBS and IPGBS are represented together because both are developed for static environment whereas the evaluation results of DynamicPGBS are presented in separately because it is designed for dynamic environment.

All the experiments are conducted on a computer with Intel core i7-5500 2.4 GHZ processor and 8GB of RAM running on a Windows 10 operating system. In all test runs it is ensured that the system state is similar and gives close results when repeated. The algorithms are implemented in Visual Studio .NET C# 2015 Ultimate Edition.

5.1. Databases

The PGBS, IPGBS and DynamicPGBS algorithms are evaluated both on real and synthetic databases. The real databases used in the performance evaluations are Connect, and Retail where the Connect is obtained from UCI database repository [51] and the Retail is obtained from [52]. The Connect database contains all legal 8-ply positions of connect-4 game where none of the player has won yet and the next move is not forced. The Retail database contains anonymous market basket data from a Belgian retail supermarket store. Also two synthetic databases; SyntheticSparse and SyntheticDense with different characteristics are generated by using the IBM quest data generator [53].

The characteristics of all databases in terms of size of database, number of distinct items, average transaction length, shortest and longest transaction length and density are given in Table 7. The ratio of average transaction length to number of distict items is denoted as density. Density of a database indicates whether a given database is dense or sparse. As the density of a given database increases the correlation between items

increases so the frequent itemsets generated from dense databases are usually long [43-45].

Table 7. Characteristics of databases.

Name	Transactions	Distinct Items	Average Length	Shortest Length	Longest Length	Density (%)
Connect	67,557	129	43	43	43	33.4
Retail	88,162	16,470	10.3	2	77	0.0625
SyntheticDense	29,166	99	43.09	2	44	43.5
SyntheticSparse	28,417	9,479	11.48	2	11	0.1212

5.2. Frequent Itemset Hiding Algorithms in Static Environment

The sanitization algorithms that are analyzed in this section are Pseudo Graph Based Sanitization (PGBS), Itemset Oriented Pseudo Graph Based Sanitization (IPGBS) and Transaction Oriented Pseudo Graph (TGBS) algorithms. The TPGBS algorithm is similar to the Hiding Process of DynamicPGBS algorithm proposed in [36]. This algorithm employs same methodologies as in Dynamic PGBS algorithm. All these three algorithms enable to assign multiple sensitive thresholds and also they use pseudo graph data structure for representing the transactions of the database. The vertices in pseudo graph data structure of thePGBS algorithm represents all items of the original database, the TBGS algorithm represents only items of sensitive transactions and the IPGBS represents each different sensitive itemsets. The objective of all these three algorithms is to minimize the loss of non-sensitive knowledge with modifying minimum number of transactions.

For each database a set of 10 to 100 sensitive itemsets are selected and different sensitive thresholds are assigned. The sensitive itemsets of each database is determined as follows; first the set of frequent itemsets from each database is generated with predefined minimum support thresholds, next this set of frequent itemsets in each database is partitioned into 5 support bins where each bin contains nearly the same number of itemsets. As the last step 2 to 20 itemsets are randomly selected from each bin as sensitive itemset and the sensitive threshold of each sensitive itemset is assigned as the corresponding support of the bin. The support bin of each database is shown in Table 8.

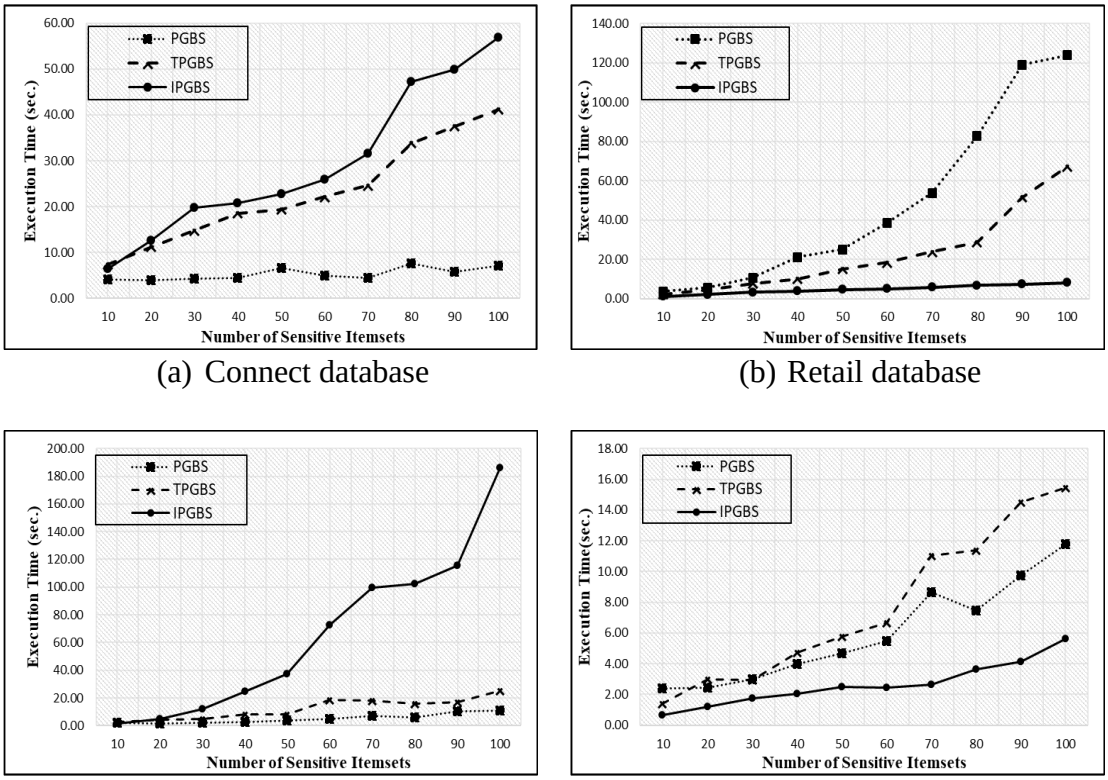
Table 8. Support bins of the databases in static environment.

	Connect	**Retail**	**SyntheticDense**	**SyntheticSparse**
Bin	**Support Range**	**Support Range**	**Support Range**	**Support Range**
1	(0.85, 0.857]	(0.0001, 0.00011]	(0.3, 0.308]	(0.0002, 0.000024]
2	(0.8576, 0.8672]	(0.00011, 0.00013]	(0.308, 0.3185]	(0.00024, 0.00028]
3	(0.8673, 0.8792]	(0.00013, 0.0017]	(0.3185, 0.3339]	(0.00028, 0.00035]
4	(0.8793, 0.8985]	(0.00017, 0.00026]	(0.3339, 0.3619]	(0.00035, 0.00049]
5	(0.8986, 0.9987]	(0.00026, 0.05072]	(0.3619, 0.9546]	(0.00049, 0.038]

The basic aim of the performance evaluation is to observe how the number of sensitive itemsets affects the performance of the algorithms. The performance of all algorithms is evaluated with respect to execution time, distance, information loss, accuracy and total memory consumption. The time need for performing the sanitization process by each algorithm is considered as execution time. The information loss, distance and accuracy are considered as side effects caused by the sanitization algorithms. The total memory allocation is considered as the total memory allocated by each algorithm for performing the sanitization process.

5.2.1. Execution Time

Execution time of PGBS, TPGBS and IPGBS algorithms on Connect, Retail, SyntheticDense and SyntheticSparse algorithms are shown in Figure 28 (a), (b), (c) and (d) respectively. It can be observed that PGBS algorithm has the least execution time on dense databases; Connect and SyntheticDense and the TGPS has the second lowest execution time. This is because both IPGBS and TBGS algorithms has execution time overhead while converting the database to IPG and TPG respectively. This overhead comes from analyzing the contents of each transaction in the database whereas this is not the case in PGBS, the PGBS algorithm directly converts all transactions in the database to Pseudo Graph (PG) without checking their contents. Dense databases contain too much distinct items and checking each item of all transactions increases the execution time of IPGBS and TPGBS. Figure 28 (c) and (d) show the execution time of algorithms on sparse databases. It can be inferred that the IPGBS has the least execution time on sparse databases and the TPGBS has the second lowest execution time. This is because compared to dense databases sparse databases have few different items.



