Itemset Hiding under Multiple Sensitive Support Thresholds

Ahmet Cumhur Öztürk and Belgin Ergenç Bostanoğlu
Department of Computer Engineering, Izmir Institute of Technology, 35447, Izmir, Turkey

Keywords: Privacy Preserving Association Rule Mining, Itemset Hiding, Multiple Sensitive Support Thresholds.

Abstract: Itemset mining is the challenging step of association rule mining that aims to extract patterns among items from transactional databases. In the case of applying itemset mining on the shared data of organizations, each party needs to hide its sensitive knowledge before extracting global knowledge for mutual benefit. Ensuring the privacy of the sensitive itemsets is not the only challenge in the itemset hiding process, also the distortion given to the non-sensitive knowledge and data should be kept at minimum. Most of the previous works related to itemset hiding allow database owner to assign unique sensitive threshold for each sensitive itemset however itemsets may have different count and utility. In this paper we propose a new heuristic based hiding algorithm which 1) allows database owner to assign multiple sensitive threshold values for sensitive itemsets, 2) hides all user defined sensitive itemsets, 3) uses heuristics that minimizes loss of information and distortion on the shared database. In order to speed up hiding steps we represent the database as Pseudo Graph and perform scan operations on this data structure rather than the actual database. Performance evaluation of our algorithm Pseudo Graph Based Sanitization (PGBS) is conducted on 4 real databases. Distortion given to the non-sensitive itemsets (information loss), distortion given to the shared data (distance) and execution time in comparison to three similar algorithms is measured. Experimental results show that PGBS is competitive in terms of execution time and distortion and achieves reasonable performance in terms of information loss amongst the other algorithms.

1 INTRODUCTION

Association rule mining uncovers frequent sequence of items (itemsets) to produce relationships (association rules) among items in a given transactional database (Agrawal et al., 1994; Bodon, 2003; Brijs et al., 1999; Han et al., 2000; Pei et al., 2000; Zheng et al., 2001). The rapid growth in the use of association rule mining and its challenging step of itemset mining exposed privacy problems while sharing data. Although in modern business, shared data brings mutual benefits in terms of decision making, itemset mining on the shared data may lead to malicious usage of private information if the database is shared without any precautions. Some itemsets may contain private information or knowledge; in other words the database owner might be unwilling to show them. These itemsets are called sensitive itemsets. Itemset hiding problem focuses on preventing the disclosure of sensitive itemsets.

The process of converting a database to a new one which does not comprise any sensitive itemset is called the sanitization process (Atallah et al., 1999). During this process, preserving the privacy while preventing the loss of non-sensitive knowledge and reducing the distortion on the database must be considered at the same time. Due to combinatorial nature of such a problem, there are various proposed sanitization methodologies; heuristic based approaches (Amiri, 2007; Keer and Singh, 2012; Oliveira and Zaiane, 2003; Verykios et al., 2004; Wu et al., 2007; Yildiz and Ergenc, 2012), border-based approaches (Moustakides and Verykios, 2008; Stavropoulos et al., 2016; Sun and Yu, 2004; Sun and Yu, 2007), reconstruction based approaches (Boora et al., 2009; Guo, 2007; Lin and Liu, 2007; Mohaisen et al., 2010) and exact hiding approaches (Ayav and Ergenc, 2015, Gkoulalas and Verykios, 2006; Gkoulalas and Verykios, 2008; Gkoulalas and Verykios, 2009; Menon et al., 2005). All these algorithms hide sensitive itemsets by decreasing their supports (number of occurrences of the itemset in the database) below a sensitive support threshold (defined by the user).

Most of the proposed algorithms allow user to define single sensitive support threshold. However single minimum itemset support threshold is not

ÃÚztÃijrk A. and ErgenÃğ BostanoÃ§lu B.
Itemset Hiding under Multiple Sensitive Support Thresholds.
DOI: 10.5220/0006501502220231
In *Proceedings of the 9th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (KMIS 2017)*, pages 222-231
ISBN: 978-989-758-273-8

adequate since it does not reflect the nature of different itemsets. During the sanitization process if supports of all itemsets are decreased below a given unique sensitive threshold then some itemsets may redundantly be protected while some may not be protected. Itemset hiding approach should enable the user to assign different sensitive thresholds for each sensitive itemset (Verykios and Divanis, 2004).

In this study we focus on hiding sensitive itemsets on a given transactional database by decreasing their supports below user specified multiple sensitive thresholds. Finding sensitive transactions (transactions that contain sensitive itemsets), counting supports of sensitive itemsets are essential operations in sanitization process. In order to speed up these operations we first represent the database as Pseudo Graph since 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. Our sanitization algorithm; Pseudo Graph Based Sanitization (PGBS) uses heuristics that minimize the loss of non-sensitive knowledge and distortion done on the database. PGBS does not create any artificial itemsets (itemsets that does not exist in the original database) during the sanitization process.

