Article

Hiding Sensitive Itemsets Using Sibling Itemset Constraints

Baris Yildiz [1,*], Alp Kut [1] and Reyat Yilmaz [2]

[1] Department of Computer Engineering, Dokuz Eylul University, 35160 Izmir, Turkey; alp@cs.deu.edu.tr
[2] Department of Electrical and Electronics Engineering, Dokuz Eylul University, 35160 Izmir, Turkey; reyad.yilmaz@deu.edu.tr
* Correspondence: yildiz.baris@ogr.deu.edu.tr

Abstract: Data collection and processing progress made data mining a popular tool among organizations in the last decades. Sharing information between companies could make this tool more beneficial for each party. However, there is a risk of sensitive knowledge disclosure. Shared data should be modified in such a way that sensitive relationships would be hidden. Since the discovery of frequent itemsets is one of the most effective data mining tools that firms use, privacy-preserving techniques are necessary for continuing frequent itemset mining. There are two types of approaches in the algorithmic nature: heuristic and exact. This paper presents an exact itemset hiding approach, which uses constraints for a better solution in terms of side effects and minimum distortion on the database. This distortion creates an asymmetric relation between the original and the sanitized database. To lessen the side effects of itemset hiding, we introduced the sibling itemset concept that is used for generating constraints. Additionally, our approach does not require frequent itemset mining executed before the hiding process. This gives our approach an advantage in total running time. We give an evaluation of our algorithm on some benchmark datasets. Our results show the effectiveness of our hiding approach and elimination of prior mining of itemsets is time efficient.

Keywords: frequent itemset mining; privacy-preserving data mining; sensitive itemset hiding

check for updates

Citation: Yildiz, B.; Kut, A.; Yilmaz, R. Hiding Sensitive Itemsets Using Sibling Itemset Constraints. *Symmetry* **2022**, *14*, 1453. https://doi.org/10.3390/sym14071453

Academic Editors: Tzu Chuen Lu, Wun-She Yap, Biswapati Jana and Jeng-Shyang Pan

Received: 27 May 2022
Accepted: 11 July 2022
Published: 15 July 2022

1. Introduction

Data mining is a successful tool for extracting knowledge from large amounts of data. It is efficiently applied to many fields, such as weather forecasting [1], biomedical [2], medical diagnosis [3], marketing [4], security [5], and fraud detection [6]. On the other hand, sensitive data used in data mining applications or sensitive knowledge gained from these applications may cause privacy breaches directly or through linkable private data. Privacy-preserving data mining (PPDM) arises from the need to continue performing data mining efficiently but while preserving private data or sensitive knowledge. PPDM can be divided into two as input privacy and output privacy. These are also known as data hiding and knowledge hiding, respectively. Data hiding techniques aim to preserve individual's sensitive data private and modify data mining algorithms in such a way that sensitive data cannot be inferred from the results of data mining algorithm. Achieving this requires some special techniques, including anonymization, distortion, randomization, and encryption [7]. Knowledge hiding techniques aim to preserve sensitive rules or patterns as private and modify original data in such a way that all sensitive patterns or rules stay unrevealed while remaining ones can still be discovered [8,9].

Finding frequently co-occurring items using data mining is popular among companies to discover valuable knowledge such as customer habits. Although this is very valuable alone, companies may be willing to share data for collaboration. In this way, a better understanding of discovered knowledge can be gained, which will help to make better strategies. However, the risk of disclosing sensitive relationships may increase. For example, let us consider a scenario in which a supermarket sells products of two rival companies. To collaborate and increase profits, one company offers lower prices to the supermarket. The collaborator company reveals relationships with its rival's products through data mining. Using this knowledge and campaigns, the collaborator company may monopolize certain products, which can negatively affect the rival company and the supermarket [10]. For similar situations, the stakeholders should sanitize the databases before sharing.

Approaches for privacy preservation in frequent itemset mining may be heuristic and exact in their algorithmic nature [11]. These approaches aim to modify the database so that sensitive itemsets or association rules are hidden, and non-sensitive ones are affected minimally. Heuristic ones are faster, while exact ones have fewer side effects on non-sensitive itemsets.

In this study, we propose an exact approach for frequent itemset hiding. Our motivations are (i) to hide all sensitive itemsets, (ii) to minimize side effects on non-sensitive itemsets, (iii) to use fewer constraints to lessen runtime, (iv) to bypass the need for prior mining of all frequent itemsets, and (v) to use relaxation techniques where an exact solution is not feasible. We have evaluated performance for different hiding scenarios on different datasets observing effectiveness, the number of lost itemsets as a side effect, and runtime efficiency.

This paper is organized as follows. Section 2 gives related work on frequent itemset hiding research. Section 3 provides preliminary information for itemset mining and hiding. Section 4 presents our itemset hiding approach with an example. In Section 5, the results of experiments for our approach are given. Lastly, Section 6 concludes the paper.

2. Related Work

The optimal itemset hiding is NP-hard [12]. Therefore, researchers have focused on approaches that rely on some assumptions, namely heuristic approaches. Based on heuristics, the database is modified for sanitization [10,13,14]. These techniques are efficient, scalable, and fast algorithms but may have too many lost itemsets as a side effect. Some other studies extend heuristics and use border theory [15]. The notion behind these approaches is that the itemsets on the border represent a boundary between the frequent and the infrequent itemsets. Instead of considering non-sensitive frequent itemsets during the hiding process, these approaches focus on preserving the border's quality [16,17]. Another heuristic approach uses intersection lattice theory and distance concepts to lessen the side effects of the hiding process [18]

Heuristic approaches are fast but may have side effects, and the number of non-sensitive itemsets accidentally hidden may increase. To cope with this, exact approaches deal with the problem as a Constraint Satisfaction Problem (CSP). These approaches present better solutions in terms of the number of lost itemsets but have more complexity and may have a longer runtime.

