Received: 17 May 2019 | Revised: 24 December 2019 | Accepted: 24 March 2020 | Published on: 20 April 2020

DOI: 10.1002/sam.11458

WILEY

RESEARCH ARTICLE

Minimizing information loss in shared data: Hiding frequent patterns with multiple sensitive support thresholds

Belgin Ergenç Bostanoğlu[1] | **Ahmet Cumhur Öztürk**[2]

[1]Department of Computer Engineering, Izmir Institute of Technology, Izmir, Turkey

[2]Department of Computer Aided Design and Animation, Aydın Adnan Menderes University, Aydın, Turkey

Correspondence
Ahmet Cumhur Öztürk, Department of Computer Aided Design and Animation, Aydın Adnan Menderes University, Aydın, Turkey.
Email: cumhur.ozturk@adu.edu.tr

Abstract

Privacy preserving data mining (PPDM) is the process of protecting sensitive knowledge from being discovered by data mining techniques in case of data sharing. Privacy preserving frequent itemset mining (PPFIM) is a subtask and NP-hard problem of PPDM. Its objective is to modify a given database in such a way that none of the sensitive itemsets of the database owner can be obtained by any frequent itemset mining technique from the modified database. The main challenge of PPFIM is to minimize the distortion given to the data and nonsensitive knowledge while sanitizing all given sensitive itemsets. Distortion-based sensitive itemset hiding algorithms decrease the support of each sensitive itemset under a predefined sensitive threshold through sanitization. Most of the distortion-based itemset hiding algorithms allow database owner to define a single sensitive threshold for each sensitive itemset. However, this is a limitation to the database owner since the importance of each sensitive itemset varies. In this paper we propose a distortion-based itemset hiding algorithm that allows database owner to assign multiple sensitive thresholds, namely itemset oriented pseudo graph based sanitization (IPGBS) algorithm. The purpose of IPGBS algorithm is to give minimum distortion to the nonsensitive knowledge and data while hiding all sensitive itemsets. For this reason, the IPGBS algorithm modifies least amount of transaction and transaction content. The performance evaluation of the IPGBS algorithm is conducted by using two different counterparts on four different databases. The results show that the IPGBS algorithm is more efficient in terms of nonsensitive frequent itemset loss on both dense and sparse databases. It has considerable good results in terms of number of transactions modified, number of items deleted, execution time and total memory allocation as well.

KEYWORDS

information loss, itemset mining, privacy preserving itemset mining

1 | INTRODUCTION

Association rule mining is a data mining task used for obtaining relationships available among items in a transactional database and it is very beneficial in decision making and business planning. Initially association rule mining was used for market basket analysis to find out how products are purchased by customers. Since then, it has been employed in many application domains such as bioinformatics, telecommunications and medical treatments. However, companies or organizations share data with each other for mutual benefit and before sharing, they want to protect their data within the framework of sensitive association rules that reveal out their strategic knowledge. This raises the privacy preserving association rule mining problem. The solution is to remove sensitive (confidential) association rules from the data before its release and this can be performed by transforming a database into a new one where the sensitive association rules can no longer be mined out. Association rules are generated in two steps, the first step is discovering frequent itemset, and the second step is generating rules using the frequent itemsets [15]. Items occurring frequently together in a transnational database are called frequent itemsets. One of the ways of hiding sensitive association rules is hiding the frequent itemsets that the sensitive association rules are generated. These frequent itemsets are called sensitive itemsets and it is possible to conceal the sensitive itemsets by decreasing their support under predefined sensitive threshold. The support decrease operation is possible by deleting some items from the database, but this type of modification brings out side effects such as loss of nonsensitive knowledge and distortion on the data. An optimal solution for frequent itemset hiding algorithm is to minimize these side effects while hiding all given sensitive itemsets. It was proven that finding such an optimal solution is NP-hard problem [7]. In the literature various approaches have been proposed for the frequent itemset hiding problem including borderbased [27,34], exact [8,16,26], reconstruction based [1,9,21], cryptography based [24] and heuristic based [20,30,31]. Each approach tries to hide all sensitive itemsets while reducing the side effects given to the data and knowledge to the minimum.

Heuristic approaches come forward with small response time and they may have optimal hiding solution if the global attributes of sensitive itemsets and transactional databases are well considered. The global attributes of sensitive itemsets are referred as characteristics of itemsets such as their support, their mutual dependencies and number of items they contain while the global attributes of transactional database are referred as the characteristics of databases such as their density and size. However, a few of the heuristic sensitive itemset hiding algorithms are designed with considering multiple sensitive thresholds. Assigning single sensitive threshold for each sensitive itemset may decrease support of some sensitive itemsets more than necessary and this brings out the increased loss in nonsensitive knowledge since in most of the databases some itemsets appear frequently while others appear rarely. Also, assigning unique sensitive threshold for each sensitive itemset gives limitation to the database owner in deciding the hiding degree of different sensitive itemsets [23].

In this study we propose a distortion-based heuristic frequent itemset hiding algorithm IPGBS (itemset oriented pseudo graph-based sanitization). As PGBS which is proposed with detailed benchmark in [31], the IPGBS uses pseudo graph data structure and allows database owner to assign multiple sensitive thresholds. The pseudo graph of IPGBS is more compact than that of PGBS and contains only sensitive itemsets as nodes and sensitive transaction ids as edge labels. The main objective of IPGBS algorithm is modifying minimum number of transactions in order to cause minimum number of nonsensitive knowledge loss. The IPGBS is compared to two similar counter parts, PGBS [31] and transaction-oriented pseudo graph based sanitization (TPGBS) where the TPGBS is a version of the hiding algorithm proposed with detailed benchmark in [32]. The Pseudo graph data structure that we use in this study has two major goals. First, graph structure only represents transactions containing sensitive itemsets rather than representing all transactions and in this way, total memory requirement is minimized. Second the longest paths in this graph structure represents transactions containing maximum number of sensitive itemsets and this makes uncovering the sensitive transactions having highest degree easier compared to scanning the actual database where degree of a transaction is the number sensitive itemsets it contains. The experiments demonstrate that IPGBS is more effective in terms of distortion given to the nonsensitive knowledge in both sparse and dense databases. It is also more effective in terms of execution time and total memory consumption in sparse databases. It is very competitive in terms of number of transactions modified and number of items deleted.

This paper is organized as follows. In Section 2 the preliminaries of itemset mining and hiding are explained together with a motivating example. In Section 3 the literature review related with itemset hiding is given. In Section 4 proposed algorithm itemset oriented pseudo graph based sanitization (IPGBS) is introduced and explained with examples. Experimental results, comparing IPGBS with PGBS [31] and TPGBS [32] on six different databases are presented and discussed in Section 5. Finally, paper ends with conclusion remarks of Section 6.

2 | PRELEMINERIES AND PROBLEM DEFINITION

This section first introduces some basic concepts and metrics related to itemset mining and itemset hiding. Next problem formulation that includes the objective and side effects is given together with a motivating example.

2.1 | Itemset mining

Frequent itemset generation is the first and computationally expensive step of association rule mining. Basic idea of this process is to find occurrences of the patterns (itemsets) that are more than user defined thresholds. When the size of the data is huge in terms of number of transactions and number of distinct items number of candidate patterns generated become enormous. There are different solutions for frequent itemset mining where user defined support threshold can be unique as in Apriori [4], Fp-Growth [17] or Eclat [38] or multiple support thresholds can be allowed as in CFP-Growth++ [19], MISFP-Growth [11], MIS-Eclat [12], dynamicity of data can be considered as in Dynamic Matrix Apriori [28], dynamic CFP-Growth [3], and dynamic MIS [2]. Formal definition of the problem and related terminology can be given as follows. Let $I = I_1, I_2, I_3, \dots, I_n$ items. An itemset X is a proper subset of I. A transaction is composed of any combination of itemset X that can be generated from I. A transactional database is a set of transactions denoted as D. Each transaction tr in D is denoted as tr = <tid,X> where *tid* is an unique identifier of the transaction *tr* and X is any itemset generated from any possible combination of items of set I. The number of transactions a given database D contains is denoted as $|D|$. The support count of X is denoted as scount(X) and it is the number of transactions containing X in D. The Support of X is the percentage representation of scount(X) and calculated as: $(\text{scount}(X)/|D|) \times 100$. A given itemset is frequent if its support is greater than predefined support threshold.