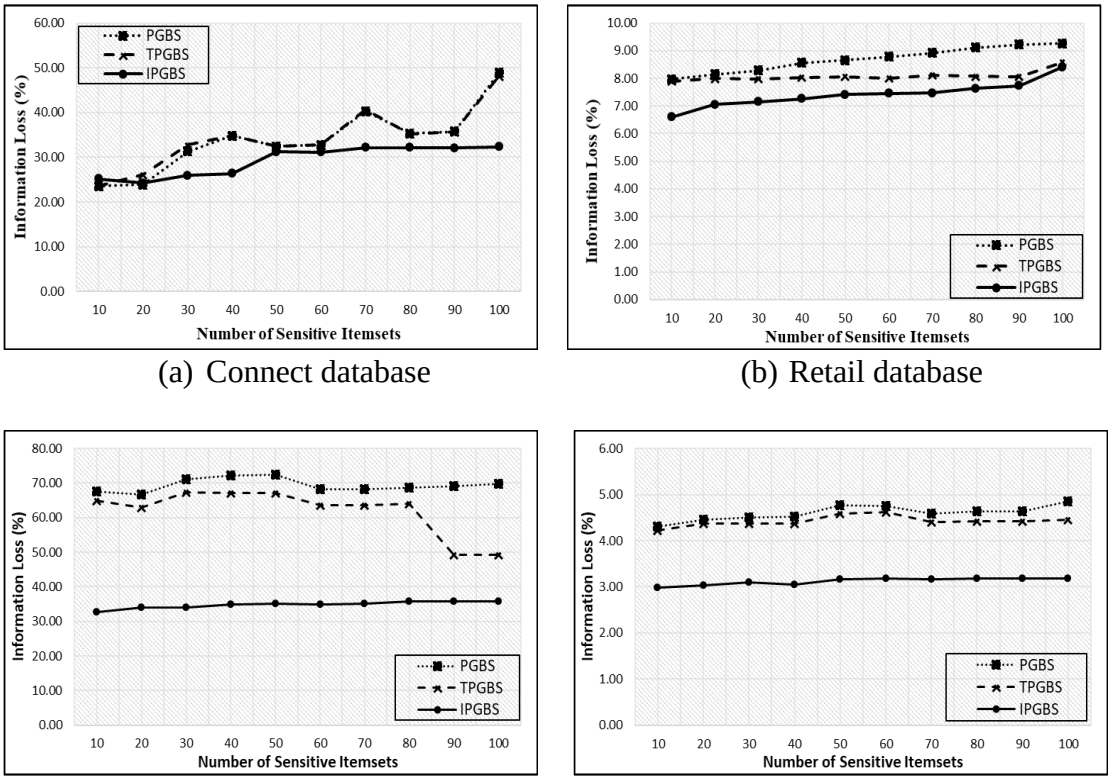
(a) Connect database (b) Retail database

(c) SyntheticDense database (d) SyntheticSparse database

Figure 28. Execution time varying the number of sensitive itemsets.

5.2.2. Information Loss

The information loss is the metric for showing the amount of non-sensitive frequent itemsets lost during the sanitization process. Figure 29 (a), (b), (c) and (d) shows information loss in percentage. It can be seen that the IPGBS algorithm has the least information loss on all databases. This is because the aim of the IPGBS algorithm is to modify least number of transactions during the sanitization process with modifying transactions containing maximum number of non-sanitized sensitive itemsets. As the total number of transaction modification decreases the information loss decreases at the same time. While the PGBS algorithm cause nearly the same information loss on Connect database with TPGS algorithm, the PGBS algorithm has the worst information loss on Retail, SyntheticDense and SyntheticSparse databases. Also it should be noted that the amount of information loss on dense databases Connect and SyntheticDense is higher than the information loss on sparse databases Retail and SyntheticSparse for all three algorithms.



(a) Connect database (b) Retail database

(c) SyntheticDense database (d) SyntheticSparse database

Figure 29. Information loss varying the number of sensitive itemsets.

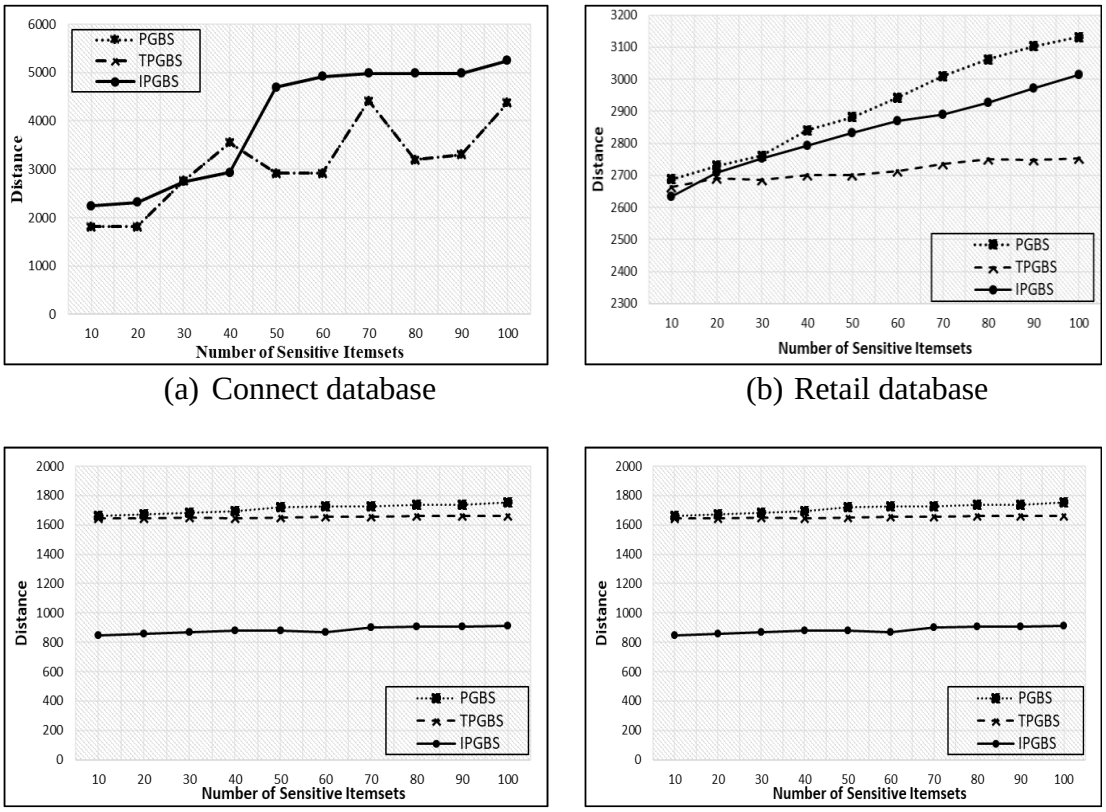
5.2.3. Distance

The distance metric shows the total number of items deleted from the database during the sanitization process and it is shown in Figure 30 (a), (b), (c) and (d). On Connect database the PGBS and the TPGBS algorithms have the same and lowest distance. On Retail database the TPGBS has the lowest distance while the PGBS has the highest distance. The IPGBS algorithm has the lowest distance on SyntheticDense and SyntheticSparse databases and the PGBS has the highest distance on sparse databases SyntheticDense and Retail.

5.2.4. Accuracy Loss

The accuracy loss metric indicates the total number of different transactions modified during the sanitization process and it is shown in percentage in Figure31 (a),

(b), (c) and (d). from Figure 31 (a) and (c) it can be inferred that the IPGBS algorithm has the minimum accuracy loss on dense databases. Also it is clear that there is no relation



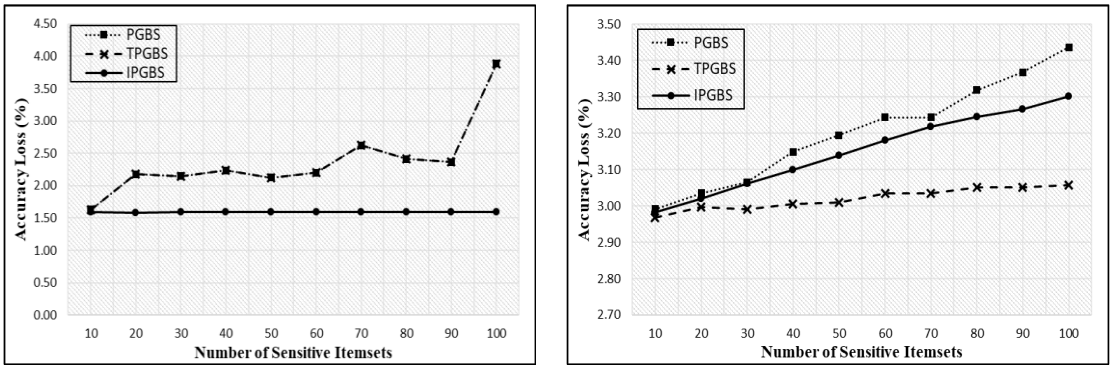
(a) Connect database (b) Retail database