The first itemset hiding approach based on constraint programming is in [19]. In this approach, first, constraints for integer programming are defined. Solving the problem would lead us to identify the selection of transactions to be modified. Following this, heuristically, items are selected from the transactions and altered. This process continues until the selected transaction no longer supports any sensitive itemsets.

In [20], the authors defined distance measures for the sanitized database. Instead of the number of transactions, they considered the number of modified items. Minimization of this distance is accomplished by maximizing the occurrences of items of sensitive itemsets. Using the positive and negative borders and the Apriori property, constraints are defined to maximize itemset occurrences and minimize item modifications. The authors also propose an approach for the degree reduction of constraints. When the constructed CSP is not solvable, this approach removes one constraint and constructs the CSP again iteratively until the CSP is solvable.

In [21], the authors revised the previous approach and gave a two-phase iterative approach. Firstly, sensitive itemsets are hidden using the revised positive border of itemsets. Secondly, transactions are modified to support accidentally hidden itemsets. For both phases, CSP is used.

In [22], authors defined new constraints and relaxation procedures to provide an exact solution. This approach observes all frequent itemsets to ensure they are kept frequent after sanitization. On the other hand, the number of constraints and variables is extremely large, resulting in an increased runtime.

There are also some techniques for itemset hiding based on evolutionary algorithms in recent years. Since the solution is NP-hard, dealing with the problem as an optimization problem is feasible. Based on the algorithm in [23], specified transactions were deleted for sanitization. In [24], authors proposed particle swarm optimization-based algorithms, which need fewer parameters to be set compared to previous algorithms. In [25], an algorithm was proposed formulating an objective function that estimates the effect on non-sensitive rules with recursive computation.

3. Preliminaries

Frequent itemset mining has been one of the most essential and popular data mining techniques since it was first introduced in [26]. Some of the most popular algorithms are Apriori [27], Eclat [28], and FP-Growth [29]. Originating from market basket data analysis, it can be applied to many fields [6,30–32].

The basic concepts can be defined as follows. You can refer to Table 1 for used notations. Let $I = \{i_{1,}, i_2, \ldots, i_m\}$ be a set of literals, called items. Let $D = \{T_{1,}, T_2, \ldots, T_n\}$ be a dataset of transactions where each transaction T_i is a set of items in I such that $T_i \subseteq I$. Each transaction can be defined in the binary form where $d_{ij} = 1$ if the j-th item of I appears in the transaction t_i. Considering all transactions, for ease of calculation, we have a binary form of D as a matrix that is called bitmap notation. It is given in Equation (1). All items have the same level of importance; therefore, they are symmetric.

$$d_{ij} = \begin{cases} 1, & if\ I_j \in T_i \\ 0, & otherwise \end{cases} \tag{1}$$

Table 1. Notation descriptions.

Notation	Description
$\boldsymbol{D}$	Dataset
$\boldsymbol{\sim D}$	Intermediate form of dataset
$\boldsymbol{D^s}$	Sanitized dataset
$\boldsymbol{T_i}$	*i*-th transaction of dataset
$\boldsymbol{I}$	Set of items
$\boldsymbol{\sigma(X)}$	Support count of itemset *X* in dataset
$\boldsymbol{\sigma^s(X)}$	Support count of itemset *X* in sanitized dataset
$\boldsymbol{\sigma_{min}}$	Minimum support count threshold
$\boldsymbol{S}$	Set of sensitive itemsets
$\boldsymbol{Ss}$	Set of supersets of sensitive itemsets
$\boldsymbol{F}$	Set of frequent itemsets in dataset
$\boldsymbol{F^n}$	Set of non-sensitive frequent itemsets in dataset
$\boldsymbol{F^s}$	Set of frequent itemsets in sanitized dataset
$\boldsymbol{d_{ij}}$	Item of dataset in bitmap notation at *i*-th row *j*-th column
$\boldsymbol{\sim d_{ij}}$	Item of intermediate form of dataset in bitmap notation at *i*-th row *j*-th column
$\boldsymbol{d^s_{ij}}$	Item of sanitized dataset in bitmap notation at *i*-th row *j*-th column
$\boldsymbol{u_{ij}}$	Binary variable of intermediate form of dataset in bitmap notation at *i*-th row *j*-th column
$\boldsymbol{SI}$	Set of sibling itemsets
$\boldsymbol{SI(X)}$	Set of sibling itemsets of itemset *X*
$\boldsymbol{r_Y}$	Binary variable of constraint defined for itemset *Y*

Let X be a set of items where $X \subseteq I$, which we called an itemset. If $X \subseteq t_i$, then itemset X is said to be supported by transaction t_i. In other words, all items of the itemset appear in the transaction. The number of transactions in D supporting itemset X is defined as the support count of X. Benefiting from the symmetric nature of the items, the support count of itemset X in bitmap notation can be calculated as given in Equation (2).

$$\sigma(X) = \sum_{i=1}^{n} \prod_{I_j \in X} d_{ij} \tag{2}$$

If the support count of itemset X is at least equal to the minimum support count, i.e., $\sigma(X) \geq \sigma_{min}$, then itemset X is called frequent or large. The frequent itemset mining problem is to find all frequent itemsets in the database for a predefined minimum support threshold. We can define the set of all frequent itemsets F, as stated in Equation (3).