2.2 | Itemset hiding

An itemset is sensitive if it needs to be hidden in database D based on some privacy concerns of the database owner. Sensitive threshold is defined by the database owner and support of a sensitive itemset must be smaller than sensitive threshold to hide the sensitive itemset. For the convenience of organization, in the rest of this paper, sensitive itemset that has support greater than predefined sensitive threshold is called strong sensitive itemset. A transaction containing at least one sensitive itemset is called sensitive transaction. Degree of a sensitive transaction indicates the number of sensitive itemsets that the transaction contains. The cover degree of an item i is the number of sensitive itemsets containing item *i*.

For illustrating the given definitions suppose a transactional database D is given as in Figure 1A. The set of items are $I = \{A, B, C, D, E, F, G\}$ and all possible itemsets generated from D with minimum support of 60% are given in Figure 1B. The sensitive itemsets with their sensitive thresholds are given in Figure 1C. All transactions in D are sensitive and degree of each transaction is shown in the degree column of Figure 1A. The support of sensitive itemset AB is 62.5%, BC is 75% and DF is 50% where support of all sensitive itemsets are greater than the predefined sensitive threshold values so all of them are strong sensitive itemsets. Cover degree of all items are: $\{A,1\}$, $\{B,2\}$, $\{C,1\}$, $\{D,1\}$, $\{F,1\}$.

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 strong sensitive itemsets under their predefined sensitive thresholds. 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 and is first proposed by [7]. The distortion-based frequent itemset hiding approaches may pose some side effects as hiding failure, information loss, accuracy, artificial cost and database dissimilarity.

Hiding failure (HF) is the side effect measured as the amount of the sensitive itemsets left unhidden by the sanitization process. It is calculated as:

$$HF = \frac{|SI(D')|}{|SI(D)|} \tag{1}$$

where $|SI(D')|$ is the number of sensitive itemsets in D' and $|SI(D)|$ is the number of sensitive itemsets in D.

Information loss is the side effect measured as the amount of nonsensitive itemsets hidden unintentionally by the sanitization process. It is calculated as:

$$HF = \frac{(|FI(D)| - |SI(D)|) - (|FI(D')| - |SI(D')|)}{|FI(D)| - |SI(D)|} \tag{2}$$

Accuracy is the side effect to evaluate total number of transactions changed by the sanitization process and it is calculated as:

$$Accuracy = |Tr(D)| - |TrChanged(D')| \tag{3}$$

where $|Tr(D)|$ is the total number of transactions in D and $|TrChanged(D')|$ is the total number of changed transactions in D'.

Artificial cost (AC) is the side effect to evaluate the amount of infrequent itemsets turned into frequent itemsets after the sanitization operation. The AC is calculated as:

TID	Transaction	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)

Itemset	Support
F	62.5%
A	62.5%
AB	62.5%
C	75%
BC	75%
B	87.5%

(B)

Sensitive Itemset	Sensitive Threshold
AB	30%
BC	60%
DF	20%

(C)

FIGURE 1 A, Transactional database *D*, B frequent itemsets in *D* with minimum support 60%, C, sensitive itemsets with their sensitive thresholds

$$AC = \frac{(FI(D) - FI(D'))}{FI(D)} \tag{4}$$

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

Database dissimilarity is the side effect to evaluate the amount of items removed in the sanitized database. Database dissimilarity is calculated as:

$$Dissimilarity = \frac{(I(D) - I(D'))}{I(D)} \tag{5}$$

where $I(D)$ is the total number items in original database D and $I(D')$ is the total number of items in sanitized database D'.

In many hiding algorithms the main focus is achieving zero hiding failure, minimum possible knowledge/information loss, minimum possible accuracy and minimum possible database dissimilarity. The artificial cost never occurs in item removal process, it probably occurs when transactions are deleted or new items or transactions are inserted into the database. Also when the main concern is to protect the nonsensitive knowledge then the database dissimilarity metric is not desired as the baseline metric because it does not have a direct relation with information loss. The quality of the victim item is more important than its quantity when the amount of information loss is tried to be minimized. The quality in this case is selecting the victim item having low correlation with other items in the database. In this article we consider accuracy instead of database dissimilarity and do not measure artificial cost since our algorithm do not cause any.

3 | RELATED WORK

Privacy preserving data mining research concentrates on data (input) hiding or knowledge (output) hiding. The objective of data or input hiding is removing confidential and proprietary information inside the data before it is released [25]. On the other hand, the purpose of knowledge or output hiding task is sanitization of data in a way that disclosure of private and confidential knowledge becomes impossible after it is released. Privacy preserving association rule or itemset hiding is a subtask and NP-hard problem of knowledge hiding [7]. There are many approaches proposed to preserve the privacy of association rules and sensitive itemsets in the literature. These approaches are categorized as border based [27,34], exact [8,16,26] reconstruction based [1,9], cryptography based [24], and heuristic based [20,30,32] sanitization approaches. The border-based approaches separate the nonfrequent and frequent itemsets with a border and perform the sanitization operation by revision on the border. Exact approaches transform the frequent itemset hiding problem into constraint satisfaction problem (CSP) and then they use integer programming for solving the CSP. Reconstruction based frequent itemset hiding approaches first conceal the sensitive itemsets and then they generate a new database from the sanitized data. Cryptography based approaches are commonly utilized when association rule mining is outsourced to a cloud data. These approaches secure the sensitive data and patterns by encryption and decryption processes. 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 therefore in the literature majority of the studies are based on heuristic approaches [35]. Heuristic approaches hide sensitive itemsets either by decreasing support of sensitive itemsets or giving uncertainty to the support of sensitive itemsets. The first kind of approaches is called distortion based and the latter is called blocking based.

Table 1 gives a comparative summary of distortion-based heuristic sanitization algorithms. As one can notice hiding strategy of a sanitization algorithm may differ according to its objective of hiding (itemset or association rule), its selection of transaction to be distorted (degree, length, trial, and weight), its selection of item to be distorted (conflict, support, weight, and trial), its consideration of item grouping inside sensitive itemsets or their threshold definition.

TABLE 1 Comparison of distortion-based heuristic algorithms

		Victim item selection				Transaction selection					
Algorithm	**Hide**	**Cover**	**Support**	**Weight**	**Trial**	**Degree**	**Size**	**Trial**	**Weight**	**Item grouping**	**Multiple threshold**
PGBS [31]	Itemset	✓				✓				✓	✓
TTBS [20]	Itemset	✓				✓				✓	✓
SWA [30]	Itemset		✓			✓				✓	✓
MICF [22]	Itemset	✓							✓	✓	
IGA [29]	Itemset	✓				✓				✓	
Aggregate [5]	Itemset							✓		✓	
Disaggregate [5]	Itemset				✓			✓		✓	
Hybrid [5]	Itemset				✓			✓		✓	
SIF-IDF [18]	Itemset		✓						✓		
Algorithm 2b [36]	Itemset		✓						✓		
Algorithm 2c [36]	Itemset						✓				
HSARWI [14]	Rule				✓				✓	✓	
FHSAR [37]	Rule		✓						✓	✓	
RelevanceSorting [10]	Rule			✓					✓		
PDA [33]	Rule				✓				✓		

Some sanitization approaches select the transactions for modification according to their size where transaction size is the number of items a transaction contains. The number of association rule or frequent itemset combinations may decrease with the size of the transaction. The SWA [30] algorithm removes victim item from sensitive transactions with the shortest size. Verykios et al [36] propose 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 [36] hides sensitive itemsets by removing the item having maximum support from smallest size sensitive transactions and the Algorithm 2c [36] hides sensitive itemsets by removing sensitive itemsets from smallest size sensitive transactions.

Another type of distortion-based sanitization methodology is modifying transactions according to number of itemsets or rules they contain. By this way it is possible to reduce the distortion on nonsensitive knowledge. Item conflict first (MICF) [22] algorithm selects transactions according to how many sensitive itemset a transaction contains and then deletes the item having maximum degree of conflict from these transactions. The fast hiding sensitive association rules (FHSAR) [37] 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. The algorithm SIF-IDF [18] is inspired from TF-IDF (term frequency-inverse document frequency). The SIF-IDF calculates the relation degree of a transaction with sensitive itemsets and modifies transactions having maximum relation degree. RelevanceSorting [10] modifies transactions containing minimum number of nonsensitive itemsets. HSARWI [14] algorithm selects sensitive transactions with the same procedures as in FHSAR algorithm; two algorithms differ in victim item selection.