We carry on experiments to measure the execution time and side effect performance of our algorithm. In our experiments, we compare PGBS with two recent heuristic based algorithms; Sliding Window Algorithm (SWA) (Oliveira and Zaiane, 2003)and Template Table Based Sanitization (TTBS) (Kuo et al., 2008). Execution time and side effect (distortion and loss of non-sensitive knowledge) performance of all algorithms are measured on 4 real life databases with different characteristics in two scenarios; 1) single support threshold, and 2) multiple support thresholds. In both scenarios we measure the performance of the algorithms by varying the number of sensitive itemsets. Experimental results show that PGBS is competitive in terms of execution time and distortion while achieving reasonable performance in terms of loss of non-sensitive knowledge amongst the other algorithms especially on dense databases and with multiple support thresholds. Superiority of PGBS can better be observed on dense databases.

This paper is organized as follows; in section 2 basic definitions and metrics of frequent itemset hiding is introduced and a motivating example which is referred throughout the paper is given. In section 3 the proposed Pseudo Graph Based Saniti-zation (PGBS) algorithm is given. Section 4 gives performance evaluation of PGBS in comparison to other 3 algorithms on 4 real databases. Section 5 presents detailed survey of the related work. Section 6 is dedicated for conclusion remarks.

2 PRELIMINARIES

In this section we define the preliminaries which should be known in order to understand the problem of itemset hiding. Preliminaries include the basic definitions and metrics used in the itemset hiding process. At the end of the section we give our motivating example used throughout the paper.

2.1 Basic Definitions

Support and Frequent Itemset: Let $I = \{i_1,\ldots i_n\}$ be a set of items, a k-itemset X is a non-empty subset of I with length k. A transaction is an ordered pair of items denoted as <TID, X> where TID is the unique identifier and X is the itemset. A transactional database is a set of transactions and total number of transactions in D is denoted as |D|. Support count of an itemset X is denoted as scount(X) and it is the number of transactions containing X, the support of an itemset X is denoted as supp(X) and it is calculated as scount(X) divided by |D|. An itemset X is frequent if $supp(X) \geq \sigma$, where σ is the user specified minimum support threshold.

Sensitive Itemset and Sanitization: If FI is the set of frequent itemsets in database D and SI ($SI \subset FI$) is sensitive itemsets (the set of itemset to be hidden), the sanitization operation transforms the given transactional database D into D' where none of the itemsets in SI can extracted and the data and knowledge loss from D is kept as minimum as possible. One of the ways to hide sensitive itemsets SI from database D is to decrease their supports till the sensitive itemsets become infrequent. This process of modifying the transactions to the point where no sensitive itemset can be discovered is called the sanitization process (Atallah et al., 1999). Decreasing the support of sensitive itemsets can be achieved by deleting items called victim items (selected for deletion) from a sufficient amount of transactions called victim transactions (selected for modification). The support threshold used for hiding a given sensitive itemset X is the sensitive support threshold and it is denoted as sst(X).

Cover Degree: An item can be common in more than one sensitive itemsets. If more than one sensitive itemsets have a common item then deleting this item may sanitize more than one sensitive itemset at once (Pontikakis et al., 2004). Hiding sensitive itemsets at

once reduces the distortion on the modified database. As an example; assume that XY and YZ are two sensitive itemsets, removing the common item Y from a transaction containing XYZ decreases the support value of both XY and YZ by 1 at the same time. The cover degree of an item is the number of sensitive itemsets containing the item, e.g., cover degree of item Y in this example is 2 since it is contained by two of the given sensitive itemsets.

2.2 Sanitization Metrics

The main objective of the distortion based sanitization is hiding all sensitive itemsets while keeping the side effects at minimum level. Objective is achieving zero hiding failure.

Hiding Failure (HF) is the metric that defines the ratio of sensitive itemsets which can still be discovered with mining techniques after the database is sanitized. HF = |SI'| / |SI| where |SI| is the number of sensitive itemsets in the original database D and |SI'| is the number of sensitive itemsets in the sanitized database D'.

Side effect in this context is the unintentional loss of knowledge and data from the original database. Basic side effects are distance and information loss.

Distance (Dist*)* is the metric that defines the number of modifications made on the original database during the sanitization process. Dist = (total number of items in D) – (total number of items in D') where D is the original database and D' is the sanitized database.

Information Loss (IL) is the metric showing the number of non-sensitive frequent itemsets unintentionally removed during the sanitization process. IL = ((|FI| - |SI|) - (|FI'| - |SI'|)) / (|FI| - |SI|) where |FI| is the number of frequent itemsets and |SI| is the number of sensitive itemsets in the original atabase D, |FI'| is the number of frequent itemsets in D' and |SI'| is the number of sensitive itemsets in the sanitized database D'.

2.3 Motivating Example

In this paper we refer the transactional database given in Table 1 as a motivating example. Suppose sensitive itemsets are defined by the database owner as AD, CD and BD with 15%,10% and 5% sensitive support thresholds (sst) respectively. Degree column in Table 1 gives the number of sensitive itemsets contained by a transaction, e.g., degree of transaction 6 is 2 since it contains two sensitive itemsets AD and CD.