$$F = \{X \subseteq I \ : \ \sigma(X) \geq \sigma_{min}\} \tag{3}$$

Some itemsets in F may contain sensitive information. Denoting these as S, referring to sensitive itemsets, we need to modify database D into D^s in such a way that the frequent itemsets of sanitized database F^s exclude sensitive itemsets. As known from the a priori property, if an itemset is frequent, all of its subsets are also frequent. Rephrasing vice versa for the itemset hiding concept, we can say that when an itemset is sensitive, its supersets are also sensitive. Sensitive supersets Ss should also be hidden, which can be defined in Equation (4).

$$Ss = \{X \in F \ : \ \forall Y \in S, \quad X \supset Y\} \tag{4}$$

The remaining frequent itemsets are non-sensitive frequent itemsets donated by F^n, as given in Equation (5).

$$F^n = F - (S \cup Ss) \tag{5}$$

Then, we can define frequent the itemset hiding problem as modifying dataset D into D^s in such a way that F^s—frequent itemsets of sanitized dataset D^s—excludes sensitive frequent itemsets S whereas non-sensitive frequent itemsets F^n can still be mined from D^s with the same minimum support threshold. D and D^s have an asymmetric relationship since we delete some items to sanitize the dataset.

$$F^s = \{X \subseteq I \ : \ X \in F^n \ and \ \sigma^s(X) \geq \sigma_{min}\} \tag{6}$$

For an ideal sensitive itemset hiding methodology, as many of the following goals should be accomplished on the sanitized database with the same minimum support threshold.

1. Modification of the database is minimized, such that originality of the database is kept as much as possible.
2. All sensitive itemsets are hidden and do not appear in the sanitized database.
3. Supersets of sensitive itemsets are also hidden and do not appear in the sanitized database. We know from the Apriori property that this goal is also accomplished if the first goal is achieved.
4. All non-sensitive frequent itemsets appear in the sanitized database. If an itemset doesn't appear in the new database, it is called a lost itemset.
5. No new itemset appears in the sanitized database. Such itemsets are called ghost itemsets. However, approaches that delete items from the dataset naturally accomplish this goal, and no new itemset can be mined.

Goal 1 can be rewritten as accomplishing $\min(D - D^s)$. Minimization of modification for approaches use item deletion; we can say that number of items deleted should be minimized. Using the bitmap notation given in Equation (1), let us define items in the new dataset as d^s_{ij}. Then, the minimization of the number of 1s converted to 0s is the maximization of the 1s in D^s and can be defined as follows.

$$max \sum_{i,\ j} d^s_{ij} \tag{7}$$

Goal 2 can be accomplished by keeping the support count of all sensitive itemsets below the minimum support count in the new dataset.

$$\begin{gathered} \forall X \in S \\ \sigma^s(X) < \sigma_{min} \end{gathered} \tag{8}$$

Goal 3 is accomplished if goal 2 is already satisfied.

Goal 4 can be accomplished by keeping the support count of all non-sensitive itemsets at the same or above the minimum support count in the new dataset.

$$\begin{gathered}\forall X \in F^n \\ \sigma^s(X) \geq \sigma_{min}\end{gathered} \tag{9}$$

Goal 5 is satisfied if the approach uses item deletion for the sanitization method and does not add any item to the new dataset. Some approaches use reconstruction methods and may also add new transactions to the sanitized dataset. Such approaches may be exposed to this side effect.

$$\begin{gathered}\forall X \in F^s \\ X \in F^n\end{gathered} \tag{10}$$

The majority of sensitive itemset hiding approaches aim to hide sensitive itemsets while minimizing modified items in the dataset and the number of lost itemsets. They are focused on goals 1, 2, 3, and 4.

CSP Formulation

Preliminaries for sensitive itemset hiding are already given, and this process can be formulated as a constraint satisfaction problem. By satisfying goal 1, we can say that there are two kinds of constraints. The first type of constraint defined for accomplishing goal 2 is defined in Inequation (8). Other constraints are determined to achieve goal 4 given in Inequation (9). The first type is compulsory since hiding sensitive itemsets is the primary goal of frequent itemset hiding. The second type serves for the preservation of the non-sensitive frequent itemsets.

For CSP formulation, we modified the dataset into an intermediate form. Consider X as one of the sensitive itemsets. Then, all transactions supporting X should be modified to an intermediate state for constraint formulation. The items of the sensitive itemset are modified to temporary binary u variables. Using the bitmap notation, this modification is given in the intermediate dataset $\sim D$ and can be defined as shown in Equation (11).

$$\begin{gathered}\forall X \in S \\ \sim d_{ij} = \begin{cases} u_{ij}, & if\ X \subseteq Ti\ and\ I_j \in X \\ d_{ij}, & otherwise \end{cases}\end{gathered} \tag{11}$$

Since the items that may be modified in the sanitized dataset are u variables, the optimal itemset hiding problem can be formulated as follows.

$$\begin{gathered} maximize\left(\sum_{u_{ij} \in U} u_{ij}\right) \\ subject\ to \begin{cases} \forall X \in S,\ \sigma^s(X) < \sigma_{min} \\ \forall Y \in F^n,\ \sigma^s(Y) \geq \sigma_{min} \end{cases}\end{gathered} \tag{12}$$