Another group of distortion-based sanitization algorithms determine the transactions for modification and items to be removed in a trial and error manner. These kinds of approaches try every possible sensitive transaction selection and item modification for finding the best one. The Aggregate [5] algorithm selects sensitive transactions which gives impact on minimum number of nonsensitive itemsets and at the same time gives impact on maximum number of sensitive itemset, then it removes the selected transactions from database. Disaggregate [5] approach removes items from sensitive transactions whose removal effect least number of nonsensitive itemsets and at same time effect maximum number of sensitive itemsets. The Hybrid [5] approach chooses the sensitive transactions for modification with using Aggregate approach then modifies selected transactions with using the Disaggregate approach. Priority-based distortion algorithm (PDA) [33] assigns priority to each sensitive transaction by calculating how many nonsensitive rules will be affected for all possible item removal of a sensitive rule contained in a sensitive transaction.

One of the distortion-based sanitization strategy is grouping sensitive itemsets and then modifying transactions containing each different groups of sensitive itemsets. In this way it is possible to conceal multiple sensitive itemset at once. The sensitive itemsets can be grouped according to the item common at each sensitive itemset. Oliveria et al [29] proposed four sanitization algorithms and among these algorithms the IGA algorithm introduced the multiple sensitive itemset hiding concept. Template table sanitization algorithm (TTBS) [20] overcomes the "overlapping groups problem" faced in IGA with using a table called template table. The PGBS [31] employs a graph based data structure to speed up the hiding process. As IGA, PGBS also 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.

According to Table 1, only few of the existing distortion-based heuristic algorithms focus on hiding sensitive itemset under multiple thresholds [18,29,31]. As the size of the database increases the SWA [30] has high execution time because it sorts each transaction in the given database according to its length. The TTBS [20] employs a table called template table to generate combinations of different sensitive itemsets sharing common item however these combinations may not cover the optimum sensitive itemset combination although there exists. Both PGBS [31] and TTBS [20] algorithms designed for uncovering transactions containing maximum number of sensitive itemsets, but they are unable to uncover them if these sensitive itemsets do not share any common item. Also none of these two algorithms dynamically adjust the degree of conflict of items or sensitive transactions whenever a sensitive itemset is hidden from the database and this might result in preventing to find the optimal hiding solution.

4 | ITEMSET-ORIENTED PSEUDO GRAPH BASED SANITIZATION

In this section the proposed frequent itemset hiding algorithm-IPGBS is presented. This algorithm is similar to PGBS [31] since it uses a pseudo graph structure and hides sensitive itemsets by decreasing their support under predefined sensitive thresholds. However the content of the graph and strategies for choosing the transactions and items for modification are different.

IPGBS keeps only sensitive itemsets and the transactions containing those in pseudo graph structure in order to reduce execution time and memory whereas previous algorithm PGBS keeps all items and transactions in the pseudo graph data structure. IPGBS tries to find out optimal sanitization solution by modifying least number of sensitive transactions whereas PGBS focuses on the modification of common items of sensitive itemsets.

General flow of IPGBS is illustrated in Figure 2. As it can be seen from the figure IPGBS uses three important data structures: IPG, sensitive count table (SCT), and sanitization table (ST). First, the sensitive itemsets in each sensitive transaction of the given database *D* is converted to IPG. Second, by using the IPG, support count of each sensitive itemset is calculated and SCT that stores the support count decrease required for each sensitive itemset is created. Next, the ST that stores the result of the sanitization process is prepared by using the IPG and SCT. Finally, necessary modification indicated in ST is applied to the original database and the sanitized database is generated.

4.1 | Creating the itemset-oriented pseudo graph

This section introduces the data structure IPG. The IPG is a directed graph with allowing multiple edges and loops. Each vertex *V* of IPG represents sensitive itemset and each edge *E* represents the set of transaction ids that contain the sensitive itemsets on the path between starting vertex and its direct successor. The IPG makes it possible to find out transactions containing maximum number of sensitive itemsets by uncovering longest paths starting from a predefined origin vertex. The edges of IPG are directed for avoiding to revisit the same vertices while discovering the longest paths of the graph.

To clarify the construction of IPG, the process of converting the database *D* given in Figure 1A into IPG is explained with an example. First each transaction in *D* are sorted alphabetically, then transaction "ABCDEF" is checked to determine whether it contains any sensitive itemset. The transaction "ABCDEF" contains the sensitive itemsets "AB," "BC," and "DF" so vertices labeled by these sensitive itemsets are created and connected with edges labeled with transaction id 1 as shown in Figure 3A. The next transaction in *D* is "BC," and it contains only one sensitive itemset "BC," because there is only one sensitive itemset, a loop labeled with transaction id 2 is created on vertex "BC" as shown in Figure 3B. The resulting IPG is shown in Figure 3C after all remaining sensitive transactions in *D* are inserted into the IPG with the same way as in Figure 3A,B. The procedure of deleting transaction from

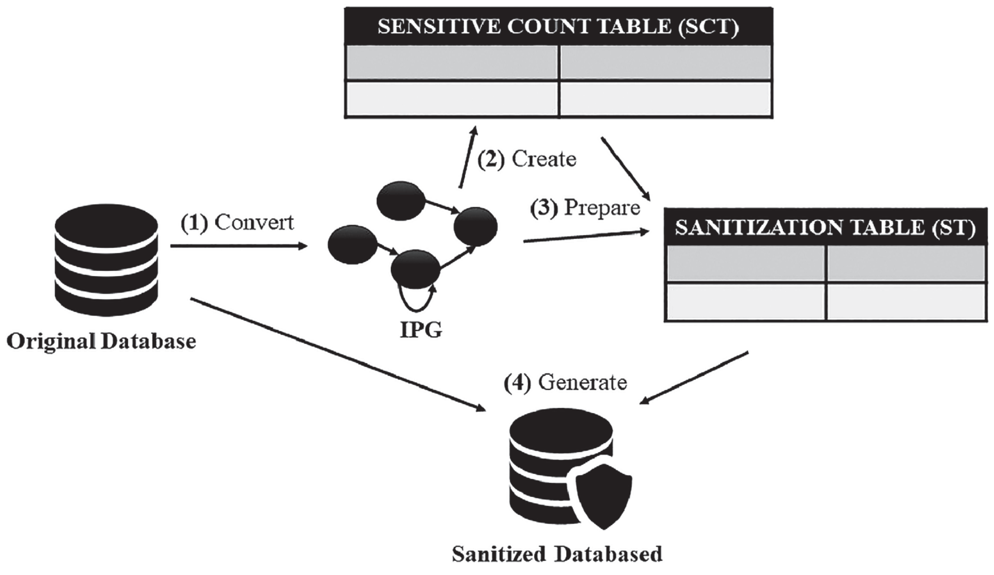


FIGURE 2 Itemset-oriented pseudo graph based sanitization algorithm flow

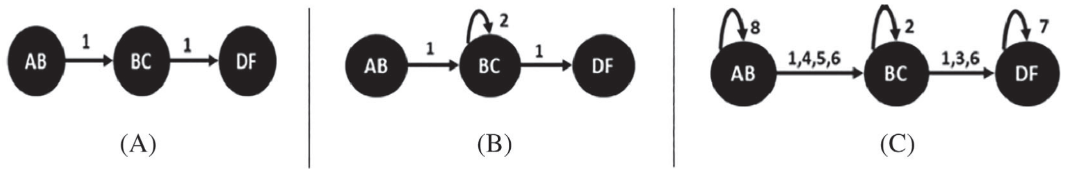


FIGURE 3 A, Itemset-oriented pseudo graph (IPG) after transaction 1 is inserted, B, IPG after transaction 2 is inserted, C, IPG after all remaining transactions are inserted

the graph includes deleting specified transactions from the outgoing edges of a given vertex.