(c) SyntheticDense database (d) SyntheticSparse database

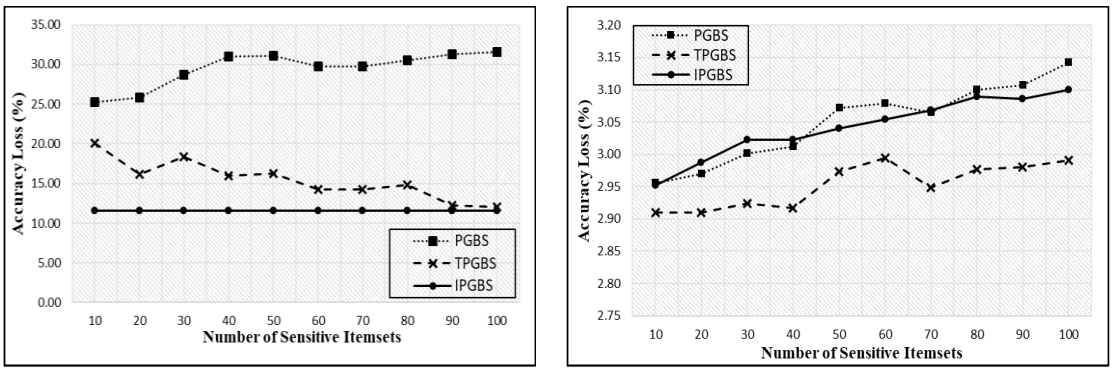
Figure 30. Distance varying the number of sensitive itemsets.

between the number of sensitive itemsets and accuracy loss for the IPGBS algorithm on dense databases while this is not the case for both PGBS and TPGBS algorithms. On Connect database the PGBS and the TPGBS have the same accuracy loss and it increases with the number of sensitive itemsets. On SyntheticDense database the PGBS algorithm has the maximum accuracy loss while the TPGBS has the second highest accuracy loss and also the accuracy loss of TPGBS is inverse proportional to the number of sensitive itemsets.

Figure 31 (b) and (d) show accuracy loss for sparse databases Retail and SyntheticSparse. From these two figures it is clear that PGBS algorithm has the maximum accuracy loss on sparse databases, the IPGBS has the second highest accuracy loss and it is proportional to number of sensitive itemsets.



(a) Connect database (b) Retail database



(c) SyntheticDense database (d) SyntheticSparse database

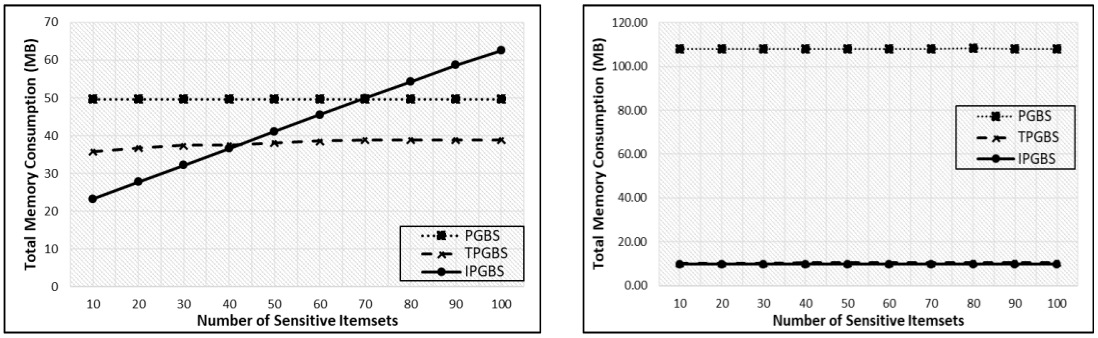
Figure 31. Accuracy loss varying the number of sensitive itemsets.

5.2.5. Memory Consumption

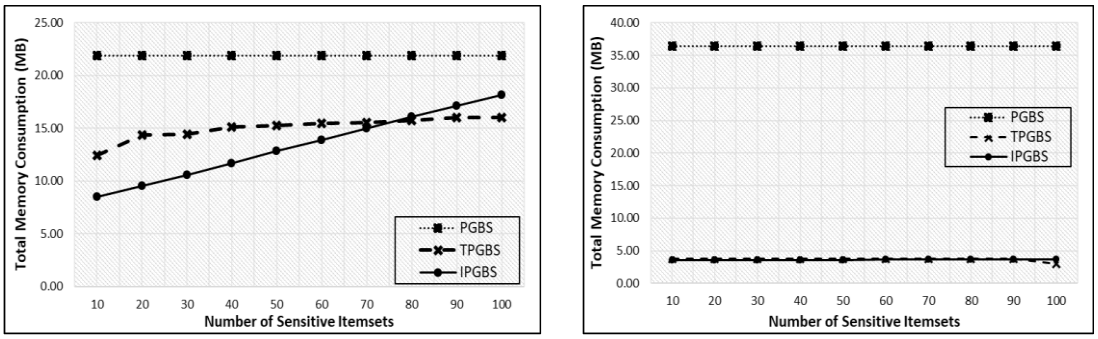
The total memory allocation in megabytes (MB) of each algorithm; PGBS, IPGBS and TPGBS are shown in Figure 32 (a), (b), (c) and (d). It can be seen that memory consumption of IPGBS algorithm is proportional to the number of sensitive itemsets on dense databases Connect and SyntheticDense. This is because sensitive itemsets on dense databases have higher support compared to sparse databases and as the IPGS algorithm stores information directly related to the sensitive itemsets and transactions they are contained in the memory consumption increasing with the number of sensitive itemsets. On sparse databases both PGBS and the IPGBS algorithms have the minimum memory allocation while the TPGBS has the maximum memory allocation.

5.2.6. Discussion of the Results

The PGBS algorithm converts each item in every transaction of database D to



(a) Connect database (b) Retail database



(c) SyntheticDense database (d) SyntheticSparse database

Figure 32. Total memory consumption varying the number of sensitive itemsets.

vertex of Pseudo Graph (PG) without checking the content of the transactions. Besides the IPGBS and TPGBS algorithms first check the content of each transaction and if the transaction contains any sensitive itemset then they convert the transaction either to Transaction Oriented Pseudo Graph (TPG) or Itemset Oriented Pseudo Graph (IPG). Because of this transaction content checking the PGBS algorithm is more advantageous in terms of execution time on dense databases because number of different items in dense databases are higher than sparse database so the number of item checking increases for both IPGBS and TPGBS algorithms. On sparse databases the IPGBS has the lowest execution time because support of itemsets in sparse databases is smaller than dense database consequently the number of sensitive transactions in sparse database is smaller than dense database. As the number of sensitive transactions decrease the iteration number of transaction content checking decreases.

The IPGS algorithm has the minimum information loss on both dense and sparse databases because it tries to modify transactions containing maximum number of non-sanitized sensitive itemsets. Besides both PGBS and TPGBS algorithms modify transactions containing maximum number of sensitive itemsets without checking they are

already sanitized or not. There is not direct relation between the distance and information loss metric. As the density of a given database increases the support of sensitive itemsets increases and this results in more execution time overhead during the Pseudo Graph data structure creation for both IPGBS and TPGBS algorithms. When the density of a given database is sparse the execution time of IPGBS increases proportional to the number of sensitive itemsets and it has the minimum execution time.

In the performance evaluation the loss of non-sensitive knowledge is measured with the *information loss* metric and as it can be seen in the second subsection the IPGBS has the minimum information loss compared to TPGBS and PGBS for all databases. The memory allocation of IPGBS is better than both PGBS and TPGBS algorithms when the given database is sparse. Also the IPGBS has the best memory allocation score on Connect and Chess databases till 50 sensitive itemsets are *sanitized* and has the best memory allocation score on SyntheticDense database till 80 sensitive itemsets are sanitized.

There is no direct relation between density of the database and distance metric. The distance directly depends on the set of sensitive itemsets. As the amount of common items in a given set of sensitive itemsets increases the distance is going to decrease because for all algorithms it is possible to sanitize more than one sensitive itemsets at once by deleting a single item from transactions.

5.3. Frequent Itemset Hiding Algorithms in Dynamic Environment

In this section the performance of the proposed dynamic environment frequent itemsets hiding algorithm DynamicPGBS is evaluated with using two similar counterparts SPITF and RHID algorithms. Both SPITF and RHID algorithms are designed for the dynamic environment as DynamicPGBS but they differ in sanitization methodologies. The SPITF algorithm performs the sanitization opetarion on the whole updated database (D ∪d) where D is the original database and the d is the incremental part. The RHID algorithm performs the sanitization operation only in the incremental part d and then combines D with d’ where d’ is the sanitized incremental part. As in DynamicPGBS algorithm both SPITF and RHID allows to assign multiple sensitive thresholds for sensitive itemsets.

The experiments are conducted on 2 real databases Connect and Retail and two synthetic databases SytntheticSparse and SyntheticDense. The main purpose of the experiments is to evaluate the increment size on performance of each algorithm. So the experiments are conducted with fixed sized sensitive itemsets while varying the size of the arriving increments. 10 sensitive itemsets are selected randomly with using the support bins as in the previous section. The support bins of each database is given in Table 9, and from each support bin, 2 itemsets are selected as sensitive. The algorithms are compared for 10 different increments in each database where the increment sizes are varied from 10% to 100% of the original database. The performance of algorithms is evaluated with respect to execution time, information loss, distance and total memory allocation.

Table 9. Support bins of the databases in dynamic environment.

