BULGARIAN ACADEMY OF SCIENCES

CYBERNETICS AND INFORMATION TECHNOLOGIES • Volume **22**, No 1
Sofia • 2022 Print ISSN: 1311-9702; Online ISSN: 1314-4081
DOI: 10.2478/cait-2022-0001

Hiding Sensitive High Utility and Frequent Itemsets Based on Constrained Intersection Lattice

Huynh Trieu Vy[1], *Le Quoc Hai*[2], *Nguyen Thanh Long*[2], *Truong Ngoc Chau*[3], *Le Quoc Hieu*[4]

[1]*Information Technology Faculty, Pham Van Dong University, Vietnam*
[2]*Information Technology Faculty, Quang Tri Teacher Training College, Vietnam*
[3]*Information Technology Faculty, Da Nang University, Vietnam*
[4]*Faculty of Information Systems, University of Economics And Law, Vietnam*
E-mails: htvy@pdu.edu.vn hailq79@gmail.com long_nt@qtttc.edu.vn
truongngocchau@yahoo.com hieulq@uel.edu.vn

***Abstract*:** *Hiding high utility and frequent itemset is the method used to preserve sensitive knowledge from being revealed by pattern mining process. Its goal is to remove sensitive high utility and frequent itemsets from a database before sharing it for data mining purposes while minimizing the side effects. The current methods succeed in the hiding goal but they cause high side effects. This paper proposes a novel algorithm, named HSUFIBL, that applies a heuristic for finding victim item based on the constrained intersection lattice theory. This algorithm specifies exactly the condition that allows the application of utility reduction or support reduction method, the victim item, and the victim transaction for the hiding process so that the process needs the fewest data modifications and gives the lowest number of lost non-sensitive itemsets. The experimental results indicate that the HSUFIBL algorithm achieves better performance than previous works in minimizing the side effect.*

***Keywords*:** *High utility mining, High utility and frequent itemset, Sensitive high utility and frequent itemset hiding, Privacy-preserving utility mining.*

1. Introduction

Today the development of international business and e-commerce requires companies to share their data with partners to promote their collaboration. However, sharing data between companies could potentially create more risks; therefore, the sensitive information, which is discovered by data mining algorithms can be explored by their competitors. This harms their business and collaboration. In order to preserve sensitive information while maintaining other valuable information in database, a number of algorithms have been proposed for hiding sensitive information from database before sharing it with the partners [2, 3, 9, 13].

Privacy-Preserving high Utility Itemset Mining (PPUIM) is a technique that preserves sensitive information which can be extracted by High Utility Itemset Mining (HUIM) algorithm from database [5]. Its goal is to remove all sensitive high utility itemsets from a database by reducing the utility of the data item. This process leads to possible loss of non-sensitive itemsets, difference of data mining results, and data values of the sanitized database in comparison to the original database. They are defined as side effects of the PPUIM algorithms. The target of the PPUIM algorithm is to sanitize a database in order to hide all sensitive itemsets with the lowest side effects. To achieve this, many heuristic methods have been proposed to optimize the hiding process by finding exactly victim items and victim transactions for the data modification. The first PPUIM algorithms named HHUIF (Hiding High Utility Item First) algorithm and MSICF (Maximum Sensitive Itemsets Conflict First) algorithm have been proposed by Y e h and H s u [16] and their improvements have been then developed [7, 10, 11].

Privacy-Preserving high Utility and Frequent Itemset Mining (PPUFIM) is an extension of the PPUIM. Although the PPUIM algorithms can be applied to the PPUFIM problem, they result in low performance. To solve this issue, R a j a l a x m i and N a t a r a j a n [14] proposed two algorithms named MSMU (Minimum Support and Maximum Utility) and MCRSU (Maximum Conflict Ratio for Support and Utility). They succeeded in hiding all sensitive itemsets but they failed in minimizing the side effects. To overcome this, L i u, X u and L v [12] proposed a maximal border to specify whether the algorithm should reduce support or utility of a sensitive itemset in order to achieve higher performance. This method gained better results compared to previous works. However, the side effects are still high because the proposed algorithm HUFI (Hiding Utility and Frequent Itemsets) uses the same method for selecting victim items and victim transactions in both cases of support reduction and utility reduction.

This paper aims to propose a novel algorithm for the PPUFIM problem to overcome the drawback of the HUFI algorithm [12] and the current PPUIM algorithms when dealing with the PPUFIM problem. In order to minimize the side effect, this paper proposes a heuristic to specify which method, including support reduction and utility reduction method, contributes to reducing the side effects. Moreover, this paper defines a constrained intersection lattice of high utility and frequent itemset. Then, a heuristic strategy is proposed based on this lattice to select exactly the victim item for the data modification process. This strategy therefore achieves better performance than the previous works.

The remaining of this paper consists of four main sections. In Section 2, we introduce some works related to PPUIM. In Section 3, we provide definitions and statements related to the high utility and frequent itemset mining (HUFIM) problem. Then in Section 4, we propose a new algorithm named HSUFIBL. In this section, by experimental results, we will prove the efficiency of our algorithm and show that it causes lower side effects in comparison to the previous algorithms proposed in [7, 12, 14]. And the last section is the conclusion.

2. Related works

The general process of data sanitization approach for PPUIM includes two steps: Firstly, an item is specified as a victim item and a transaction is specified as a victim transaction in such a way that modifying value of the victim item in the victim transaction causes the lowest side effects. Secondly, the victim item in the victim transaction is then reduced in order to decrease the utility of the sensitive itemset below a given a threshold. The side effects of a high utility itemset hiding algorithm depend on the strategy for specifying victim item and victim transaction. The first algorithms named HHUIF and MSICF have been proposed by Y e h and H s u [16] in 2010. In these algorithms, the victim item is specified as an item that has maximal utility at the victim transaction. With a simple strategy for the hiding process, HHUIF and MSICF result in high side effects. To overcome this drawback, more efficient algorithms such as MSU-MAU and MSU-MIU [11], EHSHUI [7] have been then proposed.

In reality, both frequency and utility of itemsets are considered for business or decision making. High utility and frequent itemset hiding algorithm therefore is developed for the sensitive knowledge protection purpose. In 2012, R a j a l a x m i and N a t a r a j a n [14] have proposed two novel algorithms named MSMU and MCRSU. The methodology [14] is to modify the value of data item to reduce both support and utility of the sensitive itemset to less than the minimum support threshold and minimum utility threshold, respectively. Both algorithms firstly reduce support of the sensitive itemset. In the case of the sensitive itemset being still not hidden, they then reduce its utility below the minimum utility threshold. Although the MSMU and MCRSU strategies succeeded in hiding all of the sensitive itemsets, they failed to minimize the side effects. Accordingly, when hiding a sensitive high utility and frequent itemset, MSMU and MCRSU cause the loss of many non-sensitive high utility and frequent itemsets. This weakness has been then overcome by the HUFI algorithm which have been proposed by L i u, X u and L v [12]. In order to hide the sensitive itemset, the HUFI algorithm modifies the value of a sensitive item until either the support of the sensitive itemset is less than the minimum support threshold or the utility of the sensitive itemset is less than the minimum utility threshold. To minimize the side effect, L i u, X u and L v [12] has proposed a maximal border to specify whether the algorithm should reduce support or utility to achieve better performance. This method gains better results in comparison to the previous works. However, the HUFI algorithm uses the same method to select victim item and victim transaction for both support and utility reduction. This still causes too many side effects. In order to solve this problem, in this paper, we propose a heuristic to specify the victim item and victim transaction for the case of utility reduction. For the support reduction case, we propose a constrained intersection lattice of high utility and frequent itemsets, and then based on this lattice we propose a heuristic for victim item specification.

3. Basic preliminaries and problem statement

In this section, we recall the basic concepts presented in previous works [7, 11, 12, 15, 16]. We apply these theories to propose the novel algorithm.