4.2 | Sensitive itemset-oriented pseudo graph based sanitization

The IPGBS (itemset-oriented pseudo graph based sanitization) algorithm is a distortion-based frequent itemset hiding algorithm. The IPGBS algorithm depends on uncovering transactions that fully support the sensitive itemsets and then modifying them by deleting item/items in order to reduce support of sensitive itemsets. These items/items that are selected for delete operation are called victim or victims. For keeping the difference minimum between number of nonsensitive frequent itemsets in the original and sanitized database, the sensitive itemsets should hidden with minimal impact on the original database. To measure this impact, the metric "Accuracy" [26] which is the total number of transaction modification is used. The Accuracy can be minimized by minimizing the number of transaction modification during the sanitization operation. However, the Accuracy is not the only challenge for minimizing nonsensitive knowledge loss. Two sensitive itemsets are mutually dependent if they share a common item and take part in the same transaction. Modifying transactions by deleting the item that is common in mutually dependent sensitive itemsets makes it possible to reduce support of more than one sensitive itemset at one iteration. However, this may bring out the problem of reducing support of sensitive itemsets more than necessary and results in the increase of nonsensitive knowledge loss. Because one or more of the sensitive itemset that is mutually dependent may already been sanitized and deleting the common item is going to keep the support decrease for the sanitized sensitive itemset. To avoid this problem and minimize the transaction modification the IPGBS modifies transactions containing maximum number of nonsanitized sensitive itemsets. The straightforward method would be scanning the whole database *D* in the beginning of the sanitization operation and then evaluating the sensitive transactions according to their degrees. However, each time a sensitive itemset is hidden from *D*, degree of the sensitive transactions may change, so calculating degree of each sensitive transaction at the beginning and evaluating them would not be practical, the degree of each sensitive transaction should be updated after each time a sensitive itemset is hidden from the database. IPGBS uncovers sensitive transactions from the itemset-oriented pseudo graph (IPG) to reduce the execution time and also it does not uncover all sensitive transactions at once instead it uncovers sensitive transactions for each sensitive itemset one by one whenever needed. The ST represents the final hiding solution that is going to be applied to the original database.

Creating the ST process is depicted in Algorithm 1. First, for each sensitive itemset, the necessary number of support decrease is calculated and then the SCT is created (line 1) and sorted in alphabetically or numerically (line 2). The necessary number of support decrease need for hiding a given sensitive itemset X is called $X.N_{Modify}$ and it is calculated by the following equation:

$$X \cdot N_{Modify} = \lfloor scount(X) - |D| * st(X) + 1 \rfloor \qquad (6)$$

Proof. Assume that TR is the set of all transactions containing sensitive itemset X in D. Removing an item x from transaction tr where $x \subset X$ and $tr \subset TR$ results in decreasing the support of X by 1. For hiding the sensitive itemset X from database D, sufficient amount of support decrease is need for hiding X from D. To assume that X is hidden in D, the sensitive threshold of X need to be greater than percentage of transactions containing X. So we can get: $(|TR| - X \cdot N_{Modify}) / |D| < st(X) \rightarrow |TR| - |D| * st(X) < X \cdot N_{Modify} \rightarrow |TR| - |D| * st(X) + 1 \approx X \cdot N_{Modify}$ where $|TR|$ is the size of TR and is equal to $scount(X)$. Because the $st(X)$ is a decimal number, we can take floor of the equation for having minimum number of transactions that need to modification.

After the *SCT* is created, it is sorted alphabetically or numerically in decreasing order of *SI* column. Then each sensitive itemset stored in the *SI* column of *SCT* started to be selected by one by and if the selected *si* is not already sanitized while decreasing support of other sensitive itemsets the hiding operation for *si* starts (line 3). The variable *longestPath* stores the longest path in *IPG* starting from the given sensitive itemset. The *longestPath* does not contain any sensitive itemset that is already sanitized because in the opposite case support of some sensitive itemset may decreased more than necessary and as a result nonsensitive frequent itemsets in the resulting sanitized database may be effected. After the longest path containing maximum number of strong sensitive itemsets determined by using depth first search traversal technique (line 4), the transactions containing this path are extracted from *IPG* with not exceeding the N_{Modify} value of the current sensitive itemset (line 5). Next the item having the maximum cover degree among items in *longestPath* is selected as victim (line 6). If there exists more than one item having the same cover degree, the item with maximum support count is selected. If there still is more than one, then the victim item is selected randomly. After both transactions and victim item are determined, the *IPG* is updated by removing these transactions from outgoing edges of the sensitive itemset that is being sanitized which is for avoiding to modify the same transactions more than once in the next iteration (line 7). Then these transactions and victim items are inserted into the ST. If sensitive itemsets stored in *longestPath* contain the victim item, then N_{Modify} values of these sensitive itemsets are decreased by the number of transactions uncovered (line 9). If the selected victim item is not common in any sensitive itemset of the longest path then these sensitive itemsets are inserted into the variable *notContain* (line 10) and a new victim item is selected for these sensitive itemsets (line 13). Then this victim item and sufficient number of transactions from previously uncovered transactions (line 13) are inserted into the ST (line 15). The algorithm continues to generate the longest path and select victim item and transactions till the given sensitive itemset *si* is sanitized or until there does not exist any new longest path for *si*. If there are inadequate number of transactions uncovered to hide the sensitive itemset *si* and there is no more different longest path left for the vertex *si* then this means that the given sensitive itemset appear as the last element in most or all sensitive transactions when sensitive itemsets are sorted in alphabetically or numerically. As a result there would not be sufficient number of longest paths different from the self-loop of vertex *si* in *IPG*. The algorithm solves this problem by selecting the victim item among items in *si* having maximum support count value (line 18) and uncovering sufficient number of transactions from the incoming edges of the given sensitive itemset *si* (line 19).

In the worst case scenario each transaction in database D is sensitive and every item in each transaction is a subset of at least one of the sensitive itemset. Computational complexity of representing each sensitive itemset the graph data structure is $O(|SI|*|D|)$ in the worst case where $|SI|$ is the number of sensitive itemsets and $|D|$ is the total number of transactions in D.

The hiding process uncovers longest paths from predefined starting vertices in a depth first search fashion so it takes $O(|V| + E|)$ computational complexity, where $|V|$ is the total number of vertices in IPG and $|E|$ is the total number of edges in IPG. As a result identifying longest paths for all sensitive itemsets takes $O(|SI| * |V| + |E|)$ computational time complexity in the worst case scenario.

For illustration suppose the transnational database D, sensitive itemsets with their sensitive thresholds and the IPG that represents D are given as in Figures 1A,C and 3C, respectively. First the IPGBS algorithm creates the SCT and sorts the SCT according to the *SI* values as in Table 2. Next the algorithm selects the sensitive itemset "AB" from SCT and finds the longest path that starts from the vertex *AB* in *IPG*. The resulting longest path is *ABCDF* and contains three strong sensitive itemsets (*AB*, *BC*, and *DF*).

Algorithm 1 Creating sanitization table

```
INPUT: Itemset Pseudo Graph IPG, Sensitive Thresholds of Itemsets
OUTPUT: Sanitization Table ST, Sensitive Count Table SCT
1: Calculate $N_{Modify}$ of each sensitive itemset and put them into SCT
2: Sort SCT in alphabetically or numerically increasing order of SI column
3: for every sensitive itemset si ∈ SCT where si.$N_{Modify}$ > 0 do
4:     for every longestPath starting from vertex si ∈ IPG do
5:         transactions ← at most si.$N_{Modify}$ of transactions on longestPath in IPG
6:         victim ← maximum cover degree item in longestPath
7:         For all vertices v⊂ longestPath, Remove transactions from outgoing edges of v
8:         Insert victim and transactions pair into ST
9:         For all sensitive itemset s in longestPath where victim ⊂ s Reduce s.$N_{Modify}$ by |transactions|
10:        For all sensitive itemsets s in longestPath where victim⊄ s, Add s to the variable notContain
11:    end for
12:    while notContain is not empty do
13:        victim ← maximum cover degree item in notContain
14:        For all sensitive itemsets s in longestPath where victim ⊂ s Reduce s.$N_{Modify}$ by |transactions|
15:        Insert victim and transactions pair into ST
16:    end while
17:    if si.$N_{Modify}$>0 then
18:        victim ← maximum support count item in si
19:        transactions ← at most si.$N_{Modify}$ of transactions on incoming edge of vertex si in IPG
20:        Insert victim and transactions pair into ST
21:    end if
22: end for
```

Transactions containing the path *ABCDF* are 1 and 6, so these transactions are added to the transactions variable and the item *B* is selected as victim because it is contained in both *AB* and *BC*. Then transactions 1 and 6 are deleted from IPG as shown in Figure 4A and the victim item *B* and transactions 1 and 6 pair is inserted to the ST. The sensitive itemset *DF* in *ABCDF* does not contain the victim item *B* so the item "F" is selected as victim item among items "*D*" and "*F*" because "*F*" has the maximum support count (scount(*F*) = 6) and the victim item "*F*" with transactions 1 and 6 pair is inserted into the ST. The next longest path containing maximum number of nonsanitized sensitive itemsets starting from vertex *AB* is *AB*, the longest path *ABC* is ignored because it contains already sanitized sensitive itemset *BC*. The item *A* is selected randomly as victim among the items of *AB* because both support count of *A* and *B* are the same (supportCount(A) = 5, supportCount(B) = 5). The victim *A* and transaction eight pair is added to the ST and then the *IPG* is updated by removing transaction 8 from vertex *BC* as shown in Figure 4B. The next none sanitized sensitive itemset in *SCT* is *DF* and the longest path starting from the vertex *DF* is *DF* with transaction 7. Finally the victim *D* and transaction 7 pair is inserted into *ST* and the *IPG* is updated as in Figure 4C. The resulting *ST* is shown in Table 3, where this table represents the final sanitization solution for database *D*.