Table 1: Transactional database.

TID	Transactions	Degree
1	CF	0
2	ABCE	0
3	DE	0
4	ABCD	3
5	ADE	1
6	ACD	2

3 PGBS ALGORITHM

We propose Pseudo Graph Based Sanitization (PGBS) algorithm that aims to i) convert given database D into D' where no sensitive itemsets defined by the user initially can be extracted, in other words Hiding Failure is zero, ii) keep maximum number of non-sensitive itemsets present in D to minimize information loss, iii) cause minimum distortion on D to minimize distance. Conversion or sanitization is done by reducing the support of sensitive itemsets below user defined sensitive thresholds by deleting items from sufficient amount transactions till all sensitive itemsets in D become infrequent. Two sub problems arise with this strategy; first is determining the victim transactions to be modified and the second is selecting the victim item to be removed.

Block diagram given in Fig. 1 illustrates sanitization process of PGBS algorithm with our motivating example. Transactional Database D and Sensitive Itemsets Table are main inputs of the PGBS. Sensitive itemsets (SI) and their sensitive support thresholds (SST) shown in Sensitive Itemsets Table are assumed to be defined by the preferences or privacy policies of the user.

There are 4 sub processes in PGBS; 1) ***Convert Pseudo Graph*** converts the Transactional Database into a Pseudo Graph (PG), 2) ***Create Sensitive Count Table*** creates the Sensitive Count Table (SCT) that holds the number of necessary modifications/distortions for each sensitive itemset given by the user, 3) ***Create Sanitization Table*** is the main sanitization process that works on PG and creates Sanitization Table that keeps the necessary modifications to be applied on the original database, 4) ***Sanitize Database*** deletes each victim item in D from its corresponding transaction in the Sanitization Table and prepares sanitized database D'.

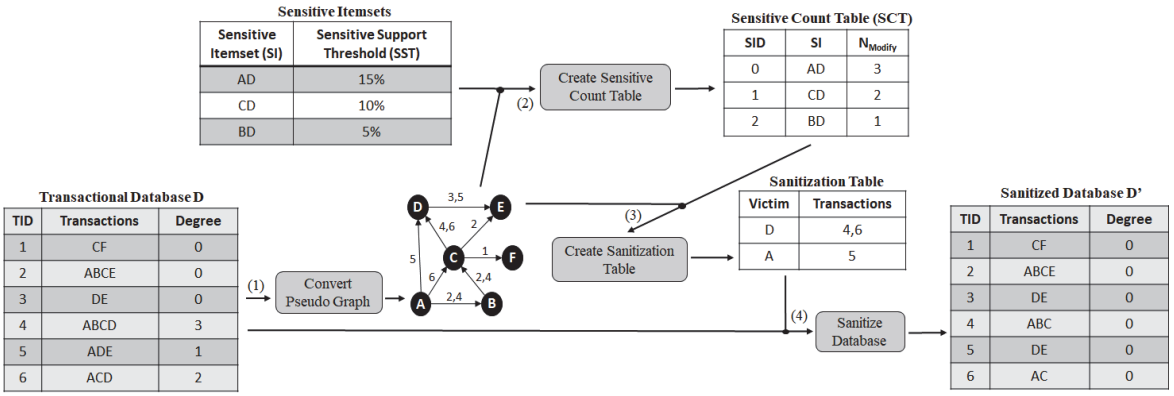

Figure 1: Block diagram of PGBS algorithm.

In the following sections we explain these processes in detail.

3.1 Convert Pseudo Graph

We use Pseudo Graph (PG) data structure to represent all transactions of the given database D. PG provides an efficient way to identify and modify transactions without accessing the actual database. A PG is a directed graph which allows multiple edges and loops. In this graph each item is represented as vertex and vertices are connected to each other by edges where edges are labelled with transaction ids, e.g. if item **X** appears with item **Y** in transaction **k** then vertex **X** is connected with vertex **Y** with a directed edge from **X** to **Y** with label **k**. Reflexive edges are required since the database might contain single length transactions resulting in loops.

Insertion of transactions to the PG is a simple process; first transactions in database D are sorted in lattice order and then each transaction is inserted one by one. For illustration Fig.2 (a), (b) and (c) shows the *PG* after transactions {CF}, {ABCD}, and {DE} in Table 1 are inserted into *PG* respectively and

Fig.2(d) shows the PG after the remaining transactions in Table 1 are inserted into PG.

Counting the support count of a given item **X** is performed by counting the total number of distinct transaction ids on incoming and outgoing edges of vertex X. Let **XY** be a 2-itemset and prefix(**X**) be the transactions on outgoing edges of vertex **X** and postfix(**Y**) be the transactions on incoming edges of vertex **Y**. Support of **XY** is simply computed as prefix(**X**) ∩ postfix(**Y**). Support of an k-itemset Z where k>2 is calculated by prefix($item_1$) ∩ postfix($item_2$) ∩… ∩ postfix($item_k$) where $item_i$ is the item at ith position in Z. As an example in Fig.2 d) transactions containing itemset {ACE} is equal to ({2,4,5,6} ∩ {2,4,6}) ∩ {2,3,5} = {2} so the support of itemset {ACE} is 1.