Let $I = \{i_1, i_2, \dots, i_m\}$ be a finite set of items, where each item $i_l \in I$ has an external utility $p(i_l)$. A transaction database *D* is a set of transactions $\{T_1, T_2, \dots, T_n\}$, where each transaction $T_c \subseteq I,\ 1 \leq c \leq n$, has a unique identifier, called Tid. Each item i in a transaction T_c is associated with a weight indicator called quantity $q(i, T_c)$, which is the number of items i appearing in the transaction T_c. An itemset $X \subseteq I$ containing *k* items is called a *k*-itemset. A transaction T_c is said to support an itemset X if $X \subseteq T$.

Table 1. Transaction database *D*

Tid	Transaction	Tid	Transaction
T1	A(3), B(1), C(5), F(2)	T6	B(2), C(2), F(1), H(2)
T2	D(2), E(3), F(3), G(1)	T7	D(1), E(1), F(3), G(2), H(2)
T3	A(2), B(3), D(3), E(5), F(1)	T8	B(1), D(2), H(2)
T4	A(3), B(2), C(1), E(2)	T9	B(4), D(3), F(1)
T5	D(2), E(3), F(5)	T10	B(4), D(1), F(3)

Table 2. External utility of transaction dataset *D*

Item	A	B	C	D	E	F	G	H
Utility	4	2	2	1	2	1	2	1

Definition 1. The support of an itemset X in the database D, denoted as $\text{support}(X)$, is defined as

$$\text{support}(X) = \text{supc}(X)/|D|,$$

where, $\text{supc}(X)$ is support count of an itemset X, is defined as

$$\text{supc}(X) = |\{T_c | X \subseteq T_c, T_c \in D\}|.$$

Definition 2. An itemset X in the database D is said to be a frequent itemset if its support is greater than or equal to a predetermined minimum support threshold δ.

Definition 3 (The utility). Let i be an item, X be an itemset, and i, $X \subseteq T_c$, we have:

The utility of an item i in a transaction T_c is defined by $u(i, T_c) = q(i, T_c) * p(i)$.

The utility of an itemset X in a transaction T_c is defined by

$$u(X, T_c) = \sum_{i \in X} u(i, T_c).$$

The utility of an itemset X in a transaction database D is defined by

$$u(X) = \sum_{X \subseteq T_c \wedge T_c \in D} u(X, T_c).$$

The utility of a transaction T_c in a transaction database D is defined by

$$tu(T_c) = \sum_{i \in T_c} u(i, T_c).$$

Definition 4. An itemset X is said to be a high utility itemset if the utility of X is not less than minimum utility threshold ε, that is $u(X) \geq \varepsilon$.

Definition 5. The itemset X is said to be a high utility and frequent itemset if it is a frequent and high utility itemset. Let HUFIs be a set of High Utility and Frequent Itemsets then we have $\text{HUFIs} = \{X | X \subseteq I, u(X) \geq \varepsilon \wedge \text{support}(X) \geq \delta\}$.

For example Table 3 is a set of high utility and frequent itemsets mined from the data set given in Table 1 and Table 2 where the minimum utility threshold is $\varepsilon = 27$ and the support threshold is $\delta = 20\%$.

Table 3. The set HUFIs mined from *D* with $\varepsilon = 27$ and $\delta = 20\%$

Itemset	Utility	Support(%)	Itemset	Utility	Support(%)	Itemset	Utility	Support(%)
A	32	30	ABF	31	20	DF	30	60
AB	44	30	B	34	70	DEF	44	40
AC	36	20	BD	33	40	E	28	50
AE	42	20	BF	36	50	EF	36	40
ABC	42	20	BDF	34	30			
ABE	44	20	DE	32	40			

Definition 6 (Sensitive high utility and frequent itemset). In the set of high utility and frequent itemsets, there are some itemsets containing useful information that is needed for making business decision. They are called Sensitive High Utility and Frequent Itemsets, in short is SHUFIs.

Accordingly, the set of non-sensitive high utility and frequent itemsets, denoted by nonSHUFIs, is defined as: nonSHUFIs = HUFIs − SHUFIs.

The set of SHUFIs needs to be preserved from being revealed when sharing the database containing them.

Definition 7. Hiding SHUFIs is the process of transforming the original transaction database *D* to a sanitized transaction database *D'*, such that only the nonSHUFIs can be mined from sanitized database *D'*.

Definition 8 (Side effects). The side effects of the SHUFI hiding process are the difference of the content of data items and the mining results of the HUFI algorithm between the original database and the sanitized database. They are:

- Missing Cost (MC): The MC is the ratio of the number of non-sensitive high utility and frequent itemsets lost by the hiding process and the number of original ones, and is defined as follows:

(1) $$\mathrm{MC} = \frac{|\mathrm{nonSHUFIs} \backslash \mathrm{nonSHUFIs}'|}{|\mathrm{nonSHUFIs}|}.$$

Where $\mathrm{nonSHUFIs}$ and $\mathrm{nonSHUFIs}'$ are non-sensitive high utility and frequent itemsets mined from original database *D* and sanitized database *D'*, respectively.

- Hiding Failure (HF): The HF is the number of SHUFI discovered from the sanitized database, namely:

(2) $$\mathrm{HF} = |\mathrm{SHUFIs} \cap \mathrm{HUFIs}'|.$$

- The similarity rate reflects the difference between the original database and the sanitized database, including: Database Structure Similarity (DSS), Database Utility Similarity (DUS), and Itemsets Utility Similarity (IUS), which are defined as [7]:

(3) $$\mathrm{DSS} = \sqrt{\sum_{k=1}^{\left|\mathrm{tp}_k^D \cup \mathrm{tp}_k^{D'}\right|} \left(f(\mathrm{tp}_k^D) - f\left(\mathrm{tp}_k^{D'}\right)\right)^2},$$

where tp^D and $\mathrm{tp}^{D'}$ are the set of transaction pattern in the original database *D* and the sanitized database *D'*, respectively; $f\left(\mathrm{tp}_k^D\right)$ and $f\left(\mathrm{tp}_k^{D'}\right)$ are frequency of a pattern in original database *D* and sanitized database *D'*.

(4) $$\mathrm{DUS} = \frac{\sum_{T_c \in D'} \mathrm{tu}(T_c)}{\sum_{T_c \in D} \mathrm{tu}(T_c)},$$

(5) $$\mathrm{IUS} = \frac{\sum_{X \in \mathrm{HUFIs}'} u(X)}{\sum_{X \in \mathrm{HUFIs}} u(X)}.$$

4. Algorithm proposal

In this section we propose a novel algorithm named HSUFIBL (Hiding Sensitive high Utility and Frequent Itemsets Based on constrained intersection Lattice) to efficiently hide sensitive high utility and frequent itemsets. This algorithm includes three stages. In the first stage, the maximal border theory is applied to analyze which method between support reduction or utility reduction is the best for minimizing side effects. In the second stage, a heuristic strategy is proposed to select victim item and victim transaction for data modification. The third stage is data modification. The detail is presented as follows.

4.1. Analyzing the impact of data modification and methods for hiding strategy

A transaction database D is considered, with a minimum utility threshold ε and a minimum support threshold δ; S_i is i-th sensitive high utility and frequent itemset that needs to be hidden. Itemset S_i is hidden if its support is less than δ or its utility is less than ε. In this paper, we reduce the support or utility of S_i by modifying the value of a sensitive item of S_i in an original database. This process is called database sanitization.

Therefore, the process for hiding itemset S_i can be done by one of the following methods:

- **Method 1 (Hiding S_i by support reduction).** To reduce the support of S_i below a minimum support threshold; this method deletes an item belonging to S_i at a transaction supporting S_i. The minimal support value needed to be reduced to hide S_i is

(6) $$\mathrm{ds} = \mathrm{supc}(S_i) - \lceil \delta * |D| \rceil + 1.$$

- **Method 2 (Hiding S_i by utility reduction).** To reduce the utility of S_i below the minimum utility threshold; this method reduces the internal utility of an item belonging to S_i at a transaction supporting S_i. The minimal utility value needed to be reduced to hide S_i is

(7) $$\mathrm{du} = u(S_i) - \varepsilon + 1.$$

The process of data modification for the hiding process always causes side effects. The target of the hiding algorithm is to hide high utility and frequent itemsets with minimal side effects. In general, the side effect of sensitive high utility and frequent itemset hiding depends on:

- Which method is selected for the hiding algorithm?
- Which strategy is applied for selecting victim item and victim transaction?