TABLE 2 Support count table

SID	SI	N_{Modify}
0	AB	3
1	BC	2
2	DF	3

TABLE 3 Sanitization table

Victim	Transactions
B	1,6
F	1,6
A	8
D	7

5 | PERFORMANCE EVALUATION

The IPGBS algorithm is compared with similar two: TPGBS and PGBS [31] algorithms. The TPGBS algorithm is similar to the hiding process of DynamicPGBS algorithm proposed in [32]. All three algorithms are designed for multiple sensitive support thresholds and use pseudo graph data structure. Also they try to modify minimum amount of transactions for reducing the information loss. Table 4 illustrates the main differences of PGBS, TPGBS and IPGBS algorithms. PGBS represents each transaction of the original database as pseudo graph, whereas in TPGBS and the IPGBS only sensitive transactions are represented as pseudo graph. Vertices of the pseudo graph in PGBS and TPGBS are items whereas in IPGBS vertices are sensitive itemsets. The objective of all PGBS, TPGBS, and IPGBS algorithms is to minimize the nonsensitive knowledge loss by modifying minimum number of transactions. While both PGBS and TPGBS algorithms hide multiple

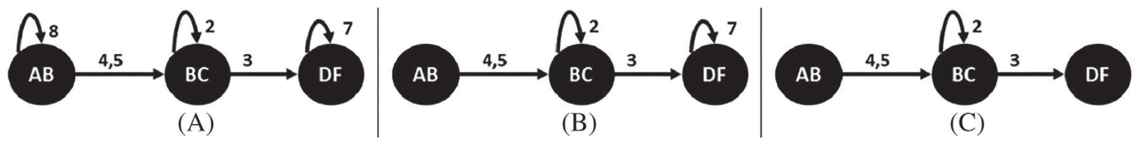


FIGURE 4 A, Itemset oriented pseudo graph (IPG) after transaction 1,6 are removed from vertices "AB,""BC," and "DF," B, IPG after transaction 8 is removed from vertex "AB," C, IPG after transaction 7 is removed from vertex "DF"

TABLE 4 Comparison of pseudo graph based sanitization (PGBS), transaction-oriented PGBS (TPGBS) and itemset oriented PGBS (IPGBS) algorithms

	PGBS	TPGBS	IPGBS
Starts modifying transactions	Maximum cover degree	Maximum cover degree	Maximum cover degree — minimum sanitized itemsets
Content of the vertex	All items	All items of sensitive transactions	Sensitive itemsets
Content of the edge	Transaction ids of all items	Transaction ids of sensitive transactions	Transaction ids of sensitive itemsets
Objective	Information loss	Information loss	Memory and information loss

sensitive itemsets at once with modifying sensitive transactions containing maximum number of sensitive itemsets whereas IPGBS algorithm modifies sensitive transactions containing maximum number of nonsanitized sensitive itemsets.

Four series of experiments are performed to measure the execution time, information loss, accuracy, and total memory consumption. The artificial cost is not measured because it only occurs when number of transactions in the database is changed or when new items are inserted in to the transactions, this is not the case in TPGBS, PGBS and IPGBS. Besides the side effect database dissimilarity is not measured because the amount of item removal process does not have a direct relation with the information loss and in additionally the TPGBS, PGBS, and IPGBS focus on reducing the nonsensitive knowledge loss. All the experiments are conducted on a computer with Intel core i7-5500 2.4 GHZ processor and 8 GB of RAM running on a Windows 10 operating system. The algorithms are implemented with Visual Studio C#.NET 2015. In all test runs it is ensured that the system state is similar and gives close results when repeated. The experiments are conducted using 2 real databases; Connect, Kosarak which are available through UCI repository [6] and 2 synthetic databases; SyntheticDense and SyntheticSparse. The SyntheticDense and SyntheticSparse databases are generated by using the IBM quest database generator [13]. The characteristics of the databases such as size, number of different items, average transaction length, shortest transaction length, longest transaction length and density are given in Table 5. The density of a database is the ratio of the average transaction length to the number of different items in the database. Dense databases contain similar transactions where each transaction has nearly the same length and contain nearly the same items. During execution tests whenever frequent itemset discovery is required MISFP-Growth [11] and MIS-Eclat [12] algorithms are used since they are devised for itemset mining under multiple support thresholds as well.

A set of 10 to 100 sensitive itemsets are selected from each database and different sensitive thresholds are assigned to them. First, itemsets in each database are partitioned into 5 support bins where each bin contains nearly the same number of itemsets, and then 2 to 20 itemsets are randomly selected from each support bin as sensitive itemset. The sensitive threshold for each sensitive itemset is assigned as the minimum support in the corresponding support bin. The support bins for each database are given in Table 6. In each experiment, the number of sensitive itemsets range from 10 to 100 by increments of 10. The size of each database is remained in the original size at each measurement.

5.1 | Execution time

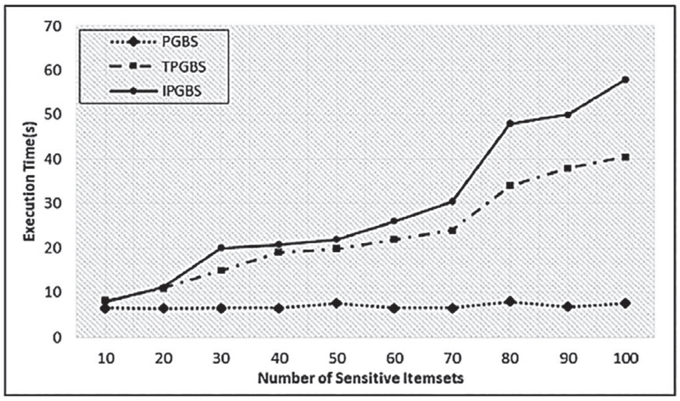
In Figure 5A,B, the execution time of PGBS, TPGBS and IPGBS algorithms on two dense databases; Connect and SyntheticDense are shown respectively. It can be seen that PGBS algorithm has the least execution time on these two databases while the TPGBS algorithm has the second lowest and the IPGBS algorithm has the highest execution time. This is because dense databases contain too much distinct items and IPGBS and TPGBS algorithms consume

TABLE 5 The characteristics of databases

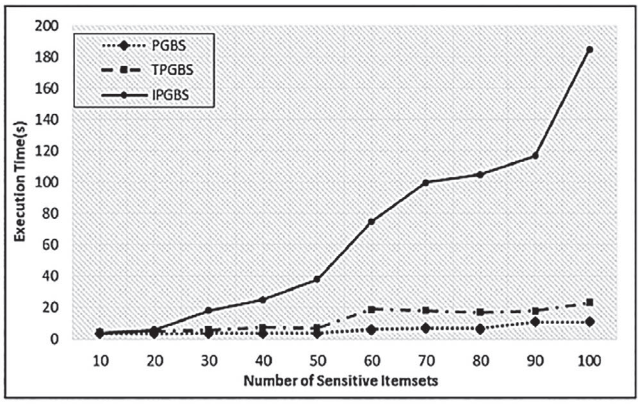
Database Name	Size	Different items	Average length	Shortest length	Longest length	Density (%)
Connect	67,557	129	43	43	43	33.4
Kosarak	990,002	41,270	8.1	1	2498	0.0196
SyntheticDense	29,166	99	43.09	2	44	43.5
SyntheticSparse	28,417	9,479	11.48	2	11	0.1212