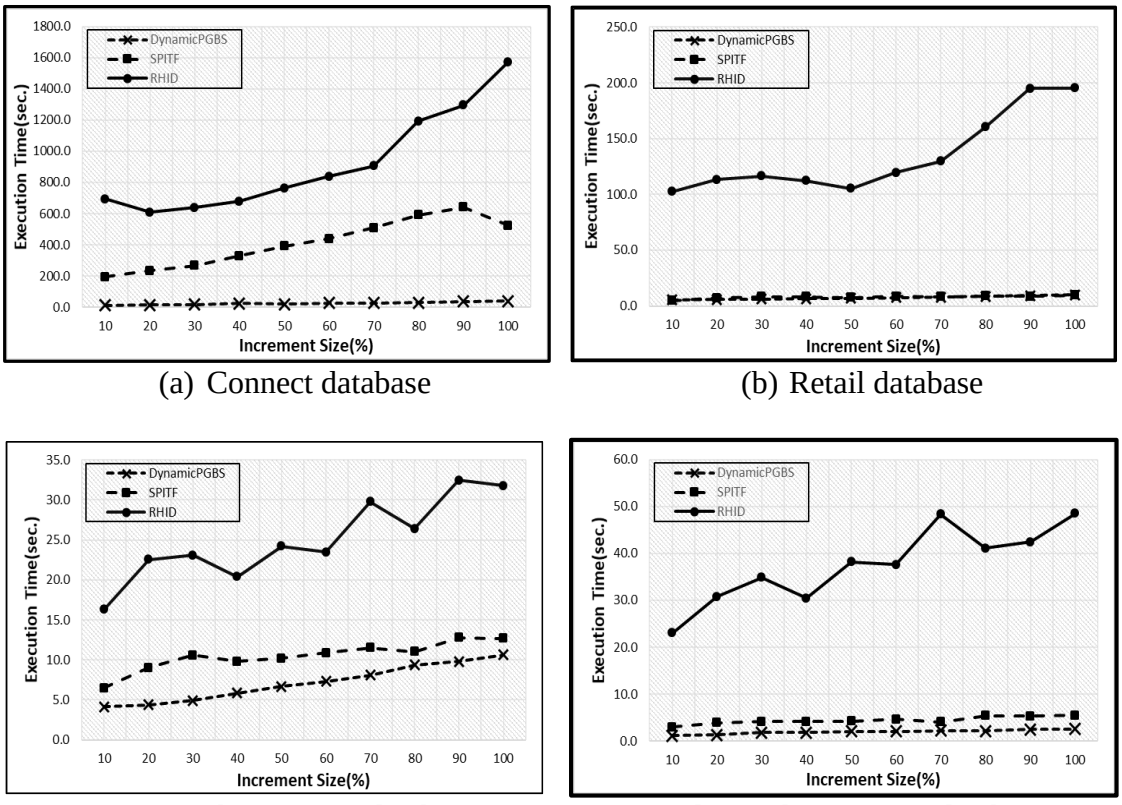
	Connect	**Retail**	**SyntheticDense**	**SyntheticSparse**
Bin	**Support Range(%)**	**Support Range(%)**	**Support Range(%)**	**Support Range(%)**
1	(85, 85.7]	(0.1, 0.118]	(30, 30.8]	(0.5, 0.544]
2	(85.76, 86.72]	(0.12, 0.142]	(30.8, 31.85]	(0.544, 0.6]
3	(86.73, 87.92]	(0.144, 0.185]	(31.85, 33.39]	(0.6, 0.709]
4	(87.93, 89.85]	(0.186, 0.287]	(33.39, 36.19]	(0.709, 0.935]
5	(89.86, 99.87]	(0.288, 5.072]	(36.19, 95.46]	(0.935, 3.8]

During the performance evaluation it was noticed that the SPITF algorithm is unable hide all given sensitive itemsets on dense databases Connect and SyntheticDense while both DynamicPGBS and RHID achieved hiding all given sensitive itemsets. The amount of sensitive itemsets failed to be hidden is represented with the hiding failure metric. The hiding failure of SPITF algorithm on Connect and SyntheticDense databases are shown in Figure 33.

5.3.1. Execution Time

The execution time of *DynamicPGBS*, SPITF and RHID are given in Figure 34. on sparse databases as in Figure 34 (c) and (d) the execution time of both SPITF and DynamicPGBS does not change linearly with the amount of new transactions added on

the original database whereas this is not the case for the RHID algorithm. This is because as sparse databases have generally short sized frequent itemsets and the sensitive itemsets are selected among them, the size of the sensitive itemsets in sparse databases have small size compared to sensitive itemsets of dense databases. As the size of a sensitive itemset decreases, the execution time for uncovering transactions containing this itemset in the data structures of both *DynamicPGBS* and SPITF decreases. For all databases given in Figure 34 (a), (b), (c) and (d) the execution time of *DynamicPBGS*is less than SPITF and RHID algorithms and also the RHID algorithm has the highest execution time on all databases.



(a) Connect database (b) Retail database

(c) SyntheticDense database (d) SyntheticSparse database

Figure 34. Execution time varying increment size.

5.3.2. Information Loss

Figure 35 shows the experimental results of Information Loss. The results indicate that SPITF causes minimum Information Loss on dense databases Connect and SyntheticDense. This is because the SPITF is not able to conceal all given sensitive

itemsets in these two databases. On the other hand the DynamicPGBS algorithm achieves the minimum Information Loss on sparse databases; Retail and SyntheticSparse.

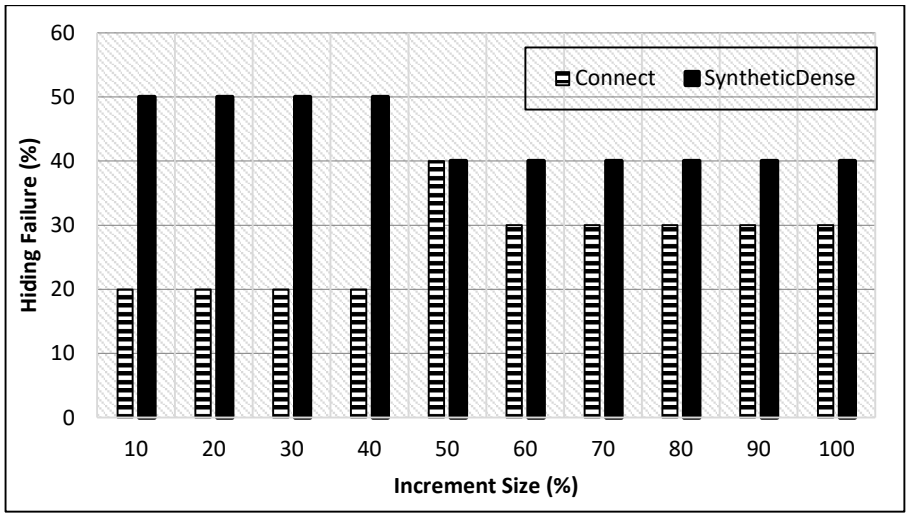
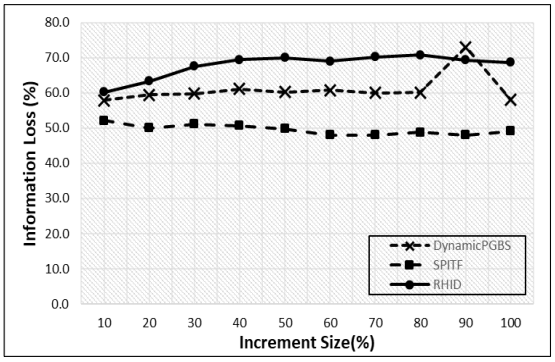
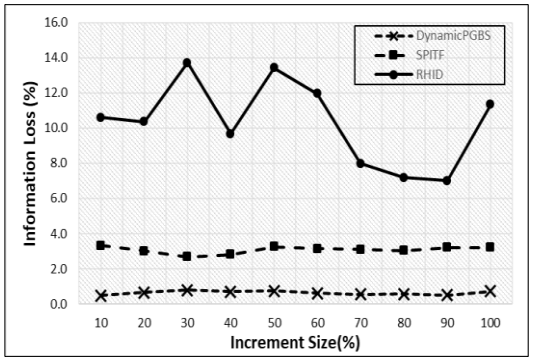


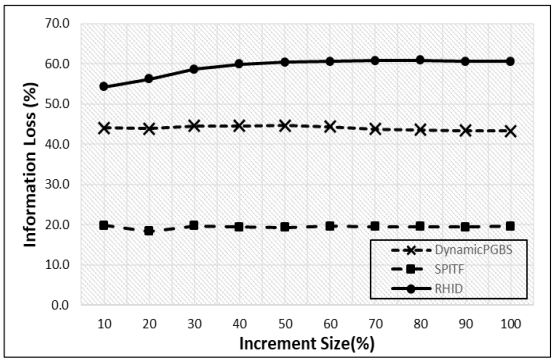
Figure 33. Hiding failure of SPITF algorithm varying the increment size.