4. Itemset Hiding Using Sibling Itemsets Constraints

Frequent itemset mining and CSP formulation preliminaries are given in the previous section. Considering that all non-sensitive frequent itemsets will increase the number of constraints, to lessen constraints, we introduce the sibling itemset concept. Sibling itemsets SI of a frequent k-itemset X are generating itemsets of k + 1 candidate itemset. The idea behind this concept is that hiding a k-itemset will also hide its k + 1 supersets but remain non-sensitive subsets of these k + 1 supersets discoverable. This represents a local border.

$$\begin{gathered}\forall X \in S, \\ SI(X) = \{Y \in F^n : |Y - X| = 1\ and\ Y \equiv X\}\end{gathered} \tag{13}$$

Using sibling itemsets instead of all non-sensitive frequent itemsets, CSP defined in (12) can be defined as follows

$$
\begin{gathered}
maximize\left(\sum_{u_{ij}\in U} u_{ij}\right) \\
subject\ to\begin{cases}\forall X \in S,\ \sigma^s(X) < \sigma_{min} \\ \forall Y \in SI,\ \sigma^s(Y) \geq \sigma_{min}\end{cases}
\end{gathered}
\tag{14}
$$

Generation and determining support of sibling itemsets of a sensitive itemset is conducted in the hiding process. In this way, the time consumption of prior itemset mining is eliminated.

There are two types of constraints for our CSP: sensitive itemset constraints and sibling itemset constraints. The first type ensures that sensitive itemsets are below the defined minimum support threshold. Thus, all of these constraints must be satisfied. The second type of constraint is satisfied to lessen information loss. There are situations when all of these cannot be satisfied, and the CSP is not solvable. Then, we need to sacrifice some of them. In our approach, information loss is preferred to a privacy breach. Therefore, some constraints for sibling itemsets can be sacrificed. Instead of removing any of those constraints, we add binary relaxation variables. By adding these, we do not need to reformulate the CSP and run solver more than once. We add a unique binary relaxation variable r to the inequality for all sibling constraints.

$$
\begin{gathered}
maximize\left(\sum_{u_{ij}\in U} u_{ij}\right) \\
subject\ to\begin{cases}\forall X \in S,\ \sigma^s(X) < \sigma_{min} \\ \forall Y \in SI,\ \sigma^s(Y) + r_Y \geq \sigma_{min}\end{cases}
\end{gathered}
\tag{15}
$$

Relaxation on constraints should be minimized to ensure that information loss is minimized.

$$
minimize\left(\sum_{Y\in SI} r_Y\right) \tag{16}
$$

So, Equation (15) gives our final CSP formulation.

$$
\begin{gathered}
maximize\left(\sum_{u_{ij}\in U} u_{ij} - \sum_{Y\in SI} r_Y\right) \\
subject\ to\begin{cases}\forall X \in S,\ \sigma^s(X) < \sigma_{min} \\ \forall Y \in SI,\ \sigma^s(Y) + r_Y \geq \sigma_{min}\end{cases}
\end{gathered}
\tag{17}
$$

Illustrative Example

In the following, an illustrative example of our hiding approach is given. Let D be the dataset of 10 transactions shown in Table 2. Our set of items is $I = \{A, B, C, D, E\}$.

Table 2. Dataset D.

TID	Items
T_1	AC
T_2	ACDE
T_3	CD
T_4	BE
T_5	ACDE
T_6	DE
T_7	C
T_8	AB
T_9	AC
T_{10}	CD

Using the bitmap notation given in (1), we have a 10×5 binary matrix representation of D. It is given in Table 3.

Table 3. Dataset D in bitmap notation.

A	**B**	**C**	**D**	**E**
1	0	1	0	0
1	0	1	1	1
0	0	1	1	0
0	1	0	0	1
1	0	1	1	1
0	0	0	1	1
0	0	1	0	0
1	1	0	0	0
1	0	1	0	0
0	0	1	1	0

Using the formulation in (2), we can calculate the support count of an itemset, for instance, the support count of itemset $\{AC\}$.

$$\sigma_{\{AC\}} = d_{1,1}d_{1,3} + d_{2,1}d_{2,3} + \ldots + d_{10,1}d_{10,3}$$

$$\sigma_{\{AC\}} = 4$$

Given minimum support count $\sigma_{min} = 2$ and Equation (3), we can find 16 frequent itemsets.

$$F = \{A,\ B,\ C,\ D,\ E,\ AC,\ AD,\ AE,\ CD,\ CE,\ DE,\ ACD,\ ACE,\ ADE,\ CDE,\ ACDE\}$$

Suppose that itemset $\{CD\}$ is given as sensitive and needs to be hidden. Then, $S = \{CD\}$

Equations (4) and (5) give supersets of sensitive itemsets and non-sensitive itemsets as follows:

$$Ss = \{ACD,\ CDE,\ ACDE\}$$

$$F^n = \{A,\ B,\ C,\ D,\ E,\ AC,\ AD,\ AE,\ CE,\ DE,\ ACE,\ ADE\}$$

All itemsets in S and Ss must be hidden to achieve privacy, whereas as many itemsets as in F^n should remain frequent after sanitization.

Using the formulation given in (11), transactions supporting itemset $\{CD\}$ are modified with binary variables. Their values will be determined after the CSP is solved. The intermediate form of the dataset is given in Table 4.

Table 4. Intermediate form of dataset D.