The victim item (i_{vic}) is an item of a sensitive high utility and frequent itemset S_i, so that modifying i_{vic} from a transaction causes minimal side effects. The victim transaction (T_{vic}) is a transaction containing sensitive high utility and frequent itemset

S_i, so that modifying the internal utility of $i_{\text{vic}} \in S_i$ from T_{vic} causes minimal side effects.

Victim item i_{vic} is selected for modification at the victim transaction T_{vic} in order to hide the sensitive itemset S_i. This means that modifying utility or support of victim item leads to the reduction of utility or support of S_i, respectively. Let SA be the high utility and frequent itemset affected by modifying the victim item i_{vic} at the victim transaction T_{vic}. Then setSA is the set of SA given by

$$\text{setSA} = \{\text{SA} | i_{\text{vic}} \in \text{SA}, \text{SA} \subseteq T_{\text{vic}}\}.$$

Property 1. If i_{vic} is deleted from transaction T_{vic} then:

- Utility of S_i will be decreased by $u(S_i, T_{\text{vic}})$ and the utility of every itemset in $\text{SA} \in \text{setSA}$ will also be decreased by $u(\text{SA}, T_{\text{vic}})$.
- Support count of S_i and every itemset $\text{SA} \in \text{setSA}$ will be decreased by 1, that is the support of S_i and every itemset in $\text{SA} \in \text{setSA}$ will be decreased by $1/|D|$.

Proof: Suppose that $\text{STS}_i = \{\text{ST}_1, \text{ST}_2, \dots, \text{ST}_k\}$ is a set of transaction supporting S_i, then $u(S_i) = \sum_{\text{ST}_j \in \text{STS}_i} u(S_i, \text{ST}_j)$, $\text{supc}(S_i) = |\text{STS}_i|$ and $\text{support}(S_i) = \text{supc}(S_i) / |D|$. If i_{vic} is deleted at transaction $T_{\text{vic}} \in \text{STS}_i$ then transaction T_{vic} does not support S_i and STSi will be replaced by $\text{STS}_i = \text{STS}_i \backslash T_{\text{vic}}$. Thus, if i_{vic} is deleted at T_{vic} then $u(S_i) = u(S_i) - u(S_i, T_{\text{vic}})$, $\text{supc}(S_i) = \text{supc}(S_i) - 1$ and $\text{support}(S_i) = \text{support}(S_i) - 1/|D|$.

Similarly, if i_{vic} is deleted at $T_{\text{vic}} \in \text{STS}_i$ then for an itemset $\text{SA} \in \text{setSA}$ we have $u(\text{SA}) = u(\text{SA}) - u(\text{SA}, T_{\text{vic}})$, $\text{supc}(\text{SA}) = \text{supc}(\text{SA}) - 1$ and

$$\text{support}(\text{SA}) = \text{support}(\text{SA}) - 1 / |D|.$$

Property 2. If the internal utility of item i_{vic} at transaction T_{vic} is decreased by a value r, $r \in N$ then:

- If $r < q(i_{\text{vic}}, T_{\text{vic}})$ then the utility of S_i and the utility of an itemset $\text{SA} \in \text{setSA}$ is decreased by a value $r * p(i_{\text{vic}})$, meanwhile the support of S_i and itemsets $\text{SA} \in \text{setSA}$ are not affected.
- If $r = q(i_{\text{vic}}, T_{\text{vic}})$ then the side effect of hiding process is equal to the side effect of deleting item i_{vic} from T_{vic}.

Proof: If the internal utility of i_{vic} at T_{vic} is decreased by a value $r < q(i_{\text{vic}}, T_{\text{vic}})$, this means that the internal utility of i_{vic} at T_{vic} is updated by a new value $q(i_{\text{vic}}, T_{\text{vic}}) = q(i_{\text{vic}}, T_{\text{vic}}) - r > 0$, then item i_{vic} is not deleted from T_{vic}. Therefore, the support of S_i and itemset $\text{SA} \in \text{setSA}$ is not affected. The internal utility of i_{vic} at T_{vic} is updated by a new value $q(i_{\text{vic}}, T_{\text{vic}}) = q(i_{\text{vic}}, T_{\text{vic}}) - r$. Consequently, the utility of S_i and itemsets $\text{SA} \in \text{setSA}$ is decreased by a value $r * p(i_{\text{vic}})$.

If $r = q(i_{\text{vic}}, T_{\text{vic}})$ then the internal utility of i_{vic} at T_{vic} is updated by a new value $q(i_{\text{vic}}, T_{\text{vic}}) = q(i_{\text{vic}}, T_{\text{vic}}) - r = 0$. This means that i_{vic} is deleted from T_{vic}. As a result, according to Property 1, the utility of S_i is decreased by a value $u(S_i, T_{\text{vic}})$ and the utility of itemset $\text{SA} \in \text{setSA}$ is decreased by a value $u(\text{SA}, T_{\text{vic}})$ at the same time. The support count of S_i and every itemset $\text{SA} \in \text{setSA}$ is decreased by 1. This means that their support is decreased by $1/|D|$.

In order to select the method for hiding sensitive high utility and frequent itemset, L i u, X u and L v [12] have defined a maximal utility border and a minimal utility border notions for sensitive itemset S_i as follows.

Definition 9. Given $\mathrm{ST}S_i = \{\mathrm{ST}_1, \mathrm{ST}_2, \ldots, \mathrm{ST}_k\}$ is a set of transactions supporting S_i, which is sorted in descending order of $u(S_i, \mathrm{ST}_j), \mathrm{ST}_j \in \mathrm{ST}S_i$. The maximal utility border of S_i which is denoted by $\mathrm{Bd}_{\max}$ and the minimal utility border of S_i which is denoted by $\mathrm{Bd}_{\min}$ are respectively defined as follows:

$$(8) \quad \begin{cases} \mathrm{Bd}_{\max} = \sum_{j=1}^{\mathrm{ds}} u(S_i, \mathrm{ST}_j), \mathrm{ST}_j \in \mathrm{STS}_i, \\ \mathrm{Bd}_{\min} = \sum_{j=k-\mathrm{ds}+1}^{k} u(S_i, \mathrm{ST}_j), \mathrm{ST}_j \in \mathrm{STS}_i. \end{cases}$$

Basing on the value of $\mathrm{Bd}_{\max}$ and $\mathrm{Bd}_{\min}$, the authors [12] proposed the conditions for selecting the best method for hiding sensitive high utility and frequent itemset and they are presented in Property 3 and Property 4.

Property 3 [12]. If $\mathrm{Bd}_{\max} \leq \mathrm{du}$ then hiding S_i by Method 1 or Method 2 causes the same side effect, however, Method 2 has achieved better performance because it needs less CPU-time.

Property 4 [12]. If $\mathrm{Bd}_{\min} > \mathrm{du}$ then hiding S_i by Method 2 achieves better results than by Method 1 in both minimizing side effect and CPU-Time.

By Property 3 and Property 4, the conditions to decide which method for hiding the itemset S_i with lower side effect is specified as follows:

$$(9) \quad \begin{cases} \text{If } \mathrm{Bd}_{\max} \leq \mathrm{du} \text{ then Method 1 selected,} \\ \text{If } \mathrm{Bd}_{\min} > \mathrm{du} \text{ then Method 2 selected.} \end{cases}$$

In case of $\mathrm{Bd}_{\max} > \mathrm{du}$ and $\mathrm{Bd}_{\min} \leq \mathrm{du}$ then neither Method 1 nor Method 2 is specified as the best. For this case, the HUFI algorithm [12] selects Method 2 for the hiding process. According to Property 1 and Property 2, the impact on non-sensitive high utility and frequent itemset caused by using Method 2 is less than the one caused by using Method 1.