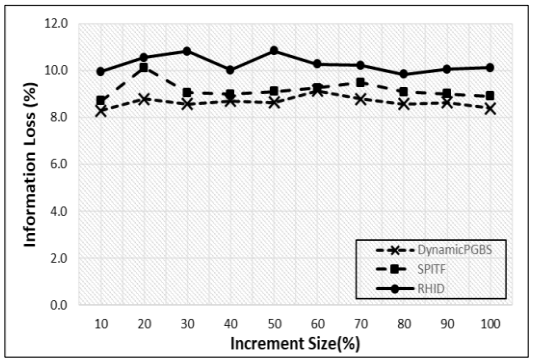
(a) Connect database



(b) Retail database



(c) SyntheticDense database



(d) SyntheticSparse database

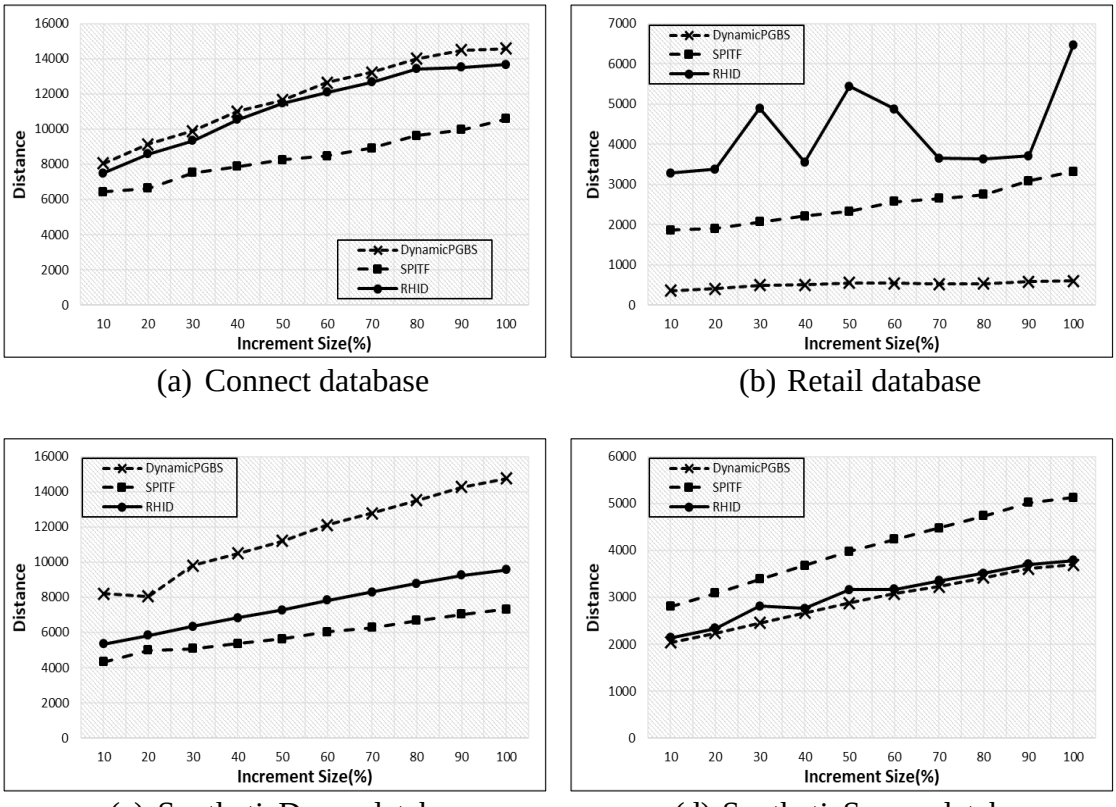
Figure 35. Information loss varying increment size.

5.3.3. Distance

In Figure 36 the experimental results for the distance are shown. As can be seen in Figure 36 (a) and (b) the SPITF has the lowest distance on dense databases Connect and SyntheticDense where as in Figure 36 (c) and (d) the DynamicPGBS has the lowest distance on sparse databases. The small distance of SPITF may due to fact that it is unable to hide all sensitive itemsets on dense databases and as a result less than necessary number of transaction modification causes less item removal.

5.3.4. Accuracy Loss

The accuracy loss of DynamicPGBS, SPITF and RHID are shown in Figure 37 (a), (b), (c) and (d). For dense databases Connect and SyntheticDense the SPITF algorithm has the lowest accuracy loss as shown in Figure 37 (a) and (c) whereas the DynamicPGBS algorithm has the lowest accuracy loss for sparse databases Retail and



(a) Connect database (b) Retail database

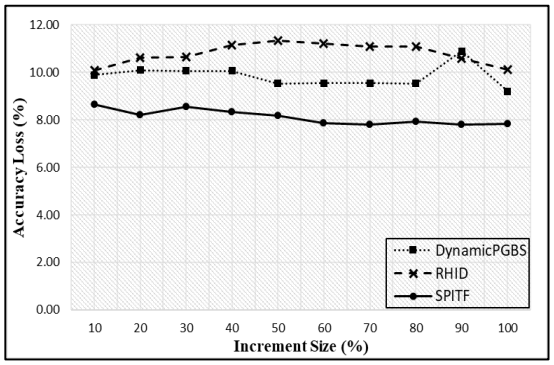
(c) SyntheticDense database (d) SyntheticSparse database

Figure 36. Distance varying increment size.

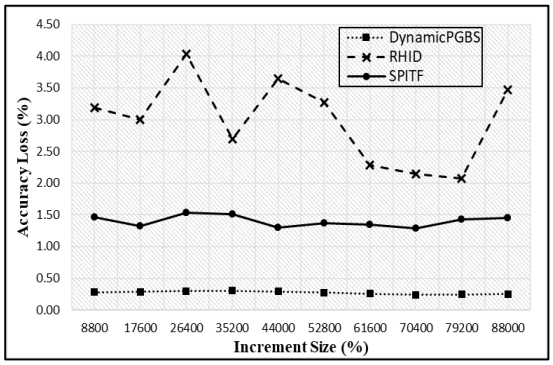
SyntheticSparse as shown in Figure 37 (b) and (d). The RHID algorithm has the maximum accuracy loss on three of the databases; Connect, Retail and SyntheticSparse. It can inferred from figures that there is no relation between the amount of the increment size and the accuracy loss.

5.3.5. Memory Consumption

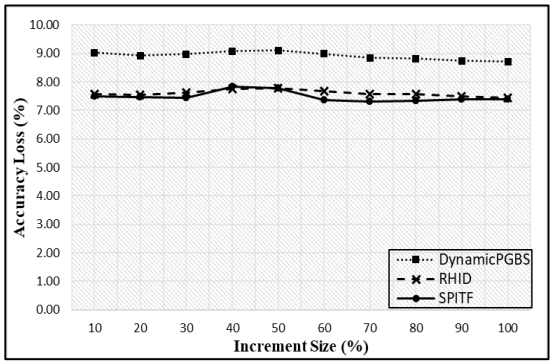
The total memory allocation in megabytes (MB) of each algorithm during the sanitization process is given in Figure 38. For dense databases Connect and SyntheticDense, the RHID algorithm requires the minimum amount of memory as shown in Figure 38 (a) and (c). The *DynamicPGBS* algorithm requires the minimum amount of memory on sparse databases Retail and SyntheticSparse as in Figure 38 (b) and (d). This is because the average transaction length of sparse databases is smaller than dense databases and also support of itemsets in sparse databases are low.



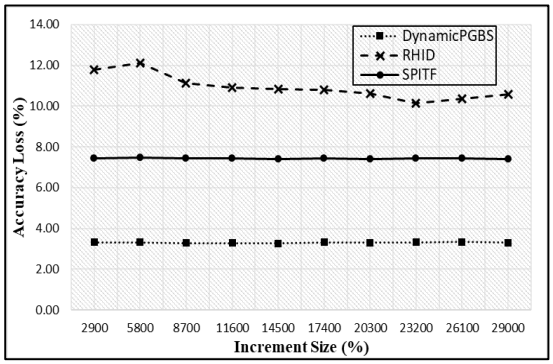
(a) Connect database



(b) Retail database



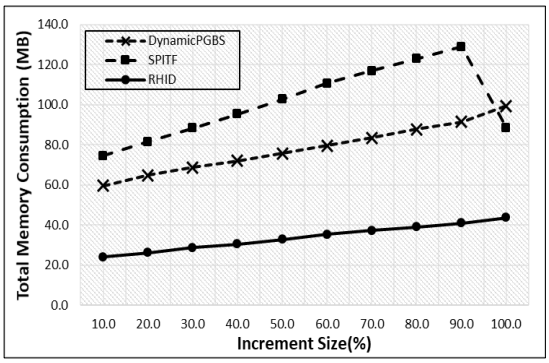
(c) SyntheticDense database



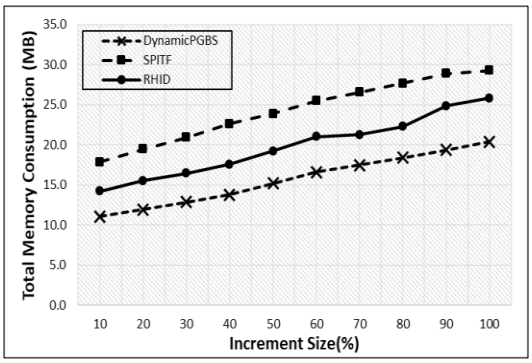
(d) SyntheticSparse database

Figure 37. Accuracy loss varying increment size.