The inverted index structure is similar to the PG. The inverted index is used to store list of transaction

The inverted index is used to store list of transaction ids containing each item As in PG transactions of an itemset can be uncovered by performing intersection operation between inverted indexes of all items in the given itemset. The iteration number for uncovering transactions ids of an itemset using the inverted index is always greater than or equal to the iteration number for uncovering transactions ids of an itemset with using PG. PG data structure considers whether an item is prefix or postfix in a given itemset and tries to put least number of transaction ids to the intersection operation.

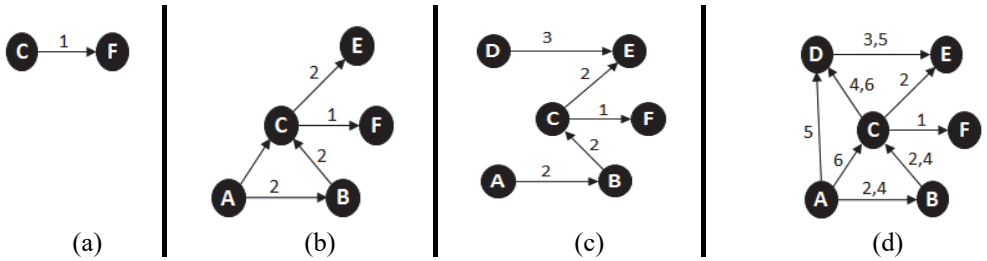

Figure 2: PG by adding transaction **(a)** CF **(b)** ABCE **(c)** DE **(d)** all.

3.2 Create Sensitive Count Table

Create Sensitive Count Table process takes sensitive itemsets with their sensitive thresholds and pseudo graph representation of the database as input and then creates the Sensitive Count Table (SCT) as shown in Figure 1. SCT has three fields; SID is the unique identifier, SI is the sensitive itemset and N_{Modify} is the minimum number of transactions that are needed to be modified for hiding the given sensitive itemset. N_{Modify} value of a sensitive itemset X is calculated by the following equation:

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

where scount(X) is the number of transactions containing the itemset X, sst(X) is the sensitive support threshold that is given by the user for the itemset X, |D| is the total number of transactions in D. Records in the Sanitization Table are sorted according to N_{Modify} values of the sensitive itemsets in descending order.

3.3 Create Sanitization Table

Create Sanitization Table procedure conceals sensitive itemsets from PG and creates the Sanitization Table by using the SCT, the steps of this procedure is given below.
For all the rows of SCT do;

1. *Select the victim item:* Select the victim item among items in sensitive itemsets stored in the first record of SCT having the maximum cover degree. If there is more than one victim item with the same cover degree, select the item which has the maximum support count. If there is still more than one victim item with the same support count, select the victim item randomly.
2. *Unify sensitive itemsets:* Unify all sensitive itemsets in SCT that contain the victim item selected in Step 1.
3. *Select the sensitive transactions:* Find enough sensitive transactions containing unified sensitive itemsets in PG (less than or equal N_{Modify} value in the active row of SCT).
4. *Delete item:* Add victim item and sensitive transactions id pairs to the Sanitization Table. Update the PG by deleting victim item from sensitive transactions. For each sensitive itemset that is in the unified sensitive itemsets, reduce the N_{Modify} values of their corresponding rows of Sanitization Table by the number of sensitive transactions uncovered. If N_{Modify} value of any record in SCT becomes zero, then remove it from the SCT and update the cover degree of each item in remaining sensitive itemsets stored in SCT.
5. *Control:* If active N_{Modify} value is still greater than zero and none of the sensitive itemsets in SCT is deleted, then remove the sensitive itemset that has the least N_{Modify} value from the unified sensitive itemsets and go back to Step 3, else go back to Step 1.

3.4 Sanitize Database

This procedure updates the database D by using the Sanitization Table. Each victim item is deleted from corresponding transaction ids stored in Sanitization Table and the resulting database brings out thee sanitized database.

3.5 Illustrating Example

Suppose we have our motivating example given in Section 2.3. Our proposed processes work as follows to hide all sensitive itemsets.

1. **Convert Pseudo Graph:** The *Convert Pseudo Graph* procedure converts the database into PG form as shown in Fig.2 (d).
2. **Create Sensitive Count Table:** N_{Modify} value of each sensitive itemset is calculated by using the equation (1), and then added to the SCT. After SCT is created it is sorted in descending order of N_{Modify} value. The resulting SCT for this example is shown in Figure 1.
3. **Create Sanitization Table:** The process *Create Sanitization Table* first selects the victim item. In this example there are three sensitive itemsets in SCT: AD, CD and BD. The sensitive itemset AD has the highest N_{Modify} value, so the victim item is selected among items in AD. The item "D" (cover degree = 3) is selected as victim item (Step 1). Sensitive itemsets in SCT containing "D" are unified. The unified sensitive itemset is ABCD and according to Fig.2 (d) only the transaction {4} contains ABCD (Step 2 - 3).