A	**B**	**C**	**D**	**E**
1	0	1	0	0
1	0	$\mathbf{u_{2,3}}$	$\mathbf{u_{2,4}}$	1
0	0	$\mathbf{u_{3,3}}$	$\mathbf{u_{3,4}}$	0
0	1	0	0	1
1	0	$\mathbf{u_{5,3}}$	$\mathbf{u_{5,4}}$	1
0	0	0	1	1
0	0	1	0	0
1	1	0	0	0
1	0	1	0	0
0	0	$\mathbf{u_{10,3}}$	$\mathbf{u_{10,4}}$	0

Using (13), we can find sibling itemsets as $SI = \{AC,\ CE,\ AD,\ DE\}$. Now, we can define the CSP formulation given in (15).

$$maximize\ u_{2,3} + u_{2,4} + u_{3,3} + u_{3,4} + u_{5,3} + u_{5,4} + u_{10,3} + u_{10,4} - r_{\{AC\}} - r_{\{CE\}} - r_{\{AD\}} - r_{\{DE\}}$$

$$subject\ to \begin{cases} u_{2,3}u_{2,4} + u_{3,3}u_{3,4} + u_{5,3}u_{5,4} + u_{10,3}u_{10,4} < \sigma_{min} \\ 1 + u_{2,3} + u_{5,3} + r_{\{AC\}} \geq \sigma_{min} \\ u_{2,3} + u_{5,3} + r_{\{CE\}} \geq \sigma_{min} \\ u_{2,4} + u_{5,4} + r_{\{AD\}} \geq \sigma_{min} \\ u_{2,4} + u_{5,4} + r_{\{DE\}} \geq \sigma_{min} \end{cases}$$

$$where\ \sigma_{min} = 2$$

The solution of such CSP is

$$u_{2,3} = u_{2,4} = u_{3,4} = u_{5,3} = u_{10,4} = r_{\{AD\}} = 1$$

$$u_{3,3} = u_{5,4} = u_{10,3} = r_{\{AC\}} = r_{\{CE\}} = r_{\{DE\}} = 0$$

When results are applied to the intermediate form of the dataset, we obtain the sanitized dataset D^s as given in Table 5. From this sanitized dataset, we can find itemsets for minimum support count $\sigma_{min} = 2$ as $F^s = \{A, B, C, D, E, AC, AE, CE, DE, ACE\}$.

Table 5. Sanitized Dataset.

A	**B**	**C**	**D**	**E**
1	0	1	0	0
1	0	1	1	1
0	0	**0**	1	0
0	1	0	0	1
1	0	1	**0**	1
0	0	0	1	1
0	0	1	0	0
1	1	0	0	0
1	0	1	0	0
0	0	**0**	1	0

The sensitive itemset $\{CD\}$ is no longer frequent for support count 2. Compared to the initial dataset, the number of itemsets decreased to 10 from 16. Two itemsets are accidentally lost, and 3 itemsets are supersets of $\{CD\}$; therefore, they are also missing.

5. Experimental Analysis

In this section, we give a performance evaluation of our approach. The reference algorithm for comparison is IPA [20]. We implemented the algorithms using Python. Constraints were solved using Minizinc [33]. Implementations used the Pymzn [34] library to be able to invoke, run, and gather results from the constraint solver. All computational experiments were conducted on a PC running MS Windows 10 with an Intel i5-4200U CPU and 8 GB of RAM.

5.1. Evaluation Metrics of Itemset Hiding

Itemset hiding aimed for transforming the dataset in a way that sensitive itemsets were concealed. Itemset hiding aimed for transforming the dataset in a way that sensitive itemsets were concealed, non-sensitive frequent itemsets were preserved, ghost itemsets were not generated and dataset distortion is minimum. These goals are mentioned in Section 3, and related metrics are given below.

5.1.1. Hiding Failure

This metric concerns sensitive itemsets remaining frequent after the sanitization process. It is defined as the percentage of sensitive itemsets that appear in the sanitized dataset divided by the ones that appeared in the original dataset.

Our approach ensures that all sensitive itemsets are hidden; therefore, HF = 0 for all scenarios. As far as we have surveyed, all proposed approaches focus on HF and ensure that they are 0. Our reference algorithm IPA also ensures that all sensitive itemsets are hidden.

5.1.2. Artifactual Patterns

This metric concerns side effects of sanitization process because some approaches insert items or transactions to the dataset during or after sanitization. It was calculated as the ratio of itemsets that did not appear in the original dataset but appeared in the sanitized dataset to the itemsets that appear in both the original and the sanitized datasets.

Since our approach does not insert items on the original dataset, it is not possible to produce new itemsets from the sanitized dataset. This is the same for the IPA algorithm.

5.1.3. Dissimilarity

This metric is the measure of the differences between the original and the sanitized dataset quantified by comparing total number of items or the frequencies.

For our approach, the number of deleted items gives dissimilarity between the original and sanitized dataset. The number of deleted items are identical with IPA algorithm.

5.1.4. Misses Cost

This metric concerns the side effects of the sanitization process. It is measured as the percentage of non-sensitive patterns that disappeared in the sanitized dataset divided by the ones that appeared in the original dataset.

We have given this measure as the number of lost itemsets. This is the only metric that differs with the IPA algorithm and further comparison is given in next title.

5.2. Comparison

We evaluated the algorithms on six different datasets obtained from [35]. Characteristics of these datasets are given in Table 6. Since the IPA algorithm uses frequent itemsets discovered before the hiding process, we also provided time consumption for tested values on datasets. Python implementation [36] of the Eclat algorithm was used for frequent itemset mining.

Table 6. Properties of datasets.