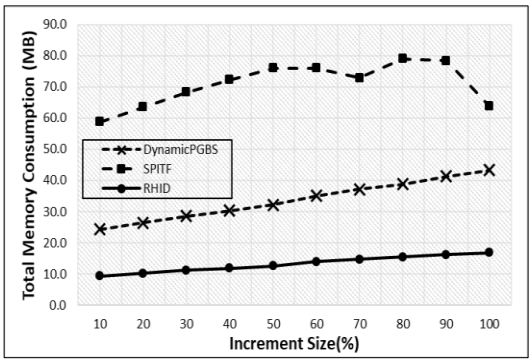
The memory requirement of the TPG data structure of DynamicPGBS decreases as there is small sized and low support valued itemsets. The SPITF algorithm requires the greatest amount of memory in all databases. This high memory consumption is due to SPITF's tree based internal data structure. This data structure represents items as vertices of the three and it does not prevent to represent the same item in more than one vertex. On the other hand in TPG each item is only represented once as vertex of the graph.



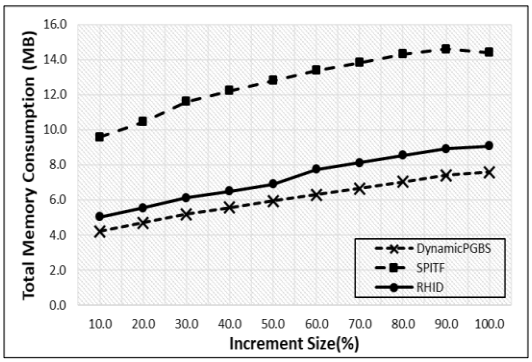
(e) Connect database



(f) Retail database



(g) SyntheticDense database



(h) SyntheticSparse database

Figure 38. Total memory consumption varying increment size.

5.3.6. Discussion of the Results

The SPITF algorithm represents each sensitive transaction of the updated database as a tree like data structure. Although this algorithm seems to be good in distance and information loss on dense databases it is unable to hide all given sensitive itemsets in the given database. Because of this fact it fails to modify sufficient quantity of transactions and as a result both information loss and distance seems to be smaller than DynamicPGBS and RHID algorithms on dense databases.

The RHID algorithm does not empty any data structure for representing the transactions and this causes smaller memory allocation on dense databases compared to DynamicPGBS and SPITF algorithms. But this small memory allocation comes with a tradeoff of high execution time compared to DynamicPGBS and SPITF algorithms.

The DynamicPGBS algorithm has the minimum distance and information loss results and also it has the second best results on dense databases. The execution time of DynamicPGBS is better than both SPITF and RHID algorithm on both sparse and dense databases and it can have inferred that the Transaction Based Pseudo Graph (TPG) data structure is advantageous in scan operations. The DynamicPGBS allocates the minimum memory although it employs a data structure to represent transactions. This is because it only converts the sensitive transactions into TPG.

CHAPTER 6

CONCLUSION

The privacy preserving frequent itemset mining problem is to conceal sensitive itemsets before the database is shared between third parties. This process can be handled by transforming the database into a new one that does not contain any sensitive itemsets. One of the transformation techniques is deleting items from the database till support of given sensitive itemsets are decreased below the predefined support thresholds. This type of hiding technique is called distortion based frequent itemset hiding. The optimal solution for distortion based frequent itemset hiding problem is to hide all given sensitive itemsets while at the same time preserve the data and non-sensitive knowledge quality in the published or modified database at maximum level.

The existing challenges of frequent itemset hiding algorithms are as follows. First in the literature there are a few algorithms that enable assigning multiple sensitive thresholds to sensitive itemsets. Decreasing support of every sensitive itemset under the same threshold may overprotect some of the sensitive itemsets. Also malicious user may infer the sanitized sensitive itemset if support of high and low support sensitive itemsets are decreased under the same support threshold. Second most of the databases are dynamic and they are continuously updated with arriving increments. In dynamic database environment two possible approaches can be followed for sanitizing the given database. The first approach conceals the sensitive itemsets after combining the incremental part with the original part. The second approach conceals the sensitive itemsets in only incremental part and then combines the sanitized incremental part with the already sanitized original part. Third preventing scan operations on the actual database increases the execution time efficiency. Appropriate data structures can be designed for representing the database and then the scan operations can be carried on this data structure.

In this thesis I have developed frequent itemset hiding algorithms for static and dynamic environment to find out optimal frequent itemset hiding solution. I proposed four different distortion based frequent itemset hiding algorithms where three of them are designed for static environment and one of them is designed for dynamic environment.

The algorithms designed for static environments are called PGBS (Pseudo Graph Based Sanitization), IPGBS (Itemset Oriented Pseudo Graph Based Sanitization) and TPGBS (Transaction Oriented Pseudo Graph Based Sanitization). The algorithm designed for dynamic environment is called DynamicPGBS. All of the four algorithms use different versions of pseudo graph data structure to hold the necessary information related with the input database in order to speed up sanitization process. Table 10 compares these proposed algorithms. The column "Transaction Selection" indicates the sensitive transactions uncovered for modification. The column "Main Objective" indicates the main focus of the algorithms. The "Environment" column shows the database environment the algorithm is designed for. The "Content of the vertex" column indicates what kind of data is stored in the vertex of the graph data structure of the algorithm. The "Content of the edge" column indicates what kind of transaction id is stored in the edges of the graph data structure of the algorithm. The "Support Count of Items" column indicates the possible support count of items that can be performed. The "MIS" column shows whether the algorithm enables assigning multiple sensitive thresholds for sensitive itemsets. The "Complexity of Creating Graph" column shows the computational complexity of creating the internal graph structure of each algorithm. The "Complexity of the Sanitization Process" indicates the computational complexity of sanitizing the database for each algorithm.

The vertices of both PG (Pseudo Graph) and TPG (Transaction Oriented Pseudo Graph) data structures represent items of the transactional database and edges connecting these vertices represent the transaction ids containing these items. The main difference between PG and TPG is while PG stores all items in each transaction the TPG only represents items in sensitive transactions. As PG represents the full transactional database it is possible to calculate support of item with using this data structure which is not the case in TPG. In TPG it is only possible to calculate support of items that exists in sensitive transactions. The TPG is advantageous in terms of memory requirement because it does not represent all transactions of the given database. The IPG (Itemset Oriented Pseudo Graph) data structure represents the sensitive itemsets as vertices and edges connecting these vertices represent the transaction ids containing these sensitive itemsets. In IPG the total number of vertices is equal to total number of sensitive itemsets. It is not possible to calculate support of items in the database or support of items that exists in the sensitive

Table 10. Comparison of proposed algorithms.

	PGBS	**IPGBS**	**TPGBS**	**DynamicPGBS**
Transaction Selection	Maximum cover degree transactions	Maximum cover degree transactions with non-sanitized sensitive itemsets	Maximum cover degree transactions	Maximum cover degree transactions
Main Objective	Execution time	Memory Information loss	Execution time	Execution time Memory
Environment	Static	Static	Static	Dynamic
Content of the Vertex of the Pseudo Graph	All items	Sensitive itemsets	Items of sensitive transactions	Items of sensitive transactions
Content of the Edge of the Pseudo Graph	Transaction ids of all items	Transaction ids of sensitive itemsets	Transaction ids of sensitive transactions	Transaction ids of sensitive transactions
Support Count of Items	In all transactions	None	Only for sensitive transactions	Only for sensitive transactions
MIS	✓	✓	✓	✓
Complexity of Creating Pseudo Graph	O(AvgL*|D|)	O(|SI|*|D|)	O(|D|*|V|)	O(|D|*|V|)
Complexity of Sanitization Process	O(|SI|*|V|)	O(|SI|*(|V|+|E|))	O(|SI|*|V|)	O(|SI|*|V|)

transactions. The IPG is designed for finding out transactions containing sensitive itemsets.