   Item "D" is removed from 4th transaction in PG, then the pair {D, 4} is added to the Sanitization Table and N_{Modify} value of each sensitive itemset in SCT is decreased by 1. N_{Modify} value of the sensitive itemset BD becomes zero so this row is removed from the SCT. Since the N_{Modify} value of AD is still greater than zero and one of the records is removed from the SCT, the sensitive itemset BD is discarded from the unification operation and the new unified sensitive itemset becomes ACD (Step 4).

The N_{Modify} value of AD is 2 so the algorithm tries to find out 2 sensitive transactions containing ACD from the PG. For the itemset ACD, supporting transactions is only {6}, so the victim item "D" is deleted from this transaction and the pair {D,6} is added to the Sanitization Table. After the delete operation the N_{Modify} value of CD becomes zero and AD becomes 1, so CD is removed from the SCT. The only remaining sensitive itemset left in SCT is AD, because both item "A" and "D" has cover degree 1, the item having the highest support count is selected as victim item which is "A" (support count of "A" is 4 in PG). Then item "A" is removed from 5th transaction in PG and the pair {A, 5} is added to the Sanitization Table as shown in Figure 1.

4. **Sanitize Database:** For this example, the *Sanitize Database* procedure deletes each item with corresponding transactions stored in Sanitization Table from the original database D given in Table 1. The resulting sanitized database D' is shown in Figure 1. As we see item "D" is deleted from transactions 4 and 6, item "A" is deleted from transaction 5.

4 PERFORMANCE EVALUATION

We compared our Pseudo Graph Based Sanitization (PGBS) algorithm with Sliding Window Algorithm (SWA) (Oliveira and Zaiane, 2003) and Template Table Based Sanitization (TTBS) (Kuo et al., 2008) using 4 real databases and these four databases are sorted in lattice order before processing but we neglect the sorting time in performance measures. The main objective of all algorithms is to achieve zero hiding failure in other words; they hide all sensitive itemsets by reducing their support below their sensitive support thresholds. We chose SWA and TTBS algorithms for performance evaluation because both focus on overlapping items in sensitive itemsets and they enable assigning different sensitive thresholds for each sensitive itemset. We measure execution time, information loss and distance performance of all algorithms. 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. The execution times include I/O and CPU time.

Table 2: Characteristics of experimental databases.

Database Name	# of Transactions	Distinct Items	Avg. Trans. Length	Density (%)
Chess	3,196	75	37	49.4
Connect	67,557	129	43	33.4
Mushroom	8,124	119	23	19.4
Pumsb	49,047	2,113	75	3.6

4.1 Databases

We conduct our experiments using 4 real databases; Mushroom, Chess, Connect and Pumsb. The characteristics of these databases in terms of number of transactions, number of distinct items, average transaction length and density (density of a database is the average transaction length divided by number of distinct items) are given in Table 2. We indicate the notion of sparse and dense for each database because as pointed out in (Bayardo et al., 1999; Gkoulalas and Divanis, 2010; Han et al., 2000; Pei et al., 2000) frequent itemsets generated from dense databases may result in generating long frequent itemsets at various levels of support. Chess, Connect and Mushroom databases are obtained from UCI Repository (Blake and Merz, 1998). Pumbs database contains census data from PUMS and can be obtained from http://www.almaden.ibm.com/soft- ware/quest.

Table 3: Support ranges for databases.

	Chess	Mushroom	Connect	Pumsb
Bin	Support Range	Support Range	Support Range	Support Range
1	(0.6001, 0.6136]	(0.1100, 0.1218]	(0.85, 0.8575]	(0.85, 0.8575]
2	(0.6136, 0.6308]	(0.1218, 0.1388]	(0.8575, 0.8672]	(0.8575, 0.8672]
3	(0.6308, 0.6555]	(0.1388, 0.1540]	(0.8672, 0.8792]	(0.8672, 0.8792]
4	(0.6555, 0.6974]	(0.1540, 0.2053]	(0.8792, 0.8985]	(0.8792, 0.8985]
5	(0.6974, 0.9962]	(0.2053, 1]	(0.8985, 0.9987]	(0.8985, 0.9987]

4.2 Results with Multiple Sensitive Support Thresholds

In this section we compare our PGBS algorithm with SWA and TTBS algorithms using 4 real databases by assigning multiple sensitive thresholds. PGBS, TTBS and SWA algorithms allow user to define different sensitive thresholds for each sensitive itemset however RS algorithm does not. Because of this, we do not use the RS algorithm for performance evaluation. We partition the Chess, Connect, Mushroom and Pumsb databases into 5 bins, and then

we randomly select 2 itemsets from each bin and assign the minimum support threshold as the minimum support given in the support range. Support ranges of the bins for each database are shown in Table 3.