Dataset Name	Number of Transactions (Count)	Average Transaction Length	Number of Items	Minimum Support Count	Number of Frequent Itemsets	Runtime (Seconds)
T10I4D100K	100,000	10.10	870	500 (%0.5)	1073	9.15
T40I10D100K	100,000	39.60	942	500 (%0.5)	1,286,037	392.96
Mushroom	8124	23.00	119	406 (%5)	3,755,704	9.77
retail	88,162	10.30	16,470	440 (%0.5)	581	2.00
BMS1	59,602	2.51	497	60 (%0.1)	3991	0.88
BMS2	77,512	4.62	3,340	77 (%0.1)	24,143	5.22

We tested the algorithms using different hiding scenarios: hiding 1 2-itemset (HS_2.1), hiding 2 2-itemset(HS_2.2), hiding 3 2-itemset(HS_2.3), hiding 1 3-itemset(HS_3.1), hiding 2 3-itemset(HS_3.2), and hiding 1 4-itemset(HS_4.1). The sensitive itemsets chosen had support counts close to the minimum support count since those itemsets were more logically hidden and indistinguishable compared to the rest. For ease of use in tables, our approach is named HISB (Hiding Itemsets with SiBlings) during this section.

The results of evaluation for the T10I4100K dataset are given in Table 7. Columns represent hiding scenarios, side effects as the number of lost itemsets, and running time in seconds for algorithm IPA and HISB. Both algorithms performed well in terms of number of lost itemsets. In defined scenarios, no itemset was lost. Our approach performed better in terms of runtime in four scenarios. It should be noted that IPA needs prior itemset mining, which costs an additional 9.15 s.

Table 7. Results of the T10I4100K dataset.

Hiding Scenario	Number of Lost Itemsets (IPA/HISB)	Algorithm IPA (Seconds)	Algorithm HISB (Seconds)
HS_2.1	0/0	5.72	4.04
HS_2.2	0/0	6.75	5.2
HS_2.3	0/0	7.62	6.03
HS_3.1	0/0	6.78	5.54
HS_3.2	0/0	9.51	10.31
HS_4.1	0/0	9.01	10.5

The results of evaluation for the T40I10100K dataset are given in Table 8. IPA performed better in terms of number of lost itemsets. On the other hand, our approach performed better in terms of runtime even though prior itemset mining consumption was not included for IPA.

Table 8. Results of the T40I10100K dataset.

Hiding Scenario	Number of Lost Itemsets (IPA/HISB)	Algorithm IPA (Seconds)	Algorithm HISB (Seconds)
HS_2.1	0/1	13.64	13.31
HS_2.2	0/1	27.59	14.59
HS_2.3	0/2	62.49	22.88
HS_3.1	0/0	95.61	18.92
HS_3.2	0/0	1,110.97	31.32
HS_4.1	0/1	408.48	24.02

The results of evaluation for the Mushroom dataset are given in Table 9. Both algorithms performed well regarding the number of lost itemset where no itemset was lost. On the other hand, our approach performed better in terms of runtime even though prior itemset mining consumption was not included for IPA.

Table 9. Results of the Mushroom dataset.

Hiding Scenario	Number of Lost Itemsets (IPA/HISB)	Algorithm IPA (Seconds)	Algorithm HISB (Seconds)
HS_2.1	0/0	8.98	1.56
HS_2.2	0/0	18.45	2.6
HS_2.3	0/0	22.45	4.23
HS_3.1	0/0	23	2.67
HS_3.2	0/0	25.78	4.96
HS_4.1	0/0	17.44	6.11

The results of evaluation for the BMS1 dataset are given in Table 10. IPA performed better in terms of number of lost itemsets. The runtime performance of hiding processes was close in five of six scenarios.

Table 10. Results of the BMS1 dataset.

Hiding Scenario	Number of Lost Itemsets (IPA/HISB)	Algorithm IPA (Seconds)	Algorithm HISB (Seconds)
HS_2.1	0/0	1.07	1.06
HS_2.2	0/0	1.14	1.17
HS_2.3	0/1	1.18	1.18
HS_3.1	0/0	1.2	1.36
HS_3.2	0/1	1.47	1.39
HS_4.1	0/2	2.96	1.57

The results of evaluation for the BMS2 dataset are given in Table 11. Both algorithms performed well in terms of number of lost itemsets. The runtime performance of hiding processes was similar in four scenarios.

Table 11. Results of the BMS2 dataset.

Hiding Scenario	Number of Lost Itemsets (IPA/HISB)	Algorithm IPA (Seconds)	Algorithm HISB (Seconds)
HS_2.1	0/0	1.07	1.06
HS_2.2	0/0	1.14	1.17
HS_2.3	0/1	1.18	1.18
HS_3.1	0/0	1.2	1.36
HS_3.2	0/1	1.47	1.39
HS_4.1	0/2	2.96	1.57

The results of the evaluation for the Retail dataset are given in Table 12. Both algorithms performed well in terms of number of lost itemsets. The runtime performance of hiding processes was close.

Table 12. Results of Retail dataset.

Hiding Scenario	Number of Lost Itemsets (IPA/HISB)	Algorithm IPA (Seconds)	Algorithm HISB (Seconds)
HS_2.1	0/0	36.21	33.21
HS_2.2	0/0	38.59	38.54
HS_2.3	0/0	43.36	52.48
HS_3.1	0/0	39.15	34.89
HS_3.2	0/0	43.92	42.98
HS_4.1	0/0	45.11	44.52

5.3. Discussion

First of all, we can say that using sibling itemsets constraints to lessen the runtime of the hiding process is effective. Even though the comparison tables do not include itemset mining time consumption before the hiding process, our approach performed faster in most cases. To add this, in some cases, the number of border itemsets or the length of some border itemsets constructed by the IPA algorithm caused distinctive runtime differences in the hiding process. Experiments on the Mushroom dataset revealed that eliminating prior mining was advantageous when the dataset is dense. Although this dataset has fewer items and transactions, the number of frequent itemsets for the given support threshold was over 3.5 million. We also observed that unsolvable constraints caused another disadvantage. However, this was not common in most cases. Secondly, the number of lost itemsets caused by our approach is tolerable when considering the number of frequent itemsets.