TABLE 6 Support bins of each database

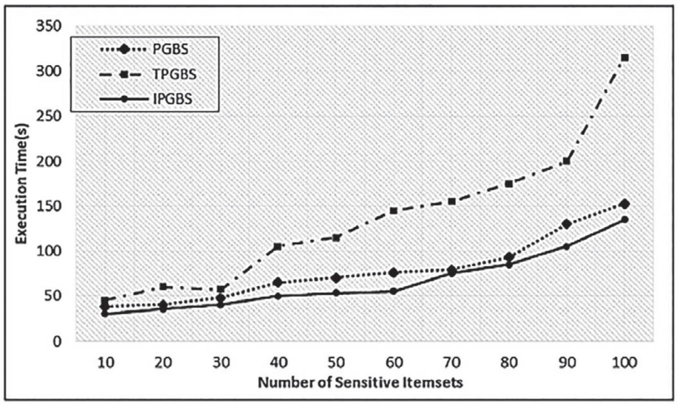
Bin	Connect	Kosarak	SyntheticDense	SytheticSparse
	Support range	Support range	Support range	Support range
1	(0.001906, 0.001936]	(0.001906, 0.001936]	(0.3, 0.308]	(0.0002, 0.000024]
2	(0.001936, 0.001971]	(0.001936, 0.001971]	(0.308, 0.3185]	(0.00024, 0.00028]
3	(0.001971, 0.002029]	(0.001971, 0.002029]	(0.3185, 0.3339]	(0.00028, 0.00035]
4	(0.002029, 0.00215]	(0.002029, 0.00215]	(0.3339, 0.3619]	(0.00035, 0.00049]
5	(0.002015, 1]	(0.002015, 1]	(0.3619, 0.9546]	(0.00049, 0.038]



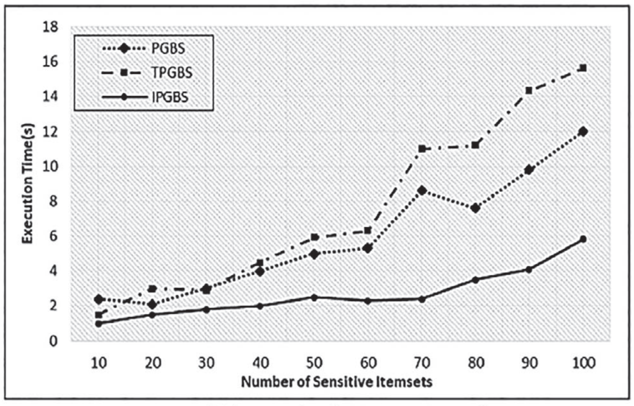
(A) Connect Database



(B) SyntheticDense Database



(C) Kosarak Database



(D) SyntheticSparse Database

FIGURE 5 Execution time with fixed database size and varying number of sensitive itemsets

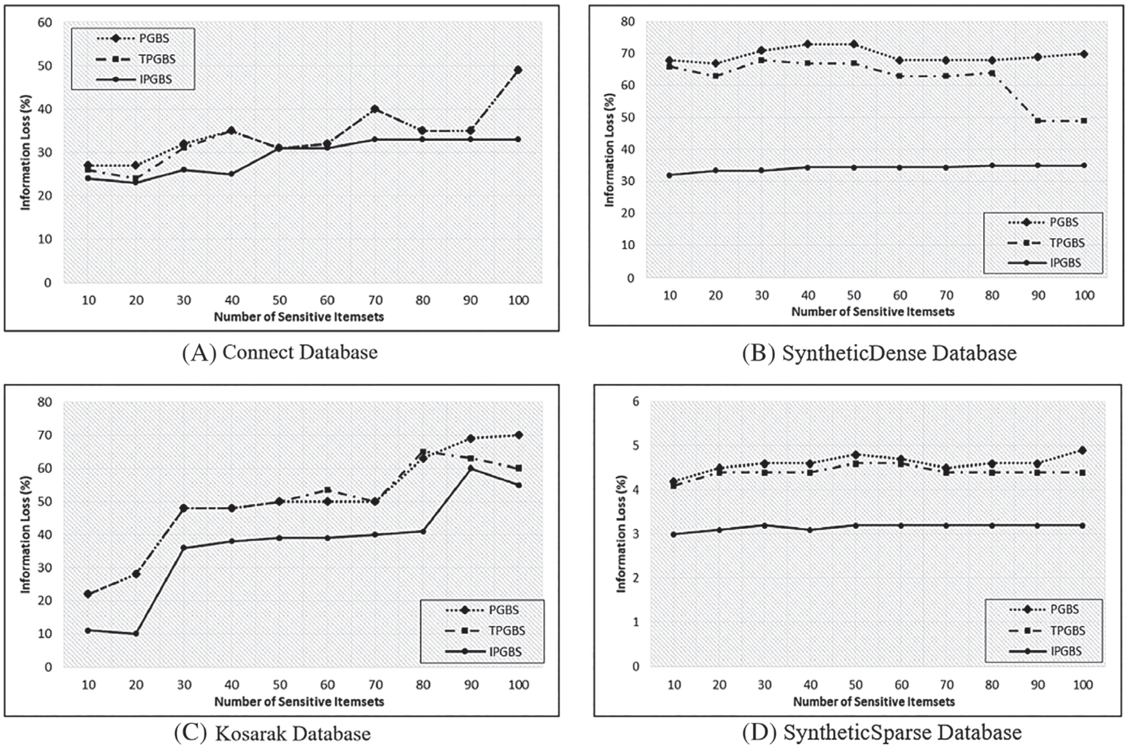


FIGURE 6 Information loss with fixed database size and varying number of sensitive itemsets

most of the time for analyzing the items in each transaction while building the Pseudo Graph. In Figure 5C,D execution time of PGBS, TPGBS, and IPGBS algorithms are shown on two sparse databases; Kosarak and SyntheticSparse respectively. These two figures show that IPGBS has the lowest execution time on sparse databases. The PGBS has the second lowest execution time on both Kosarak and SytheticSparse databases.

5.2 | Information loss

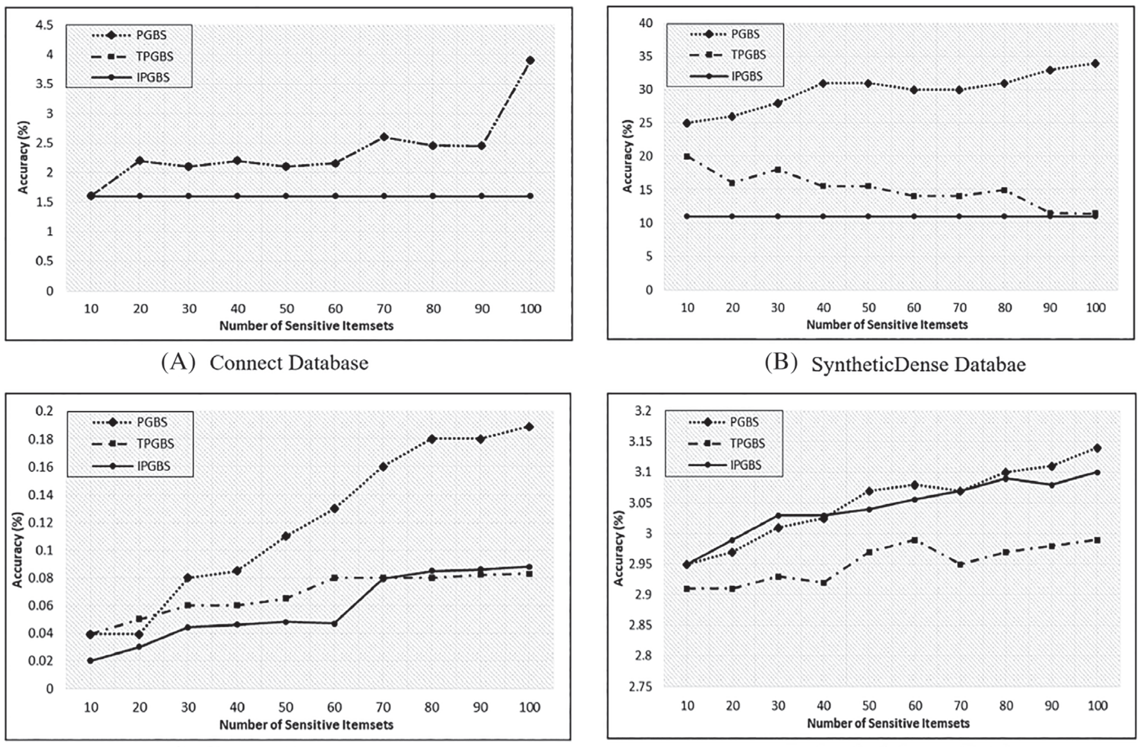
Figure 6A-D shows results for information loss in percentage. These figures demonstrate that IPGBS algorithm outperforms PGBS and TPGBS algorithms on both dense and sparse databases. This is reasonable because the aim of IPGBS algorithm is to modify sensitive transactions containing maximum number of nonsanitized sensitive itemset for minimizing the amount of transaction modification. As the number of transaction modification decreases the information loss of IPGBS decreases.