The results of SWA, TTBS and PGBS are given in Table 4 where time is CPU execution time in seconds, information loss is in percentage and distance is in percentage. We underline each best performance result among 4 algorithms for execution time, information loss and distance for better readability.

Last row of the table shows the number of best results achieved by the algorithm corresponding to execution time, information loss and distance. If we analyse the algorithms with their best result summaries; we see that SWA [0, 11, 5], TTBS [3, 0, 4] and PGBS [9, 1, 6]. In other words, PGBS achieves best execution time in 9 out of 12 cases, best information loss in 1 out of 12 and best distance 6 out of 12 cases. SWA achieves best information loss in most of the cases with a very high cost of execution time. PGBS is better than others in terms of execution time and distance especially on dense databases like Chess and Mushroom.

5 RELATED WORK

Privacy preserving association rule hiding problem was first introduced in (Atallah et al., 1999).

The authors proposed heuristic algorithms and gave the proof of NP-Hardness of optimal sanitization. Since then, many approaches have been proposed to preserve privacy for sensitive patterns or sensitive association rules in database. Most common categorizations of rule or itemset hiding approaches can be done according to the nature of the base algorithm and following classes appear; border based approaches (Moustakides and Verykios, 2008; Sun and Yu, 2004; Sun and Yu, 2007), exact approaches (Ayav and Ergenç, 2015; Gkoulalas and Verykios, 2006; Gkoulalas and Verykios, 2008; Gkoulalas and Verykios, 2009; Menon et al., 2005), reconstruction based approaches (Bodon, 2003; Guo, 2007; Lin and Liu, 2007; 26. Mohaisen et al., 2010) and heuristic

Table 4: Comparison with multiple support thresholds.

		SWA			TTBS			PGBS		
Database	|SI|	Time (sec)	Information Loss (%)	Distance (%)	Time (sec)	Information Loss(%)	Distance(%)	Time (sec)	Information Loss (%)	Distance (%)
Chess	10	4.72	49.89	0.79	0.39	68.30	0.41	0.22	49.91	0.80
	20	8.80	62.24	0.94	0.56	78.67	0.92	0.25	63.29	0.95
	30	13.08	62.84	0.96	0.82	75.41	0.81	0.35	63.29	0.97
Mushroom	10	26.79	45.29	3.34	0.88	72.67	6.92	0.37	49.87	3.34
	20	47.50	45.32	3.34	1.72	86.97	10.84	0.50	49.99	3.35
	30	66.17	49.52	3.42	2.28	88.28	11.59	0.55	54.32	3.42
Connect	10	804.21	54.31	0.25	5.18	86.44	0.68	7.18	55.28	0.25
	20	1935.28	63.40	0.33	10.44	84.93	0.68	17.55	65.03	0.3
	30	2771.64	60.81	0.30	9.83	84.93	0.68	18.23	60.25	0.29
Pumsb	10	1362.31	49.32	0.23	7.72	94.96	0.72	6.45	77.63	0.42
	20	2359.06	54.98	0.24	8.75	91.48	0.21	7.44	55.91	0.24
	30	2899.27	57.76	0.26	14.44	91.24	0.73	9.86	58.51	0.25
	# of Best Results	**-**	**11**	**5**	**3**	**-**	**4**	**9**	**1**	**6**

approaches (Amiri, 2007; Keer and Singh, 2012; Oliveira and Zaiane, 2002; Yildiz and Ergenc, 2012; Oliveira and Zaiane, 2003; Verykios et al., 2004; Wu et al.,2007).

Heuristic approaches rely on some heuristics for database sanitization, they may produce side effects such as loss of non-sensitive rules/itemsets in the sanitized database and generation of rules/itemsets that are not truly exists in the original database and the distortion done on the data. Despite their side effects majority of the research is based on heuristic approaches (Verykios and Divanis, 2004) because they are efficient and have good response time. In Table 5, we summarize and compare existing heuristic based sanitization algorithms based on their important characteristics.

First column of the table gives the name of the algorithm; second column indicates the focus of sanitization; some algorithms try to hide sensitive "itemsets" whereas some of them hide sensitive "rules". Second column shows whether the algorithm is designed for association rule ("Rules") or itemset ("Itemset") hiding. The third column shows whether the algorithm enables assigning different sensitive thresholds to each sensitive itemset or rule. Fourth column is dedicated to show the capability of the algorithm to hide single or multiple itemsets in a single run.

Column 5 of Table 5 gives information about the approach of the algorithm in selecting the victim transaction for sanitization; "Length" indicates that the algorithm selects the transaction according to its length amongst the candidate sensitive transactions; "Degree" indicates that the selection is done according to the existence of the number of sensitive itemsets in the transaction, "Greedy" indicates that the selection is done according to trial and error.

Column 6 shows the victim item selection criteria of the algorithm; "Cover" is the number of presences of the item in different sensitive

Table 5: Summary of existing heuristic based algorithms.