The PGBS algorithm starts modification from transactions containing maximum number of sensitive itemsets because in this way decreasing support of more than one sensitive itemset with only a single transaction modification is possible. Also the PGBS algorithm directly converts all transactions in the given database to PG data structure without checking their contents. This is for reducing execution time overhead in creation of the PG. The PGBS first converts all transactions of the given database D to Pseudo Graph (PG) form. In the worst case scenario each transaction contains different items and the intersection set of items in each transaction is empty, converting a given transaction tr into PG takes O(|tr|) time where |tr| is the number of items in the tr. Creating edges between vertices of PG takes O (|D|) where |D| is the database size. It should be apparent that the algorithm has a computational complexity O (AvgL*|D|) for creating the PG where the AvgL is the average length of transactions in D. The hiding process uncovers transactions from PG in O (|V|) computational complexity where |V| is the total number of vertices in PG. The uncovering process is repeated as the number of victim items so at worst case where none of the sensitive itemsets share a common item, the hiding process has O (|SI|*|V|) computational complexity where |SI| is the number of sensitive itemsets.

The main purpose of IPGBS algorithm is to achieve minimum amount of non-sensitive knowledge loss during the sanitization process. For this reason, it tries not to modify transactions containing any hidden sensitive itemsets. The IPGBS algorithm starts modification from transactions containing maximum number of non-sanitized sensitive itemsets. This is because as deleting items from transactions containing already sanitized sensitive might decrease the support of sensitive itemsets more than necessary. The internal data structure in IPGBS algorithm is called Itemset Oriented Graph (IPG). The IPG represents the relation between the sensitive itemsets and transactions. Each transaction in the given database is checked whether it contains any sensitive itemset and if so it is inserted into the IPG. Compared to PG, this process causes execution time increase in the IPG creation phase but it is compensated with low memory requirement because of reduced number of vertices. The IPGBS algorithm only represents the sensitive itemsets as Itemset Oriented Pseduo Graph (IPG). The algorithm checks contents of each transaction to identify whether it is sensitive or not. In the worst case scenario where each transaction in D is sensitive and every item in each transaction is a subset of at least one of the sensitive itemsets. Creating the Itemset Oriented Pseodo Graph (IPG) takes O(|SI|*|D|) where |SI| is the number of sensitive itemset and |D| is the

total number of transactions in D. The hiding process uncovers longest path from the IPG in a depth first search manner so identifying a path from a vertex takes (|V|+|E|) worst case computational complexity where |V| is the total number of vertices in IPG and |E| is the total number of edges in IPG. So identifying paths for all sensitive itemset takes O(|SI|*(V|+|E|)) where |SI| is the number of sensitive itemsets.

The TPGBS algorithm is the static version of the DynamicPGBS algorithm. Both TPGBS and DynamicPGBS algorithms employ the TPG data structure to represent items in each sensitive transaction. As in PGBS both DynamicPGBS and TPGBS algorithms try to modify transactions containing maximum number of sensitive itemsets. The DynamicPGBS and the TPGBS algorithms represent the sensitive transactions as Transaction Based Pseudo Graph (TPG). These two algorithms scan the database to uncover sensitive transactions and computational complexity of this process is O (|D|) where |D| is the size of the transactional database D. In the worst scenario each transaction contains different items and the intersection set of items in each transaction is empty, converting a given transaction to TPG takes O (|V|) where |V| is the total number of vertices in TPG. As a result, uncovering all sensitive transactions from the database D and converting them to TPG takes O (|D|*|V|) computational complexity. The hiding process is to discover a certain set of transactions from the TPG, discovering a transaction from the TPG has O (|V|) computational complexity and this process is iterated as the number of sensitive itemsets. In the worst case scenario where the sensitive itemsets do not contain any common item, the computational complexity of the hiding process is O (|SI|*|D|) where |SI| is the number of sensitive itemsets and |D| is the total number of transaction in database D.

As the database is being continuously updated in dynamic environment one of the problems in dynamic environment the memory requirement should be decreased. Because of this problem to reduce the memory requirement of the graph based data structure only items of the sensitive transactions are represented.

I evaluated the efficiency of all these four algorithms with resource allocation, execution time, information loss and data distortion metrics. I used four different transactional databases with different characteristics during the performance evaluation. These databases are grouped as dense and sparse according to their densities where the density of the database is a measure to evaluate the similarity of transactions in a given database. As the density increases similar transactions appears in a given database. For

all experiments that I performed in static and dynamic environment the independent variables are execution time, information loss, distance, accuracy loss and total memory allocation while the set of sensitive itemsets are the dependent variables. The sensitive itemsets are selected randomly in each experiment without restricting any characteristics. The characteristics of sensitive itemsets include the number of items the sensitive itemsets contain and the number of common items the sensitive itemsets mutually contain. Since in the performance evaluation all participant algorithms have similar properties selecting sensitive itemsets based on their characteristics would have the same effect on all algorithms. The databases employed in the performance evaluation are categorized according to their densities where two of the databases were sparse and two of them are dense. By this way I evaluated the side effects of each algorithm based on the density of the databases. In addition two of the databases were real and two of them were synthetic. The real databases were used for reflecting the reality and the synthetic databases were used for evaluating the performance with two different databases with different densities but similar characteristics.

The PGBS algorithm is advantageous in terms of execution time on dense databases as the PG data structure does not check the contents of each transaction. On the other hand, the IPGBS algorithm harms less non-sensitive knowledge during sanitization process which is the main purpose of this algorithm. The DynamicPGBS achieves minimum execution time on both sparse and dense databases and also it achieves minimum memory allocation on dense databases.

This research can be continued with some challenges left to explore as follows:

- The DynamicPGBS algorithm is designed for the dynamic environment where only database increments considered. The dynamicity of the database can be extended to include deletion of transactions in addition to the transaction insertion.
- The heuristic sanitization approaches does not give a general optimal hiding solution. These approaches are designed with considering the features of databases and the efficiency of them can vary according to sensitive itemset and database. The exact hiding solutions on the other hand are able to find a general optimal hiding solution but they have high of execution time and resource allocation. To reduce the side effects of the sanitization process the exact and heuristic approaches can be combined in a hybrid way.

REFERENCES

1. Mannila H.; Toivonen H. Levelwise search and borders of theories in knowledge discovery. Data Mining and Knowledge Discovery. **1997**, 3, 241–258.

2. Sun X.; Yu P. S. A border-based approach for hiding sensitive frequent itemsets. Proceedings of the Fifth IEEE International Conference on Data Mining, Houston, TX, USA, 27-30 November; 2005, 426–433.

3. Moustakides G.V.; Verykios V.S. A max–min approach for hiding frequent itemsets. In 6th IEEE International Conference on Data Mining, Hong Kong, China, 18-22 December; IEEE, 2006, 502–506.

4. Rusell S. ; Norving P. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2003; Vol.2.

5. Menon S.; Sarkar S.; Mukherjee S. Maximizing accuracy of shared databases when concealing sensitive patterns. Information Systems Research. **2005**, 3, 256–270.

6. Divanis A.G. ; Verykios V.S. An integer programming approach for frequent itemset hiding. Proceedings of the 15th ACM conference on information and knowledge management, Arlington, Virginia, USA, ACM: New York, NY, USA, 2006, 5-11.

7. Divanis A.G.; Verykios V.S. Hiding sensitive knowledge without side effects. Knowledge and Information Systems. **2008,** 3, 263–299.

8. Divanis A.G.; Verykios V.S. Exact knowledge hiding through database extension. IEEE Transactions on Knowledge and Data Engineering. **2009**, 5, 699–713.

9. Atallah M.; Bertino E.; Elmagarmid A.; Ibrahim M., Verykios V S. Disclosure limitation of sensitive rules. Proceedings of the IEEE Knowledge and Data Engineering Exchange Workshop, Chicago, IL, USA, 7 November; IEEE: Washington, DC, USA, 1999, 45–52.

10. Amiri A. Dare to share: Protecting sensitive knowledge with data sanitization. Decision Support Systems. **2007**, 1, 181–191.

11. Oliveira S.R.M.; Zaiane O.R. Privacy preserving frequent itemset mining. In International Conference on Data Mining (ICDM), Maebashi City, Japan, December; Australian Computer Society: Darlinghurst, Australia, 2002; 43-54.

12. Oliveira S.; Zaiane O. Protecting sensitive knowledge by data sanitization. Proceedings of the Third IEEE International Conference on Data Mining (ICDM'03), Melbourne, Florida, USA, November; IEEE, 2003, 99-106.

13. Kuo Y.; Lin P.Y.; Dai B.R. Hiding frequent patterns under multiple sensitive thresholds. In Database and Expert Systems Applications (DEXA), Turin, Italy, 01-05 September 2008; Springer-Verlag: Berlin, Heidelberg, 5-18.

14. Öztürk A.C.; Ergenç B. Itemset Hiding under Multiple Sensitive Support Thresholds, In 9th International Joint Conference Knowledge Engineering and Knowledge Management, Funchal, Madeira, Portugal, 1-3 November 2017; SCI Press, 222-231.

15. Verykios V.S.; Emagarmid A.K.; Bertino E.; Saygin Y. ; Dasseni E. Association rule hiding. IEEE Transactions on Knowledge and Data Engineering. **2004**, 4, 434–447.

16. Li Y.C; Yeh J.S.; Chang C.C. MICF: An effective sanitization algorithm for hiding sensitive patterns on data mining. Advanced Engineering Informatics. **2007**, 21, 269–280.