5.3 | Accuracy

Figure 7A,B shows results for the metric accuracy in percentage for dense databases Connect and SyntheticDense. These figures show that the IPGBS algorithm modifies less number of transactions than both PGBS and TPGBS algorithms on dense databases. Also for the IPGBS algorithm it can be observed that there is no relation between the number of sensitive itemsets and the number of transactions modified on dense databases. Figure 7C,D shows results for the metric accuracy in percentage for sparse databases Kosarak and SyntheticSparse databases. It can be seen from these figures that IPGBS has the best accuracy result till the number of sensitive itemsets are 70 on Kosarak database. The TPGBS has the best accuracy result on SyntheticSparse database.

5.4 | Memory allocation

The total memory allocation of each sanitization algorithms; PGBS, TPGBS, and IPGBS are shown in Figure 8. On dense database as in Figure 8A,B it can be observed that the memory allocation of the IPGBS algorithm is linear with the number of sensitive itemsets while this is not the case on sparse databases as in Figure 8C,D. This is due to fact that sensitive itemsets on dense databases has more support than sparse databases and as the IPGBS stores the transaction information of each sensitive itemset, the memory consumption increases with the number of sensitive itemsets. On sparse databases as in Figure 8C,D both TPGBS and IPGBS has the smallest memory allocation compared to PGBS algorithm.



(A) Connect Database

(B) SyntheticDense Databae

(C) Kosarak Database

(D) SyntheticSparse Database

FIGURE 7 Accuracy with fixed database size and varying number of sensitive itemsets

5.5 | Discussion of the results

Although the PGBS, TPGBS, and IPGBS algorithms are designed for modifying transactions containing maximum number of sensitive itemsets, the PGBS and TPGBS can only find these transactions if the sensitive itemsets in a given transaction contain common item. The IPGBS algorithm on the other hand uncovers transactions containing maximum number of nonsanitized sensitive itemsets by using IPG and uncovering transactions does not directly depend on number of common items in the given set of sensitive itemsets. The IPGBS is designed for keeping the number of transaction modification at minimum level for decreasing the loss of nonsensitive knowledge during the sanitization operation. The correlation between items in dense databases is higher than sparse databases and this increases the possibility of uncovering transactions containing more than one nonsanitized sensitive itemsets. 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 creation for both IPGBS and TPGBS algorithms. When the density of a given database is high the execution time increase of IPGBS is proportional to the number of sensitive itemsets. In the performance evaluation the loss of nonsensitive knowledge is measured with the information loss metric and as it can be seen that the IPGBS has the minimum information loss compared to TPGBS and PGBS. The IPGBS algorithm has the best accuracy results on dense databases while this is not the same on sparse databases. This is because as the density of a given databases increases the probability of uncovering transactions containing more than one nonsanitized sensitive itemsets increases and vice versa for sparse databases. The TPGBS algorithm achieves the best accuracy results on sparse databases because it does not check whether all sensitive itemset are already sanitized in the uncovered transactions for modification. The memory allocation of IPGBS is better than both PGBS and TPGBS algorithms when the given database is sparse.

6 | CONCLUSION

This paper presents a new distortion-based frequent itemset hiding algorithm. The main idea behind the algorithm is keeping the nonsensitive knowledge loss at minimum level during the sanitization operation by altering minimum number of transactions with minimum memory. To decrease the cost of directly searching nonsanitized frequent itemsets on the database, the algorithm represents the sensitive itemsets in the database as the form of IPG data structure where all sensitive itemsets are represented as nodes and all sensitive transactions

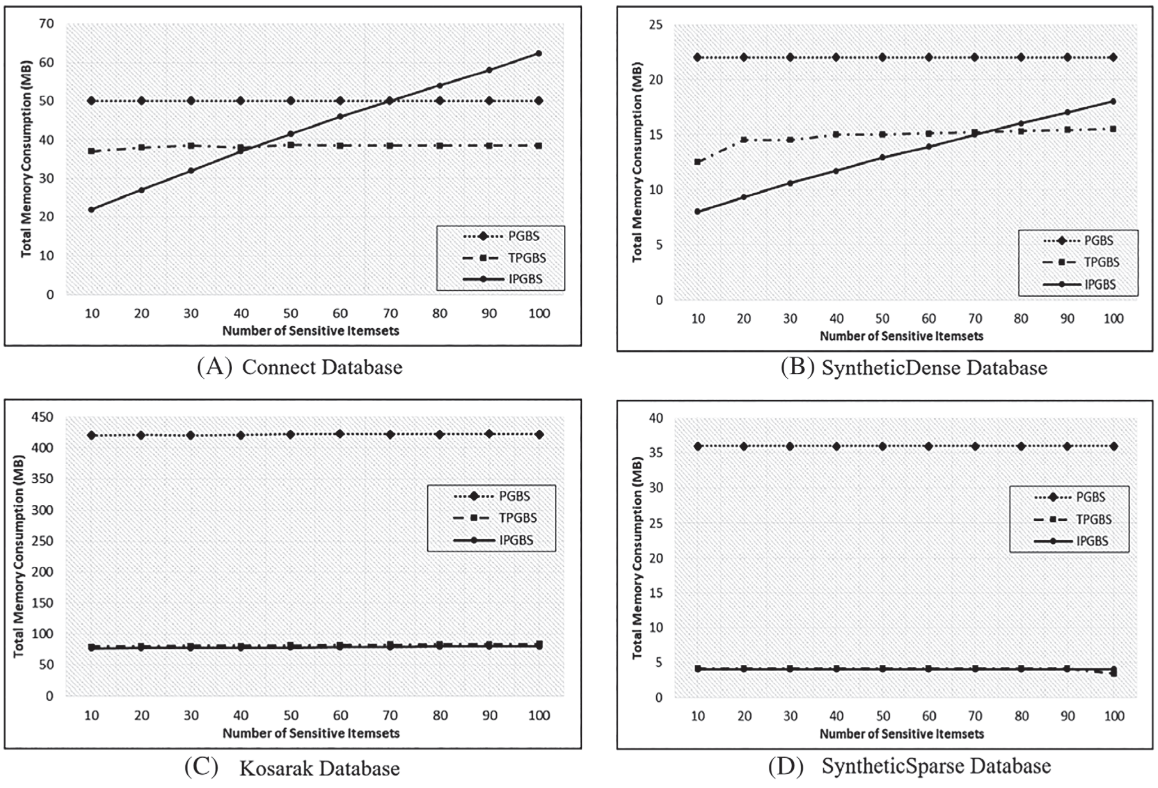


FIGURE 8 Total memory allocation with fixed database size and varying number of sensitive itemsets

are represented as edges connecting these sensitive itemsets. The IPGBS algorithm on the other hand uncovers transactions containing maximum number of nonsanitized sensitive itemsets by using the IPG where it eliminates the dependency of uncovering transactions on the given set of sensitive itemsets. The results show that the IPGBS algorithm is more efficient in terms of nonsensitive frequent itemset loss on both dense and sparse databases. It has considerable better results in terms of number of transactions modified, number of items deleted, execution time and total memory allocation.