For this reason, we apply Method 1 for the case of $\mathrm{Bd}_{\max} \leq \mathrm{du}$. Otherwise, we apply Method 2 for the remaining cases.

4.2. Victim transaction and victim item selection

The HUFI algorithm [12] applies the same strategy to select victim item and victim transaction for both cases using Method 1 and Method 2 for the hiding process. In this paper, we propose a novel algorithm, named HSUFIBL, that applies different strategies to select victim item and victim transaction for each case using Method 1 or Method 2. This contributes to reducing the side effects of the hiding process.

- **Case 1.** Applying Method 2 for hiding sensitive high utility and frequent itemset:

Property 5. Let S_i be a sensitive high utility and frequent itemset that needs to be hidden. Let two items $x, y \in S_i$, and two transactions $\mathrm{ST}x \supseteq S_i$ and $\mathrm{ST}y \supseteq S_i$ be given. If $u(x, \mathrm{ST}x) \geq \mathrm{du}$ and $u(y, \mathrm{ST}y) < \mathrm{du}$, then modifying item *x* at transaction ST*x* causes less side effect than modifying item *y* at transaction ST*y*.

Proof: In order to hide S_i, the internal utility of the victim item i_{vic} at the victim transaction T_{vic} must be decreased by a value $r = \lceil \mathrm{du}/p(i_{\mathrm{vic}}) \rceil$. In case of $u(x, \mathrm{ST}x) \geq \mathrm{du}$, S_i is hidden after subtracting a value *r* from the internal utility of

item x at transaction STx. This needs once to modify the data. Because $u(x, \mathrm{ST}x) \geq \mathrm{du}$ and $q(x, \mathrm{ST}x) \geq r$, the internal utility of item x at transaction STx is updated by a new value as $q(x, \mathrm{ST}x) = q(x, \mathrm{ST}x) - r \geq 0$. In case of $u(y, \mathrm{ST}y) < \mathrm{du}$, meaning that $q(y, \mathrm{ST}y) < r$ item y needs to be removed from transaction STy. Although item y is removed from STy, the itemset S_i is not hidden .

Consequently, if the item x is modified at transaction STx, then the hiding process needs once to modify the database. Otherwise, if the item y is modified at transaction STy then the hiding process needs more than one time to modify data. By Property 2, reducing $u(x, \mathrm{ST}x)$ causes lower side effect compared to reducing $u(y, \mathrm{ST}y)$.

By Property 5, if there is an item $x \in S_i$ and transaction $\mathrm{ST}x \in \mathrm{STS}_i$ such that $u(x, \mathrm{ST}x) \geq \mathrm{du}$, then the victim item will be $i_{\mathrm{vic}} = x$ and the victim transaction will be $T_{\mathrm{vic}} = \mathrm{ST}x$. In case that there is more than one item x such that $u(x, \mathrm{ST}x) \geq \mathrm{du}$, then victim item is selected from one of such items x in such a way that $u(i_{\mathrm{vic}}, T_{\mathrm{vic}})$ achieves the maximal value, in the other words:

(10) $(i_{\mathrm{vic}}, T_{\mathrm{vic}}) = \mathrm{argmax}_{(x,\mathrm{ST}x)}\{u(x, \mathrm{ST}x) | u(x, \mathrm{ST}x) \geq \mathrm{du}, \mathrm{ST}x \in \mathrm{STS}_i\}.$

If there is no item $x \in S_i$ and no transaction $\mathrm{ST}x \in \mathrm{STS}_i$ such that $u(x, \mathrm{ST}x) \geq \mathrm{du}$, meaning that $u(x, \mathrm{ST}x) < \mathrm{du}$, then item x at transaction STx must be reduced to zero (this means that the item x is removed from STx) in order to reduce the utility of S_i. Therefore, victim transaction is selected as:

(11) $$T_{\mathrm{vic}} = \mathrm{argmax}_{\mathrm{ST}} = \{u(S_i, \mathrm{ST}), \mathrm{ST} \in \mathrm{STS}_i\}.$$

This aims to make $u(S_i)$ to be quickly reduced below the minimal utility threshold, so that the data modification can be done faster. Accordingly, the victim item has lowest support among high utility itemsets supported by the transaction T_{vic}. This helps to minimize the number of itemsets affected by the hiding process. Namely,

(12) $i_{\mathrm{vic}} = \mathrm{argmin}_x = \{f(x), x \in S_i \wedge f(x) = |X \in \mathrm{HUFIs} | X \subseteq T_{\mathrm{vic}} \wedge x \in X|\}.$

- Case 2: Applying Method 1 for hiding sensitive high utility and frequent itemset:

For this case, sensitive high utility and frequent itemset S_i will be hidden based on the support reduction strategy. Therefore, the transaction T_{vic} at which S_i gains the minimal utility among transactions containing it is selected as a victim transaction. This means that $u(S_i)$ reaches a minimum at T_{vic}, thus:

(13) $$T_{\mathrm{vic}} = \mathrm{argmin}_{\mathrm{ST}} = \{u(S_i, \mathrm{ST}), \mathrm{ST} \in \mathrm{STS}_i\}.$$

Modifying the victim item at T_{vic} causes the least impact on non-sensitive high utility and frequent itemsets.

In order to build the strategy for specifying victim item using lattice theory [6] and intersection lattice theory proposed by L e et al. [9], and L e, S. A r c h-I n t, and N. A r c h-I n t [13], we propose a new concept of constrained intersection lattice for a set of high utility and frequent itemsets.

Definition 10 (The set lattice) [6]. Let U be a finite and non-empty set. The subsets of U are denoted by X, Y, … The universal space of subsets of U is said to be the power set of U and is denoted by $\mathrm{Poset}(U)$. It is an ordered set under relation operator $\subseteq$. The least upper bound and the greatest lower bound of two subsets $\{X, Y\}$ are $\sup(\{X, Y\})$ and $\inf(\{X, Y\})$, respectively. $(\mathrm{Poset}(U); \subseteq)$ is a lattice if and only

if existing $\sup(\{X,Y\})$ and $\inf(\{X,Y\})$ for any $X,Y \in U$ where $\sup(\{X,Y\}) = X \cup Y$ and $\inf(\{X,Y\}) = X \cap Y$. If $\mathcal{L} \subseteq U$ then $(\mathcal{L};\subseteq)$ is a lattice if $\sup(\{X,Y\}) = X \cup Y$ and $\inf(\{X,Y\}) = X \cap Y$ for any $X,Y \in \mathcal{L}$. Similarly, if $(\mathcal{L};\subseteq)$ is a lattice with $\inf(\{X,Y\}) = X \cap Y$ for any $X,Y \in \mathcal{L}$ then $(\mathcal{L};\subseteq)$ is called a semi-lattice closed under intersection operator, and called an intersection lattice.

Definition 11 (Intersection lattice of frequent itemsets) [9]. Let FIs be the set of Frequent Itemsets in a database *D*. Then $(\text{FIs};\subseteq)$ is an intersection lattice.

The $(\text{FIs};\subseteq)$ is an intersection lattice because by Apriori property [1], $\forall X,Y \in \text{FIs}$ we have $\inf(\{X,Y\}) = X \cap Y \in \text{FIs}$. In the other words, $(\text{FIs};\subseteq)$ is closed under intersection operator $\cap$.

However, $(\text{HUFIs};\subseteq)$ is not an intersection lattice because it does not satisfy Apriori property. In other words, $(\text{HUFIs};\subseteq)$ is not closed under intersection operator. For example, let us recall the set HUFIs given in Table 3, we have itemsets $\text{BF}, \text{DF} \in \text{HUFIs}$ but $\text{BF} \cap \text{DF} \notin \text{HUFIs}$.

In order to apply the property of intersection lattice for designing an improved hiding strategy, we define a new concept of intersection operator and call it as constrained intersection operator.