17. Hong T.P.; Lin C.W.; Yang K.T.; Wang S.L. Using tf-idf to hide sensitive itemsets. Applied Intelligence. **2013**, 4, 502–510.

18. Cheng P.; Roddick J.F.; Chu S.C. Privacy preservation through a greedy, distortion-based rule hiding method. Applied Intelligence.**2016**, 44, 295-306.

19. Pontikakis E.D.; Tsitsonis A.A.; Verykios V.S. An experimental study of distortion-based techniques for association rule hiding. In 18th Conference on Database Security, Sitges, Catalonia, Spain, 25-28 July 2004; Springer USCY: Boston, USA, 2004, 325–339.

20. Weng C.C.; Chen S.T.; Lo H.C. A novel algorithm for completely hiding sensitive association rules. In Eighth International Conference on Intelligent Systems Design and Applications, Kaohsiung, Taiwan, 26-28 November; IEEE, 2008, 202-208.

21. Dehkordi M.S.; Dehkordi M.N. Introducing an algorithm for use to hide sensitive association rules through perturbation technique. Journal of AI and Data Mining. **2016,** 4, 219-227.

22. Yildiz B.; Ergenç B. Integrated approach for privacy preserving itemset mining. Lecture Notes in Electrical Engineering, 2012, 110, 247-260.

23. Saygin Y.; Verykios V.S.; Clifton C.Using unknowns to prevent discovery of association rules. ACM SIGMOD Record. **2001**, 4, 45–54.

24. Jadav K.B.; Vania J.; Patel D.R. Efficient hiding of sensitive association rules for incremental datasets. International Journal of Innovations & Advancement in Computer Science IJIACS. **2014,** 4, 59-65.

25. Domadiya N.; Rao U. P. Hiding Sensitive Association Rules to Maintain Privacy and Data Quality in Database. In 3rd IEEE International Advance Computing Conference (IACC), Ghaziabad, India, 13 May 2013; IEEE, 1306- 1310.

26. Dai B.R.; Chiang L.H. Hiding frequent patterns in the updated database. In International Conference on Information Science and Applications (ICISA), Seoul, South Korea, 21-23 April 2010; IEEE,1-8.

27. Nourafkan M.; Rastegari H.; Dehkord M.N. An algorithm for hiding sensitive frequent itemsets. International Journal of Advances in Soft Computing and its Applications. **2015**, 7, 51-63.

28. Garg V.; Singh A.; Singh D. A hybrid algorithm for association rule hiding using representative rule. International Journal of Computer Applications. **2014**, 97, 9-14.

29. Wnag S.L.; Jafari A. Hiding sensitive predictive association rules. In IEEE International Conference on Systems, Man and Cybernetics, Waikoloa, HI, USA, 12-12 October 2005; IEEE, 164-169.

30. Chen C.; Orlowska M.; Li X. A new framework for privacy preserving data sharing. Proceeding of the 4th IEEE ICDM Workshop: Privacy and Security Aspects of Data Mining, 2004; IEEE Computer Society, 47-56.

31. Guo Y. Reconstruction-based association rule hiding. In SIGMOD Ph.D. Workshop on Innovative Database Research, 2007. http://www.borgelt.net/ apriori.html (accessed November 7, 2018)

32. Mielikainen T. On inverse frequent set mining. Proceeding of the 3rd IEEE ICDM Workshop on Privacy Preserving Data Mining, IEEE Computer Society, p.p.18-23, 2003.

33. Lin J.; Cheng Y. Privacy preserving item set mining through noisy items. Expert Systems with Applications. **2009**, 36, 5711–5717.

34. Kennedy J.; Eberhart R. Particle swarm optimization. In IEEE International Conference on Neural Networks, Perth, WA, Australia, 27 November – 1 December 1995; IEEE,1942–1948.

35. Lin J. C.W.; Liu Q.; Viger P. F.; Hong T.P.; Voznak M.; Zhan J. A sanitization approach for hiding sensitive itemsets based on particle swarm optimization. Engineering Application of Artificial Intelligence. **2016,** 53, 1-18.

36. Öztürk A.C.; Ergenç B. Dynamic Itemset Hiding Algorithm for Multiple Sensitive Support Thresholds. International Journal of Data Warehousing and Mining. **2018**, 14, 37-59.

37. Ayav T.; Ergenç B. Full Exact approach for itemset hiding. International Journal of Data Warehousing and Mining. **2015**, 11, 49-63.

38. Le H.Q.; Arch-Int S.; Nguyen H.X.; Arch-Int N. Association rule hiding in risk management for retail supply chain collaboration, Computers in Industry. **2013,** 64, 776-784.

39. Hai L.Q.; Somjit A.; Ngamnij A. Association rule hiding based on intersection lattice. Mathematical Problems in Engineering, **2013**, 1-11.

40. Grätzer G. *Lattice theory: Foundation*. Springer, 2010.

41. Wang S. L.; Jafari A. Using unknowns for hiding sensitive predictive association rules. Proceedings of the IEEE international conference on information reuse and integration, Las Vegas, NV, USA, USA, 15-17 August 2005; IEEE, 223–228.

42. Agrawal R.; Imilinski T.; Swami A. Mining association rules between sets of items in large databases. In International Conference on Management of Data, Washington DC, USA, 25-28 May 1993; ACM, 207-216.

43. Bayardo R. J.; Agrawal R.; Gunopulos D. Constraint based rule mining on large, dense data sets. Data Mining and Knowledge Discovery. **1999**, 4 , 217-240.

44. Gkoulalas-Divanis A.; Verykios V.S. Hiding sensitive knowledge without side effects. Knowledge and Information Systems. **2009**, 3, 263-299.

45. Han J.; Pei J.; Yin Y. Mining frequent patterns without candidate generation. In ACM SIGMOD International Conference on Management of Data, Dallas, Texas, USA, 15-18 May 2000; ACM: New York, USA, 1-12.

46. Liu B.; Hsu W.; Ma Y. Mining Association Rules with Multiple Minimum Supports. Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Diego, California, USA, 15-18 August 1999; ACM: New York, USA, 337–34.

47. Roiger R.; Geatz M.W. *Data Mining: A Tutorial-Based Primer*, Addison Wesley: Boston, 2003; Vol.2.

48. Dunham M. H. *Data mining: Introductory and Advanced Topics*. Pearson Education, 2006.

49. Jain S.; Raghuvanshi R.; Ilyas M.D. A Survey Paper on Overview of Basic Data Mining Tasks. International Journal of Innovations & Advancement in Computer Science (IJIACS). **2017**, 6, 1-11.

50. Stephens S.; Pablo T. Supervised and Unsupervised Data Mining Techniques for the Life Sciences. Technical Report, Oracle and Whitehead Institute, MIT: USA, Thorn, 2003.

51. Dua D.; Karra Taniskidou E. UCI Machine Learning Repository. University of California, School of Information and Computer Science, Irvine, California, 2017.

52. Brijs T.; Swinnen G.; Vanhoof K.; Wets G. The use of association rules for product assortment decisions: A case study. Proceedings of the 5th International Conference on Knowledge Discovery and Data Mining, San Dieago, CA, 15-19 August 1999; KDD, 254–260.

53. IBM Quest Market-Basket Synthetic Data Generator. Bhalodiya, Dharmesh, 2014.

54. Bonchi F.; Saygin Y.; Verykios V.S; Atzori M.; Gkoulalas-Divani A.; Kaya S.V.; Savaş E. Privacy in Spatiotemporal Data Mining. Springer: Berlin, Heidelberg, 2008, pp 297-333.

55. Fayyad U.; Piatetsky - Shapiro G. ; Smith P. J.; Uthurasamy R. From data mining to knowledge discovery: an overview. Advances in Knowledge Discovery and Data Mining. **1996**, 1-34.

56. Zhang C.; Zhang S. *Association Rule Mining Models and Algorithms*, Springer: Berlini Heidelberg, 2002.

57. Varma D.M. Data Mining Classification Techniques Applied to Analyze the Impact of Ambient Conditions on Aero Engine Performance - A Case Study Using Xlminer. In IEEE conference on Electrical, Computer and Communication Technologies, Coimbatore, India, 5-7 March 2015; IEEE, 1-5.

58. Torra V. Privacy models and disclosure risk measures. *Data Privacy: Foundations, New Developments and the Big Data Challenge*, Springer International Publishing, 2017, pp 111–189.

59. Wang H. Association Rule: From Mining to Hiding. Applied Mechanics and Materials. **2013**, 321-324, 2570-2573.

VITA

Ahmet Cumhur Öztürk received the BSc degree in Computer Engineering from Atılım University, Turkey. From 2006 to 2009 he worked as a software engineer in IT industry. In 2010 he joined Adnan Menderes University as lecturer. He received the MS degree in Computer Engineering from Izmir Institute of Technology. He is currently a lecturer in Adnan Menderes University.