At this juncture, we would like to mention that we have also implemented the algorithm given in [22]. It promises optimum results since it is a full exact approach and uses relaxation techniques for CSP. However, we could not finish experiments because of insufficient time or hardware limitations that caused crashes. The reason for this problem is that the proposed approach generates constraints for all non-sensitive frequent itemsets. Considering our experiments, it should generate constraints for over 1 million and 3 million frequent itemsets for T40I10D100K and Mushroom datasets, respectively, which is not feasible.

6. Conclusions

This paper presents a methodology for hiding sensitive itemsets in transactional datasets. We focused on reducing the number of constraints for exact itemset hiding. Using sibling itemset notion and defining relaxation variables for constraints, we benefited from the exact nature of algorithms to obtain an ideal solution or minimally affected dataset. We showed that sibling itemsets are an efficient solution for reducing constraints for exact approaches. Given a comparison with a reference algorithm, we also discuss the need for prior computation of frequent itemsets. Experiments revealed that eliminating prior mining of frequent itemsets on the dataset combined with a sibling itemset approach is time-efficient where side effects such as lost itemsets are tolerable. We also mention some symmetric and asymmetric properties appearing in itemset hiding.

To sum up, our approach is especially applicable where prior mining of frequent itemsets is costly. This is also valid for frequently updated databases. Therefore, we can say that skipping prior mining while using constraints is one of the most important contributions of our approach. Additionally, using fewer constraints makes our approach even better in terms of runtime. Moreover, we added relaxation variables to make our approach more efficient when initial constraints cause a CSP that is not feasible. Although this is not common, it may result in additional runtime since the constraint solver needs to be run more than once. It can be concluded that our methodology serves an exact approach with fewer constraints so that the hiding process consumes less time.

Author Contributions: Conceptualization, B.Y. and A.K.; methodology, B.Y. and A.K.; software, B.Y.; validation, B.Y., A.K. and R.Y.; investigation, B.Y.; writing—original draft preparation, B.Y.; writing—review and editing, B.Y., A.K. and R.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Datasets used in this study are obtained from Frequent Itemset Mining Dataset Repository [35].

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Feng, K.; Tian, J. Forecasting Reference Evapotranspiration Using Data Mining and Limited Climatic Data. *Eur. J. Remote Sens.* **2021**, *54* (Suppl. 2), 363–371. [CrossRef]
2. Raja, K.; Patrick, M.; Gao, Y.; Madu, D.; Yang, Y.; Tsoi, L.C. A Review of Recent Advancement in Integrating Omics Data with Literature Mining towards Biomedical Discoveries. *Int. J. Genom.* **2017**, *2017*, 6213474. [CrossRef] [PubMed]
3. Neto, C.; Brito, M.; Lopes, V.; Peixoto, H.; Abelha, A.; Machado, J. Application of Data Mining for the Prediction of Mortality and Occurrence of Complications for Gastric Cancer Patients. *Entropy* **2019**, *21*, 1163. [CrossRef]
4. Hong, J.; Park, S. The Identification of Marketing Performance Using Text Mining of Airline Review Data. *Mob. Inf. Syst.* **2019**, *2019*, 1790429. [CrossRef]
5. Amanowicz, M.; Jankowski, D. Detection and Classification of Malicious Flows in Software-Defined Networks Using Data Mining Techniques. *Sensors* **2021**, *21*, 2972. [CrossRef] [PubMed]
6. Sánchez-Aguayo, M.; Urquiza-Aguiar, L.; Estrada-Jiménez, J. Predictive Fraud Analysis Applying the Fraud Triangle Theory through Data Mining Techniques. *Appl. Sci.* **2022**, *12*, 3382. [CrossRef]
7. Clifton, C. Privacy-Preserving Data Mining. In *Encyclopedia of Database Systems*; Liu, L., Özsu, M.T., Eds.; Springer: New York, NY, USA, 2018; pp. 2819–2821. [CrossRef]
8. Zhang, L.; Wang, W.; Zhang, Y. Privacy Preserving Association Rule Mining: Taxonomy, Techniques, and Metrics. *IEEE Access* **2019**, *7*, 45032–45047. [CrossRef]
9. Mendes, R.; Vilela, J.P. Privacy-Preserving Data Mining: Methods, Metrics, and Applications. *IEEE Access* **2017**, *5*, 10562–10582. [CrossRef]
10. Verykios, V.S.; Elmagarmid, A.K.; Bertino, E.; Saygin, Y.; Dasseni, E. Association Rule Hiding. *IEEE Trans. Knowl. Data Eng.* **2004**, *16*, 434–447. [CrossRef]
11. *Association Rule Hiding for Data Mining*; Advances in Database Systems; Springer: Boston, MA, USA, 2010; Volume 41.
12. Atallah, M.; Bertino, E.; Elmagarmid, A.; Ibrahim, M.; Verykios, V. Disclosure Limitation of Sensitive Rules. In Proceedings of the 1999 Workshop on Knowledge and Data Engineering Exchange (KDEX'99), Chicago, IL, USA, 7 November 1999; pp. 45–52. [CrossRef]
13. Saygin, Y.; Verykios, V.S.; Clifton, C. Using Unknowns to Prevent Discovery of Association Rules. *ACM SIGMOD Rec.* **2001**, *30*, 45. [CrossRef]
14. Lee, G.; Chang, C.-Y.; Chen, A.L.P. Hiding Sensitive Patterns in Association Rules Mining. In Proceedings of the 28th Annual International Computer Software and Applications Conference, 2004. COMPSAC 2004, Hongkong, China, 28–30 September 2004; pp. 424–429. [CrossRef]
15. Mannila, H.; Toivonen, H. Levelwise Search and Borders of Theories in KnowledgeDiscovery. *Data Min. Knowl. Discov.* **1997**, *1*, 241–258. [CrossRef]
16. Moustakides, G.V.; Verykios, V.S. A MaxMin Approach for Hiding Frequent Itemsets. *Data Knowl. Eng.* **2008**, *65*, 75–89. [CrossRef]
17. Sun, X.; Yu, P.S. Hiding Sensitive Frequent Itemsets by a Border-Based Approach. *J. Comput. Sci. Eng.* **2007**, *1*, 74–94. [CrossRef]