Definition 12 (Constrained intersection operator). Given $\{\text{HUFIs} \cup \emptyset\}$, a constrained intersection operator $\cap_H$ is specified as follows: $\forall X,Y \in \text{HUFIs}$,

$$X \cap_H Y = \begin{cases} X \cap Y \text{ if } X \cap Y \in \text{HUFIs}, \\ \emptyset \text{ if } X \cap Y \notin \text{HUFIs}. \end{cases} \quad (14)$$

Definition 13 (Constrained intersection lattice of HUFIs). Let $\{\text{HUFIs} \cup \emptyset\}$ be a set satisfying the constrained intersection operator $\cap_H$, $(\text{HUFIs} \cup \emptyset;\subseteq)$ is a constrained intersection lattice, where $\inf(\{X,Y\}) = X \cap_H Y$ for $\forall X,Y \in \text{HUFIs}$.

This means that the constrained intersection lattice is closed under the constrained intersection operator.

Definition 14. The generating set of HUFIs, denoted by $\text{Gen}(\text{HUFIs})$, is the smallest subset of HUFIs such that for every itemset in HUFIs can be generated by constrained intersection of some itemsets in $\text{Gen}(\text{HUFIs})$. In the other words,

$$\text{HUFIs} = \left\{X \middle| X = \bigcap_{k \in N^*} Y^k, Y^k \in \text{Gen}(\text{HUFIs})\right\}. \quad (15)$$

The set $\text{Gen}(\text{HUFIs})$ can be computed as follows:

$$\text{Gen}(\text{HUFIs}) = \{X \in \text{HUFIs} | d(X) \leq 1\}, \quad (16)$$

where $d(X) = |\{Y \in \text{HUFIs} | X \subset Y\}|$.

For example, the generated set of HUFIs in Table 3 is

$$\text{Gen}(\text{HUFIs}) = \{\text{AC}, \text{AE}, \text{BD}, \text{DE}, \text{EF}, \text{ABC}, \text{ABE}, \text{ABF}, \text{BDF}, \text{DEF}\}.$$

Definition 15. For each constrained intersection lattice $(\text{HUFIs} \cup \emptyset;\subseteq)$, the set of maximal HUFIs is denoted by $\text{Coatom}(\text{HUFIs})$, and is defined as follow:

$$\text{Coatom}(\text{HUFIs}) = \{X | X \in \text{Gen}(\text{HUFIs}) \wedge d(X) = 0\}. \quad (17)$$

For example, the set containing maximal nodes of $(\text{HUFIs} \cup \emptyset;\subseteq)$ in Fig. 1 is $\text{Coatom}(\text{HUFIs}) = \{\text{ABC}, \text{ABE}, \text{ABF}, \text{BEF}, \text{DEF}\}$.

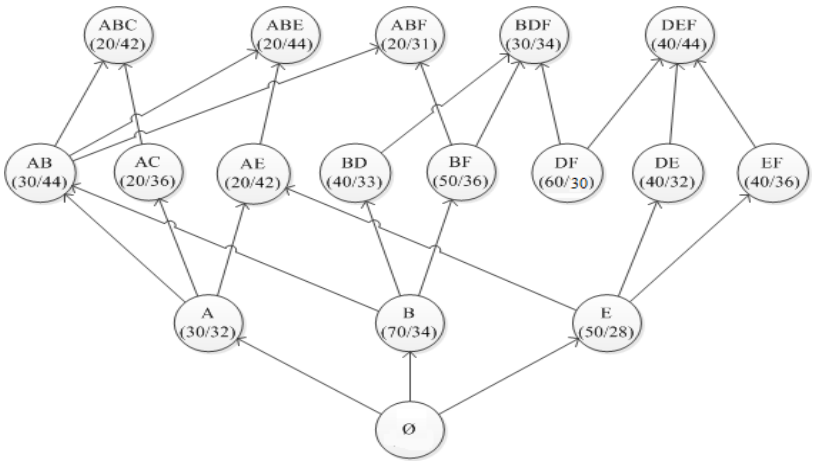

Fig. 1. The constrained intersection lattice of HUFIs presented in Table 3 (the value inside each node shows the HUFI and its Support/Utility)

Property 6. Itemsets in the set $\mathrm{Coatom}(\mathrm{HUFIs})$ have the lowest support compared to the others in the constrained intersection lattice $(\mathrm{HUFIs} \cup \emptyset; \subseteq)$.

Proof: By Definition 15 and Apriori property [1] we achieve the proof.

The constrained intersection lattice $(\mathrm{HUFIs} \cup \emptyset; \subseteq)$ contains sensitive high utility and frequent itemset and non-sensitive high utility and frequent itemset. Hiding a sensitive high utility and frequent itemset and non-sensitive high utility and frequent itemset means to remove it from the constrained intersection lattice. This can be done by reducing support (Method 1) or decreasing utility (Method 2) of SHUFIs. However, removing a node from the lattice by Method 1 affects the related nodes and this may lead to results that some non-sensitive nodes are hidden. Therefore, the important requirement for selecting a victim item is to select the item in such a way that reducing its support causes the least impacts to the other nodes of the lattice.

Definition 16. Given the itemset $C \in \mathrm{Coatom}(\mathrm{HUFIs})$, the set of high utility and frequent itemsets generated by C is denoted by $\mathrm{GenU}(C)$ and it is defined as follows:

(18) $\mathrm{GenU}(C) = \{C \backslash i_k, i_k \in C, k = 1, \dots, |C| | u(C \backslash i_k) \geq \varepsilon \wedge \mathrm{support}(C \backslash i_k) \geq \delta\}$.

For example, $\mathrm{GenU}(\mathrm{ABC}) = \{\mathrm{AB}, \mathrm{AC}\}, \mathrm{GenU}(\mathrm{DEF}) = \{\mathrm{DE}, \mathrm{DF}, \mathrm{EF}\}$.

By Apriori property [1], itemsets in the set $\mathrm{GenU}(C)$ have the lowest supports compared to the sub-itemsets of C. Therefore, they will be lost first when reducing support of their sub-itemsets. This means that the nodes in $\mathrm{GenU}(C)$ are the easiest to be removed from the lattice, when hiding a sensitive node by reducing its support using Method 1. Therefore, if a sensitive node S_i is hidden by using Method 1, then its supper itemsets in $\mathrm{Coatom}(\mathrm{HUFIs})$ will be hidden.

Theorem 1. If a node $X \in \mathrm{GenU}(C)$ in the constrained intersection lattice $(\mathrm{HUFIs} \cup \emptyset; \subseteq)$ is not hidden by using Method 1, then every node $X' \subset X$ in the lattice $(\mathrm{HUFIs} \cup \emptyset; \subseteq)$ will not be hidden, too.

Proof: By the Apriori property, if $X' \subset X$ then $\mathrm{support}(X) \leq \mathrm{support}(X')$. Therefore, if the itemset X is not hidden when reducing its support $(\mathrm{support}(X) \geq \delta)$, then $\mathrm{support}(X') \geq \delta$. In other words, X' is not hidden.

The process of hiding SHUFIs by using Method 1 may lead to hiding nonHUFIs, and this leads to increasing of side effects. In order to overcome this problem, when using Method 1 for hiding SHUFIs, we apply a heuristic to select the victim item based on protecting the set of itemsets in $\mathrm{GenU}(C)$ which are the easiest vulnerable in the lattice from being removed from the lattice.

4.3. Algorithm proposal

Based on the theory that we developed in Sections 4.1 and 4.2, in this section, we propose an algorithm for hiding sensitive high utility and frequent itemset named HSUFIBL (Algorithm 1) and for specifying victim item to hide SHUFI in case of reducing its support named FindVictimItemBasedOnLattice (Algorithm 2).