Algorithm	Hiding	Multiple Support Thresholds	Multiple Rule Hiding	Victim Transaction Selection	Victim Item Selection	Year
1	2	3	4	5	6	7
IGA (Oliveira and Zaiane, 2002)	Itemsets		✓	Degree	Cover	2002
Naïve (Oliveira and Zaiane, 2002)	Rules			Degree	All	
MaxFIA (Oliveira and Zaiane, 2002)	Itemsets			Degree	Support	
MinFIA (Oliveira and Zaiane, 2002)	Itemsets			Degree	Support	
SWA (Oliveira and Zaiane, 2003)	Itemsets	✓	✓	Length	Support	2003
DSA (Oliveira et al., 2004)	Itemsets			---	---	2004
Algorithm 2.b (Verykios et al.,2004)	Itemsets			Length	Support	
Algorithm 2.c (Verykios et al., 2004)	Itemsets			Length	All	
PDA (Pontikakis et al., 2004)	Rules			Weight	Greedy	
WDA (Pontikakis et al., 2004)	Rules			Weight	All	
Aggregate (Amiri, 2007)	Itemsets		✓	Greedy	All	2007
Disaggregate (Amiri, 2007)	Itemsets		✓	Greedy	Greedy	
Hybrid (Amiri, 2007)	Itemsets		✓	Greedy	Greedy	
MICF (Li et al., 2007)	Itemsets		✓	Degree	Cover	
TTBS (Kuo et al., 2008)	Itemsets	✓	✓	Degree	Cover	2008
FHSAR (Weng et al., 2008)	Rules		✓	Weight	Cover	
SIF-IDF (Hong et al., 2013)	Itemsets			Weight	Support	2013
RelevanceSorting (Cheng et al., 2016)	Rules			Weight	Support	2016
HSARWI (Sakenian and Naderi, 2016)	Rules		✓	Weight	Weight	

itemsets, “Support” shows the selection of the item is done depending on its support,” Greedy” shows the selection of the item is done in trial and error,” All” shows the whole sensitive itemset in sensitive transaction is deleted rather than deleting a victim. Last column shows the year of the related research.

When we analyse the existing heuristic sanitization algorithms we see that i) they use different heuristics targeting to reduce the execution time, distance, information loss while maintaining minimum hiding failure, ii) there are few heuristic based approaches that focus on sanitization under multiple support thresholds.

6 CONCLUSIONS

In the case of applying itemset mining on the shared data of organizations, each party needs to hide its sensitive knowledge before extracting global knowledge for mutual benefit. In this study we focus on privacy preserving itemset hiding under multiple support thresholds. Our algorithm (PGBS) utilizes pseudo graph data structure that is used to store the given transactional database to prevent multiple scans of the given database and allow effective sanitization process. We validate execution time and side effect performances of our algorithm, Pseudo Graph Based Sanitization (PGBS) in contrast to two recent algorithms on 4 real databases varying number of sensitive itemsets and sensitive thresholds. Experimental results show that PGBS is competitive in terms of execution time and distance especially on dense datasets amongst the other algorithms. For future work, we want to propose dynamic version of our algorithm that is able to sanitize the updated databases.

ACKNOWLEDGEMENTS

This work is partially supported by the Scientific and Technological Research Council of Turkey (TUBITAK) under ARDEB 3501 Project No: 114E779

REFERENCES

Agrawal, R., Srikant, R., 1994. Fast algorithms for mining association rules in large databases. *In: 20th International Conference on Very Large Databases, pp. 487-499.*

Amiri, A., 2007. Dare to share: Protecting sensitive knowledge with data sanitization. *Decision Support Systems 43(1), pp. 181-191.*

Atallah, M., Bertino, E., Elmagarmid, A., Ibrahim, M., Verykios, VS., 1999. Disclosure limitation of sensitive rules. *In: Workshop on Knowledge and Data Engineering Exchange, pp. 45-52.*

Ayav, T., Ergenç, B., 2015. Full Exact approach for itemset hiding. *International Journal of Data Warehousing and Mining. 11(4).*

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

Blake, CL., Merz, CJ., 1998. *UCI Repository of Machine Learning Databases.* University of California, Irvine, Dept. of Information and Computer Sciences.

Bodon, F., 2003. *A fast APRIORI implementation. Workshop Frequent Itemset Mining Implementations (FIMI’03), vol. 90, pp. 56-65.*

Boora, RK., Shukla, R., Misra, A., 2009. An improved approach to high level privacy preserving itemset mining. *International Journal of Computer Science and Information Security, 6(3), pp. 216-223.*

Brijs, T., Swinnen, G., Vanhoof, K., Wets, G., 1999. Using association rules for product assortment decisions: a case study. *In Knowledge Discovery and Data Mining, pp. 254–260.*

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

Gkoulalas-Divanis, A., Verykios, VS., 2006. An integer programming approach for frequent itemset hiding. *ACM International Conference on Information and Knowledge Management.*

Gkoulalas-Divanis, A., Verykios, VS., 2008. A parallelization framework for exact knowledge hiding in transactional databases. *IFIP International Federation for Information Processing, vol. 278, pp. 349-363.*

Gkoulalas-Divanis, A., Verykios, VS., 2009. Hiding sensitive knowledge without side effects. *Knowledge and Information Systems, 20(3), pp. 263-299.*

Gkoulalas-Divanis, A., Verykios VS., 2010. Association rule hiding for data mining. Springer.