ORCID

Belgin Ergenç Bostanoğlu https://orcid.org/0000-0001-6193-9853
Ahmet Cumhur Öztürk https://orcid.org/0000-0002-2677-3269

REFERENCES

1. O. Abul, *Hiding co-occurring frequent itemsets*, in *Proceedings of the 2009 EDBT/ICDT Workshops*, ACM, New York, 2009, 117–125.
2. N. Abuzayed and B. Ergenc, *Dynamic itemset mining under multiple support thresholds*, FSDM 293 (2016), 141–148.
3. N. N. Abuzayed and B. Ergenç, *Comparison of dynamic itemset mining algorithms for multiple support thresholds*, in *Proceedings of the 21st International Database Engineering & Applications Symposium*, ACM, New York, 2017, 309–316.
4. R. Agrawal et al., *Fast algorithms for mining association rules*, in *Proceedings of the 20th International Conference on Very Large Data Bases*, Chile, 1994, 487–499.
5. A. Amiri, *Dare to share: Protecting sensitive knowledge with data sanitization*, Decision Support Syst. 43 (2007), 181–191.
6. A. Asuncion and D. Newman, UCI machine learning repository, 2007. Available at http://archive.ics.uci.edu/ml
7. M. Atallah et al., *Disclosure limitation of sensitive rules*, in *Proceedings 1999 Workshop on Knowledge and Data Engineering Exchange (KDEX'99) (Cat. No. PR00453)*, IEEE, New York, 1999, 45–52.
8. T. Ayav and B. Ergenc, *Full-exact approach for frequent itemset hiding*, Int. J. Data Warehous. Mining 11 (2015), 49–63.
9. R. K. Boora, R. Shukla, and A. K. Misra, *An improved approach to high level privacy preserving itemset mining*, arXiv preprint arXiv:1001.2270, 2010.
10. P. Cheng et al., *Privacy preservation through a greedy, distortion-based rule-hiding method*, Appl. Intell. 44 (2016), 295–306.
11. S. Darrab and B. Ergenç, Frequent pattern mining under multiple support thresholds, WSEA Transactions on Computer Research 4, (2016), 1–10.
12. S. Darrab and B. Ergenc, *Vertical pattern mining algorithm for multiple support thresholds*, in *International Conference on Knowledge Based and Intelligent Information and Engineering*, France, 2017, 417–426.
13. I. Q. M.-B. S. Data, *Generator*, Bhalodiya, Dharmesh, 2014.
14. M. Dehkordi and M. Dehkordi, *Introducing an algorithm for use to hide sensitive association rules through perturbation technique*, J. AI and Data Min. 4 (2016), 219–227.

15. P. Fournier-Viger et al., *A survey of itemset mining*, WIRE Data Min. Knowl. Discov. 7 (2017), e1207.
16. A. Gkoulalas-Divanis and V. S. Verykios, *A parallelization framework for exact knowledge hiding in transactional databases*, Springer, IFIP International Information Security Conference, Berlin, 2008, 349–363.
17. J. Han, J. Pei, and Y. Yin, *Mining frequent patterns without candidate generation*, ACM Sigmod Record 29 (2000), 1–12.
18. T.-P. Hong et al., *Using tf-idf to hide sensitive itemsets*, Applied Intelligence 38 (2013), 502–510.
19. R. U. Kiran and P. K. Reddy, *Novel techniques to reduce search space in multiple minimum supports-based frequent pattern mining algorithms*, in *Proceedings of the 14th international conference on extending database technology*, ACM, New York, 2011, 11–20.
20. Y.-P. Kuo, P.-Y. Lin, and B.-R. Dai, *Hiding frequent patterns under multiple sensitive thresholds*, in *International Conference on Database and Expert Systems Applications*, Springer, Berlin, 2008, 5–18.
21. S. Li et al., *Privacy preserving frequent itemset mining: Maximizing data utility based on database reconstruction*, Comput. Secur. 84 (2019), 17–34.
22. Y.-C. Li, J.-S. Yeh, and C.-C. Chang, *MICF: An effective sanitization algorithm for hiding sensitive patterns on data mining*, Advanced Engineering Informatics 21 (2007), 269–280.
23. B. Liu, W. Hsu, and Y. Ma, *Mining association rules with multiple minimum supports*, ACM, Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining, 1999, 337–341.
24. F. Liu, W. K. Ng, and W. Zhang, *Encrypted association rule mining for outsourced data mining*, in *2015 IEEE 29th International Conference on Advanced Information Networking and Applications*, IEEE, New York, 2015, 550–557.
25. G. J. Matthews et al., *Data confidentiality: A review of methods for statistical disclosure limitation and methods for assessing privacy*, Stat. Surv. 5 (2011), 1–29.
26. S. Menon, S. Sarkar, and S. Mukherjee, *Maximizing accuracy of shared databases when concealing sensitive patterns*, Inform. Syst. Res. 16 (2005), 256–270.
27. G. V. Moustakides and V. S. Verykios, *A maxmin approach for hiding frequent itemsets*, Data Knowl. Eng. 65 (2008), 75–89.
28. D. Oguz, B. Yildiz, and B. Ergenc, *Dma: Matrix based dynamic itemset mining algorithm*, Int. J. Data Warehous. Mining 9 (2013), 62–75.
29. S. R. Oliveira and O. R. Zaiane, *Privacy preserving frequent itemset mining*, in *Proceedings of the IEEE International Conference on Privacy, Security and Data Mining*, Vol 14, Australian Computer Society, New York, 2002, 43–54.
30. S. R. Oliveira and O. R. Zaïane, *Protecting sensitive knowledge by data sanitization*, in *Third IEEE International Conference on Data Mining*, IEEE, New York, 2003, 613–616.
31. A. C. Öztürk and B. E. Bostanoglu, Itemset hiding under multiple sensitive support thresholds, in *Ninth International Conference on Knowledge Management and Information Sharing*, Portugal, 2017, 222–231.
32. A. C. Öztürk and B. Ergenç, *Dynamic itemset hiding algorithm for multiple sensitive support thresholds*, Int. J. Data Warehous. Mining 14 (2018), 37–59.
33. E. D. Pontikakis, A. A. Tsitsonis, and V. S. Verykios, *An experimental study of distortion-based techniques for association rule hiding*, in *Research Directions in Data and Applications Security XVIII*, Springer, Berlin, 2004, 325–339.
34. X. Sun and P. S. Yu, *Hiding sensitive frequent itemsets by a border-based approach*, J. Comput. Sci. Eng. 1 (2007), 74–94.
35. A. Telikani and A. Shahbahrami, *Data sanitization in association rule mining: An analytical review*, Expert Syst. Appl. 96 (2018), 406–426.
36. V. S. Verykios et al., *Association rule hiding*, IEEE Trans. Knowl. Data Eng. 4 (2004), 434–447.
37. C.-C. Weng, S.-T. Chen, and H.-C. Lo, *A novel algorithm for completely hiding sensitive association rules*, in *2008 Eighth International Conference on Intelligent Systems Design and Applications*, Vol 3, IEEE, New York, 2008, 202–208.
38. M. J. Zaki, *Scalable algorithms for association mining*, IEEE Tran Knowl Data Eng 12 (2000), 372–390.

AUTHOR BIOGRAPHIES

Belgin Ergenç Bostanoğlu received the diploma degree in Computer Science from Middle East Technical University, Ankara, Turkey. She worked with different titles and responsibilities in companies like Kordsa, Aksa, Dusa and Tesco during 1983 and 2000. She received her masters degree in Computer Engineering from Izmir Institute of Technology, Turkey in 2002. She received her masters degree in Computer Engineering from Izmir Institute of Technology, Turkey in 2002. She continued her education and research in joint PhD program between Paul Sabatier University of Toulouse, France and Ege University, Izmir, Turkey between the years of 2002 and 2008 and received her PhD degree from both universities. She is associate professor in Department of Computer Engineering at Izmir Institute of Technology, Turkey

Ahmet Cumhur Öztürk received his PhD degree in Computer Engineering from Izmir Institute of Technology in 2018 and is currently an Instructor at Computer Aided Design and Animation Department of Aydın Adnan Menderes University. His research interests include data mining, frequent itemset mining, frequent itemset hiding and graph mining.

How to cite this article: Ergenç Bostanoğlu B, Öztürk AC. Minimizing information loss in shared data: Hiding frequent patterns with multiple sensitive support thresholds. *Stat Anal Data Min: The ASA Data Sci Journal*. 2020;13:309–323. https://doi.org/10.1002/sam.11458