Description of Algorithm 1 in pseudo-code is shown in Fig. 2. The algorithm hides sensitive high utility and frequent itemset *S* in the given set SHFUIs, respectively. For each itemset *S*, the algorithm executes three steps, including:

Step 1 (lines 2 to 7). To calculate du and ds, needed to be reduced in order to hide *S*; extract the set of transactions that contain *S*, denoted by STS_i; sort the set STS_i in the descending by value of $u(S, T_c)$, where $T_c \in \mathrm{STS}_i$.

Step 2 (lines 9 to 27). If the loop condition is not satisfied then Step 3 is executed. Otherwise, data modification continues to be executed. Namely:

If $Bd_{\max} \leq \mathrm{du}$ then the algorithm reduces support of the victim item below a given minimum support threshold:

- Specify T_{vic} and i_{vic} to modify: Select the victim transaction in the set STS_i in such a way that utility of *S* is minimal at this transaction; the victim item is specified by Algorithm 2 basing on the intersection lattice HUFIs;
- Data modification: Remove i_{vic} from T_{vic} and remove T_{vic} from STS_i; update du and ds; repeat Step 2.

Otherwise, if $\mathrm{Bd}_{\max} > \mathrm{du}$ then the victim item is modified in order to reduce utility of *S* below the minimum utility:

- Specify T_{vic} and i_{vic}: The tuple $(i_{\mathrm{vic}}, T_{\mathrm{vic}})$ is specified using criteria that the utility of i_{vic} at T_{vic} is maximal among transactions of STS_i;
- If $\mathrm{u}(i_{\mathrm{vic}}, T_{\mathrm{vic}}) \geq \mathrm{du}$ then the algorithm modifies i_{vic} at T_{vic} to be $q(i_{\mathrm{vic}}, T_{\mathrm{vic}}) - \left\lceil \frac{\mathrm{du}}{p(i_{\mathrm{vic}})} \right\rceil$; then set du = 0. Otherwise, if $u(i_{\mathrm{vic}}, T_{\mathrm{vic}}) < \mathrm{du}$ then specify T_{vic} and i_{vic}: T_{vic} is the transaction such that $u(S, T_{\mathrm{vic}})$ is maximal (among transactions in STS_i); i_{vic} is the most frequent item among items of HUFIs supported by T_{vic}; update value of du and ds; Remove i_{vic} from T_{vic}; remove T_{vic} from STS_i; Repeat Step 2.

Step 3. Update the database.

Algorithm 1: HSUFIBL

Input: D: Original database; HUFIs: the set of high utility and frequent itemsets; SHUFIs: the set of sensitive high utility and frequent itemsets; ε: minimal utility threshold; δ: minimal support threshold

Output: D': Sanitized database

1 **foreach**($S \in$ SHUFIs) **do**
2 $\mathrm{du} = u(S) - \varepsilon + 1$;
3 $\mathrm{ds} = \mathrm{supc}(S) - \lceil(\delta) * |D|\rceil + 1$;
4 $\mathrm{STS}_i = \{T_c | S \subseteq T_c, T_c \in D\}$;
5 Sort(STS_i) in descending order of $\mathrm{u}(S, T_c)$;
6 $\mathrm{Bd}_{\max} = \sum_{c=1}^{ds} u(S, T_c)$;
7 $\mathrm{flag} = (\mathrm{Bd}_{\max} \leq \mathrm{du}?\,\mathrm{true}:\mathrm{false})$;
8 **while**$\big((\mathrm{du} > 0)\mathrm{and}(\mathrm{ds} > 0)\big)$ **do**
9 **if**(flag) **then**
10 $T_{\mathrm{vic}} = \mathrm{argmin}_{T_c}\{u(S, T_c), T_c \in \mathrm{STS}_i\}$;
11 $i_{\mathrm{vic}} = \mathrm{FindVictimItemBasedOnLacttice}\big(S, \mathrm{Coatom(HUFIs)}\big)$
12 $\mathrm{du} = \mathrm{du} - u(S, T_{\mathrm{vic}})$;
13 $\mathrm{ds} = \mathrm{ds} - 1$;
14 $q(i_{\mathrm{vic}}, T_{\mathrm{vic}}) = 0$;
15 remove T_{vic} from STS_i;
16 **else**
17 $(i_{\mathrm{vic}}, T_{\mathrm{vic}}) = \mathrm{argmax}_{(x,T_c)}\{u(x, T_c), x \in S_i \wedge T_c \in \mathrm{STS}_i\}$;
18 **if**$(u(i_{\mathrm{vic}}, T_{\mathrm{vic}}) \geq \mathrm{du})$ **then**
19 $q(i_{\mathrm{vic}}, T_{\mathrm{vic}}) = q(i_{\mathrm{vic}}, T_{\mathrm{vic}}) - \left\lceil \frac{\mathrm{du}}{p(i_{\mathrm{vic}})} \right\rceil$;
20 $\mathrm{du} = 0$;
21 **else**
22 $T_{\mathrm{vic}} = \mathrm{argmax}_{T_c}\{u(S, T_c), T_c \in \mathrm{STS}_i\}$;
23 $i_{\mathrm{vic}} = \mathrm{argmin}_x\{f(x), x \in S, f(x) = |X \in \mathrm{HUFIs} | X \subseteq T_{\mathrm{vic}} \wedge x \subseteq X|\}$;
24 $\mathrm{du} = \mathrm{du} - u(S, T_{\mathrm{vic}})$;
25 $\mathrm{ds} = \mathrm{ds} - 1$;
26 $q(i_{\mathrm{vic}}, T_{\mathrm{vic}}) = 0$;
27 remove T_{vic} from STS_i;
28 Update(D);
29 return D;

Fig 2. The HSUFIBL Algorithm

Description of Algorithm 2 in pseudo-code is shown in Fig. 3: Algorithm 2 performs the following steps:

Step 1 (lines 1, 2). Extract the super set of sensitive set S, denoted by Coatom_S from the maximal support sets Coatom(HUFIs). This step aims to create the variable $\mathrm{setTuple}_{\min}$ which will be used to contain the minimal tuples specified at Step 2.

Step 2. (lines 3-9). Scan the set Coatom_S. For each $C \in \text{Coatom}_S$, the algorithm finds the minimal by these steps:

Step 2.1. Find the generating set of *C* (the supper set of *C*)

Step 2.2. Scan *S*, for each $x_l \in S$, specify the minimal tuple as follow:

- In the generating set of *C*, specify the set $\text{set}Gx_l$ including the itemsets which contains item x_l
- In the set $\text{set}Gx_l$, get the set *G*, the set of itemsets which have minimal support among other itemsets.
- Add the minimal tuple $\text{tuple}(x_l, C, G, \lambda)$ into $\text{setTuple}_{\min}$
- Repeat Step 2.2

Step 3 (lines 10, 11). In the $\text{setTuple}_{\min}$, get the tuple $\text{tuple}\big(x_l, C, G, \text{supc}(G)\big)$ in such a way that $\text{supc}(G)$ of this tuple is highest compared to other tuples of $\text{setTuple}_{\min}$. The victim item is x_l of $\text{tuple}\big(x_l, C, G, \text{supc}(G)\big)$.

Algorithm 2: FindVictimItemBasedOnLattice

Input: S **:** The sensitive itemset; Coatom(HUFIs): the set of maximal node

Output: i_{vic}: Victim item

1 $\text{Coatom}_{Si} = \{C, C \in \text{Coatom(HUFIs)} | S \subseteq C\}$;
2 $\text{setTuple}_{\min} = \{\emptyset\}$;
3 **foreach**$(C \in \text{Coatom}_S)$ **do**
4 Compute $\text{GenU}(C)$;
5 **for**$(x_l \in S)$ **do**
6 $\text{set}Gx_l = \{G_k, G_k \in \text{GenU}(C) | x_l \in G_k\}$;
7 $G = \text{argmin}_{G_k}\{\text{supc}(G_k), G_k \in \text{set}Gx_l\}$;
8 $\lambda = \text{supc}(G)$;
9 $\text{setTuple}_{\min} = \text{setTuple}_{\min} \cup \text{tuple}(x_l, C, G, \lambda)$;
10 $i_{\text{vic}} = \text{argmax}_{x_l}\{\lambda, \text{tuple}(x_l, C, G, \lambda) \in \text{setTuple}_{\min}\}$;
11 return i_{vic};

Fig 3. The FindVictimItemBasedOnLattice Algorithm

4.4. Experimental results

4.4.1. Experiment descriptions

- The HSUFIBL algorithm is implemented by the Java programming language and run on a computer with the configuration: CPU Core I5 2.4GHz, RAM 8GB, Windows 10.
- Database: Retail, Mushroom, Chess and Chainstore. These databases have been published by professor Philippe-Fournier. They have been published and available in [4] and are available at **http://www.philippe-fournier-viger.com/spmf/index.php?link=datasets.php**. These databases have been popularly used for experiments of pattern mining and privacy-preserving in pattern mining. Many papers published in well-known journals use those databases for their experiments. The detail of databases is presented in Table 4.