Guo, Y., 2007. Reconstruction-based association rule hiding. SIGMOD Ph.D. *Workshop on Innovative Database Research.*

Han J., Pei J., Yin, Y., 2000. Mining frequent patterns without candidate generation. *In ACM SIGMOD International Conference on Management of Data, pp. 1-12.*

Hong, T-P., Lin, C-W., Yang, K-T., Wang, S-L., 2013. Using tf-idf to hide sensitive itemsets. *Appl Intell, 38(4), pp. 502–510.*

Keer, S., Singh, A., 2012. Hiding sensitive association rule using clusters of sensitive association rule. *International Journal of Computer Science and Network, 1(3).*

Kuo, Y., Lin, PY., Dai, BR., 2008. Hiding frequent patterns under multiple sensitive thresholds. *Database and Expert Systems Applications Lecture Notes in Computer Science, 5181, pp. 5-18.*

Lin, J., Liu, JYC., 2007. Privacy preserving itemset mining through fake transactions. *22nd ACM Symposium on Applied Computing, pp. 375-379.*

Li, YC., Yeh, JS., Chang, CC., 2007. MICF: An effective sanitization algorithm for hiding sensitive patterns on data mining. *Advanced Engineering Informatics, 21, pp. 269–280.*

Menon, S., Sarkar, S., Mukherjee, S., 2005. Maximizing accuracy of shared databases when concealing sensitive patterns. *Information Systems Research 16(3), pp. 256-270.*

Mohaisen, A., Jho, N., Hong, D., Nyang, D., 2010. Privacy preserving association rule mining revisited: Privacy enchancement and resource effcieny. *IEICE Transactions on Information and Systems 93(2), pp. 315-325.*

Moustakides, GV., Verykios, VS., 2008. A maxmin approach for hiding frequent itemsets. *Data and Knowledge Engineering 65(1), pp. 75-89.*

Oliveira, SRM., Zaiane, OR., 2002. Privacy preserving frequent itemset mining. *International Conference on Data Mining, pp. 43-54.*

Oliveira, SRM., Zaiane, OR., 2003. Algorithms for balancing privacy and knowledge discovery in association rule mining. *Seventh International Database Engineering & Applications Symposium, pp. 54-63.*

Oliveira, SRM., Zaiane OR., Saygin, Y., 2004. Secure association rule sharing. Advances in knowledge discovery and data mining, *8th Pacific-Asia Conference, pp. 74–85.*

Pei, J., Han, J., Mao, R., 2000. CLOSET: An efficient algorithm for mining frequent closed itemsets. *ACM-SIGMOD Int. Workshop Data Mining and Knowledge Discovery, pp. 11–20.*

Pontikakis, ED., Tsitsonis, AA., Verykios, VS., 2004. An experimental study of distortion-based techniques for association rule hiding. *18th Conference on Database Security, pp. 325–339.*

Sakenian Dehkordi, M., Naderi Dehkordi, M., 2016. Introducing an algorithm for use to hide sensitive association rules through perturbation technique. *Journal of AI and Data Mining.*

Stavropoulos, E.C., Verykios, V.S., Kagklis, V., 2016. A transversal hypergraph approach for the frequent itemset hiding problem. *Knowledge and Information Systems, pp. 625-645.*

Sun, X., Yu, PS., 2004. A border-based approach for hiding sensitive frequent itemsets. *5th IEEE International Conference on Data Mining, pp. 426-433.*

Sun, X., Yu, PS., 2007. Hiding sensitive frequent itemsets by a border-based approach. *Computing Science and Engineering 1(1), pp. 74-94.*

Verykios, VS., Emagarmid, AK., Bertino, E., Saygin, Y., Dasseni, E., 2004. Association rule hiding. *IEEE Transactions on Knowledge and Data Engineering, 16(4), pp. 434-447.*

Verykios, V.S., Divanis, A.G., 2004. A survey of association rule hiding methods for privacy. *Advances in Knowledge Discovery and Data Mining: 8th Pacific-Asia Conference.*

Weng, C., Chen, S., Che Lo, H., 2008. A novel algorithm for completely hiding sensitive association rules. *Eighth International Conference on Intelligent Systems Design and Applications IEEE.*

Wu, YH., Chiang, CM., Chen, A., 2007. Hiding sensitive association rules with limited side effects. *IEEE Transactions on Knowledge and Data Engineering, 19(1), pp. 29-42.*

Yildiz, B., Ergenc, B., 2012. Integrated approach for privacy preserving itemset mining. *Lecture Notes in Electrical Engineering Volume 110, pp. 247-260.*

Zheng, Z., Kohavi, R., Mason, L., 2001. Real world performance of association rule algorithm. *Proceedings of 7th ACM-SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 401–406.*