18. Quoc Le, H.; Arch-Int, S.; Arch-Int, N. Association Rule Hiding Based on Distance and Intersection Lattice. In *International Conference on Software Technology and Engineering (ICSTE 2012)*; ASME Press: New York, NY, USA, 2015. [CrossRef]
19. Menon, S.; Sarkar, S.; Mukherjee, S. Maximizing Accuracy of Shared Databases When Concealing Sensitive Patterns. *Inf. Syst. Res.* **2005**, *16*, 256–270. [CrossRef]
20. Gkoulalas-Divanis, A.; Verykios, V.S. An Integer Programming Approach for Frequent Itemset Hiding. In Proceedings of the 2006 ACM CIKM International Conference on Information and Knowledge Management, Arlington, VA, USA, 6–11 November 2006; pp. 748–757. [CrossRef]
21. Gkoulalas-Divanis, A.; Verykios, V.S. Hiding Sensitive Knowledge without Side Effects. *Knowl. Inf. Syst.* **2009**, *20*, 263–299. [CrossRef]
22. Ayav, T.; Ergenc, B. Full-Exact Approach for Frequent Itemset Hiding. *Int. J. Data Warehous. Min.* **2015**, *11*, 49–63. [CrossRef]
23. Lin, C.W.; Zhang, B.; Yang, K.T.; Hong, T.P. Efficiently Hiding Sensitive Itemsets with Transaction Deletion Based on Genetic Algorithms. *Sci. World J.* **2014**, *2014*, 398269. [CrossRef]
24. Lin, J.C.-W.; Liu, Q.; Fournier-Viger, P.; Hong, T.-P.; Voznak, M.; Zhan, J. A Sanitization Approach for Hiding Sensitive Itemsets Based on Particle Swarm Optimization. *Eng. Appl. Artif. Intell.* **2016**, *53*, 1–18. [CrossRef]
25. Bux, N.K.; Lu, M.; Wang, J.; Hussain, S.; Aljeroudi, Y. Efficient association rules hiding using genetic algorithms. *Symmetry* **2018**, *10*, 576. [CrossRef]
26. Agrawal, R.; Imieliński, T.; Swami, A. Mining Association Rules between Sets of Items in Large Databases. *ACM SIGMOD Rec.* **1993**, *22*, 207–216. [CrossRef]
27. Agrawal, R.; Srikant, R. Fast Algorithms for Mining Association Rules. In Proceedings of the 20th International Conference on Very Large Data Bases (VLDB '94), Santiago, Chile, 12–15 September 1994; pp. 487–499.
28. Zaki, M.J.; Parthasarathy, S.; Ogihara, M.; Li, W. New Algorithms for Fast Discovery of Association Rules. In *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining, Newport Beach, CA, USA, 14–17 August 1997*; AAAI Press: Palo Alto, CA, USA, 1997; pp. 283–286.
29. Han, J.; Pei, J.; Yin, Y. Mining Frequent Patterns without Candidate Generation. *SIGMOD Rec.* **2000**, *29*, 1–12. [CrossRef]
30. Bustio-Martínez, L.; Cumplido, R.; Hernández-León, R.; Bande-Serrano, J.M.; Feregrino-Uribe, C. On the Design of Hardware-Software Architectures for Frequent Itemsets Mining on Data Streams. *J. Intell. Inf. Syst.* **2018**, *50*, 415–440. [CrossRef]
31. Mahmood, S.; Shahbaz, M.; Guergachi, A. Negative and Positive Association Rules Mining from Text Using Frequent and Infrequent Itemsets. *Sci. World J.* **2014**, *2014*, 973750. [CrossRef] [PubMed]
32. Naulaerts, S.; Meysman, P.; Bittremieux, W.; Vu, T.N.; Berghe, W.V.; Goethals, B.; Laukens, K. A Primer to Frequent Itemset Mining for Bioinformatics. *Brief. Bioinform.* **2015**, *16*, 216–231. [CrossRef]
33. MiniZinc. Available online: https://www.minizinc.org/ (accessed on 16 June 2022).
34. PyMzn—PyMzn Documentation. Available online: http://paolodragone.com/pymzn/ (accessed on 16 June 2022).
35. FIMI. Frequent Itemset Mining Dataset Repository. Available online: http://fimi.uantwerpen.be/data/ (accessed on 16 June 2022).
36. Borgelts, C. Christian Borgelt's Web Pages. Available online: http://www.borgelt.net/fpm.html (accessed on 16 June 2022).