- Sets of sensitive-high utility and frequent itemsets: For each database, SHUFIs are randomly selected from the set HUFIs mined by the HUFIM algorithm [8]. The experiment has been executed with four datasets presented in Table 4 and the corresponding minimal utility threshold and minimal support threshold presented in Table 5.

Table 4. The data sets description

Databases	$\|D\|$	$\|I\|$	Average length	Max length
Retail	88,162	16,470	10.3 76	76
Mushroom	8,124	119	23	23
Chess	3,196	75	37	40
Chainstore	1,112,949	46,086	7.23	170

Table 5. The parameter settings of the four datasets

Databases	Sensitive itemset size	ε(%)	δ(%)	Databases	Sensitive itemset size	ε(%)	δ(%)
Retail	Varied	0.04	0.1	Chess	Varied	12	4
	20	Varied	0.1		20	Varied	4
	20	0.04	Varied		20	12	Varied
Mushroom	Varied	24	75	Chainstore	Varied	0.02	0.015
	20	Varied	75		30	Varied	0.015
	20	24	Varied		30	0.02	Varied

4.4.2. Experimental results and discussions

In this section, we compare the experimental results between our algorithm and the HUFI [12], MSMU, MCRSU [14], EHSHUI[7] algorithms. The results are presented for each side effects and time consuming as follows:

- Missing cost: Figs 4, 5, 6 show the performance of algorithms HSUFIBL and HUFI, MSMU, MCRSU, EHSHUI in comparing the missing cost side effect. The result indicates that the HSUFIBL algorithm achieves better performance. It causes lower MC than the HUFI, MSMU, MCRSU, EHSHUI algorithms in every case of the comparison, including the change of the number of sensitive itemsets, minimum support threshold, and minimum utility threshold. This is clearer in the case of datasets containing long average length of transaction (Chess, Mushroom dataset). The reason for this result is that the HUFI algorithm applies only one method for specifying victim item and victim transaction for both hiding strategies using Method 1 or Method 2. The MSMU, MCRSU algorithms apply both Method 1 and Method 2. The EHSHUI algorithm hides SHUIs by using Method 2. In order to promote optimization of both Method 1 and Method 2, the HSUFIBL algorithm applies a heuristic to select exactly the condition that Method 1 or Method 2 must apply for achieving better performance. Especially, the HSUFIBL algorithm applies a heuristic based on constrained intersection lattice of high utility and frequent itemset to specify victim item for hiding sensitive itemsets for the case of Method 1.

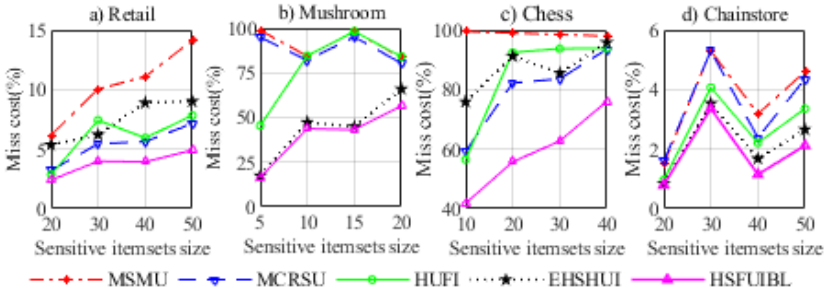

Fig. 4. Miss cost with various sensitive itemset

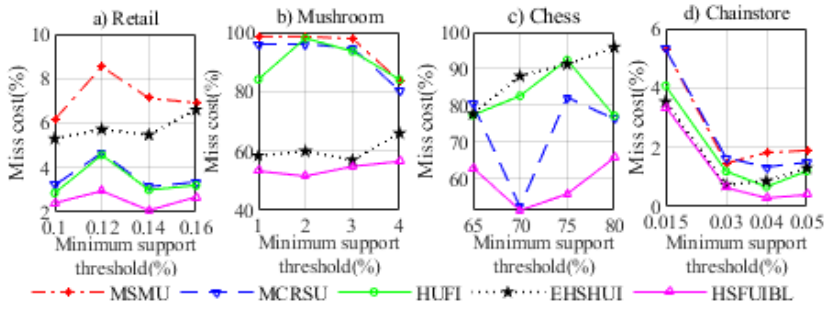

Fig. 5. Miss cost with various minimum support thresholds

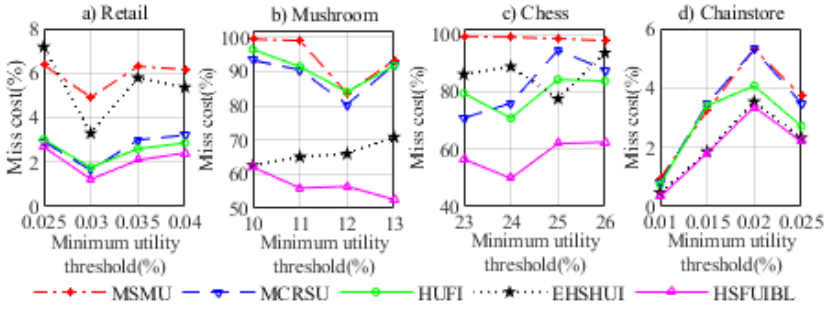

Fig. 6. Miss cost with various minimum utility thresholds

- Database structure similarity, database utility similarity, and itemsets utility similarity: The DSS, DUS, and IUS between the original database and the sanitized database after completing the hiding process performed by the HSFUIBL algorithm and the HUFI, MSMU, MCRSU, EHSHUI algorithms is presented in Figs 7-15. The results show that the DSS, DUS, and IUS produced by the HSFUIBL algorithm is higher than those produced by the other algorithms. This means that hiding sensitive high utility and frequent itemsets by the HSFUIBL algorithm caused the database less distortion than by the HUFI, MSMU, MCRSU, EHSHUI algorithms. The reason for good performance is that the HSFUIBL algorithm tries to reduce the number of iterations for modifying the database when applying the heuristic to select the victim item and the victim transaction.

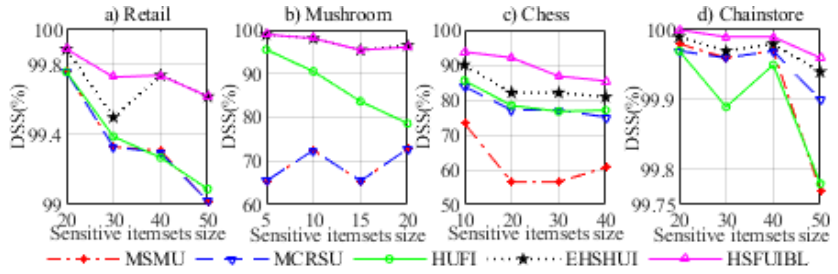

Fig. 7. DSS with various sensitive itemset

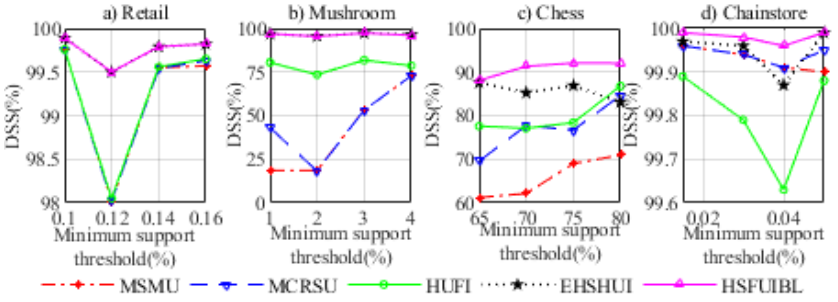

Fig. 8. DSS with various minimum support thresholds

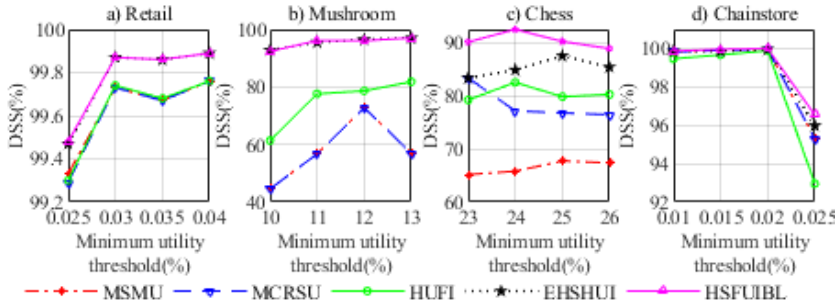

Fig. 9. DSS with various minimum utility thresholds

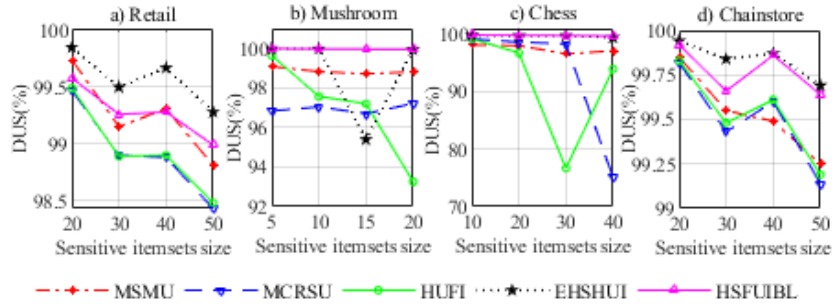

Fig. 10. DUS with various sensitive itemset

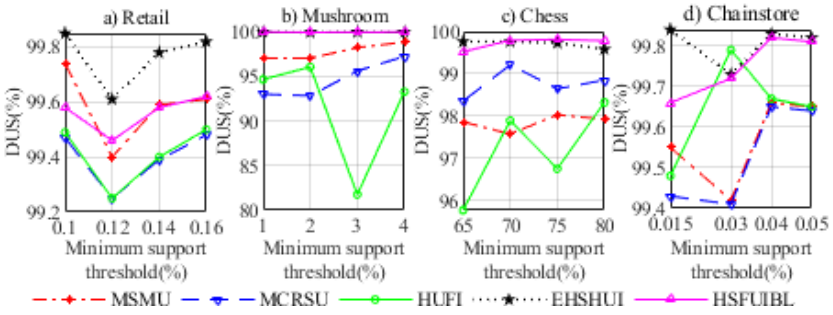

Fig. 11. DUS with various minimum support thresholds

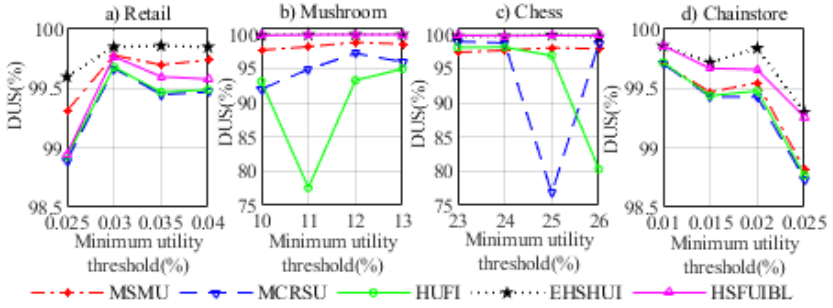

Fig. 12. DUS with various minimum utility thresholds

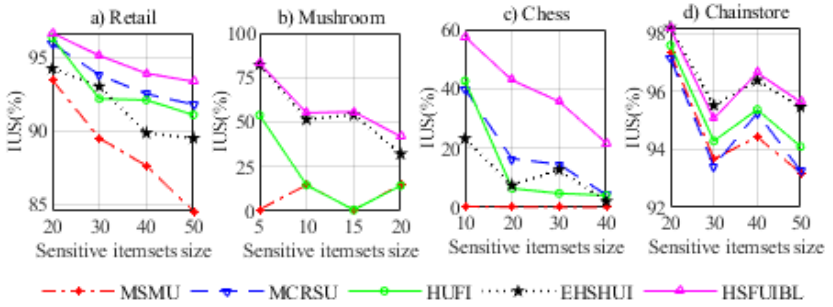

Fig. 13. IUS with various minimum sensitive itemset

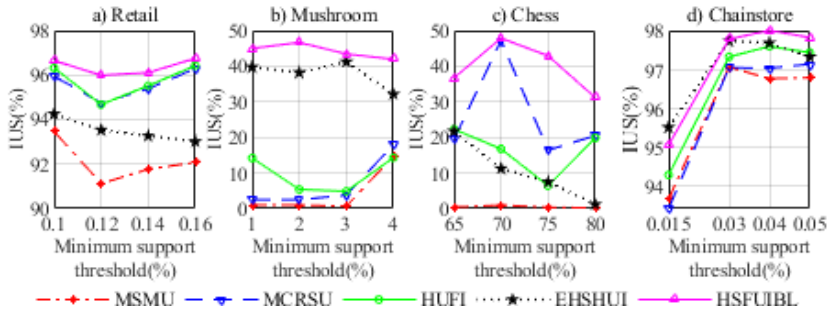

Fig. 14. IUS with various minimum support thresholds

- Run time: MSMU, MCRSU, and EHSHUI algorithms applied Method 2 for finding victim item and victim transaction so that they consume lower running time than HUFI and HSUFIBL for most of the datasets. The running time consumed by

the HUFI algorithm is higher because this algorithm must create the HUFI-table and t-Table. For the HSUFIBL algorithm, running time depends on the victim item specification step for Method 1 because this process needs to scan the lattice at least one time. This is clearer when the sensitive itemset is longer. The experiment results presented in Figs 8, 9, 10 indicate that the running time for hiding sensitive itemsets mined from Chess by using the HSUFIBL algorithm is higher than by using the HUFI algorithm. The reason is that the sensitive itemsets discovered from Chess are long so that the hiding process mostly uses Method 1 and needs more time to find the victim item. For the remaining datasets, the running time needed for the HUFI algorithm is higher than for the HSUFIBL algorithm because the heuristic for finding the victim item and the victim transaction applied in the HSUFIBL algorithm contributes to decrease the time of data modification.

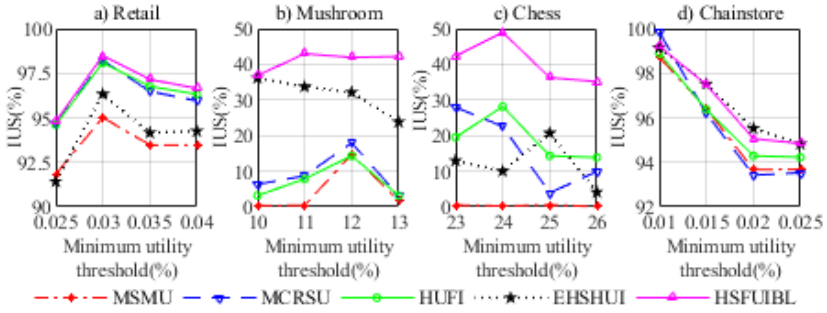

Fig. 15. IUS with various minimum utility thresholds

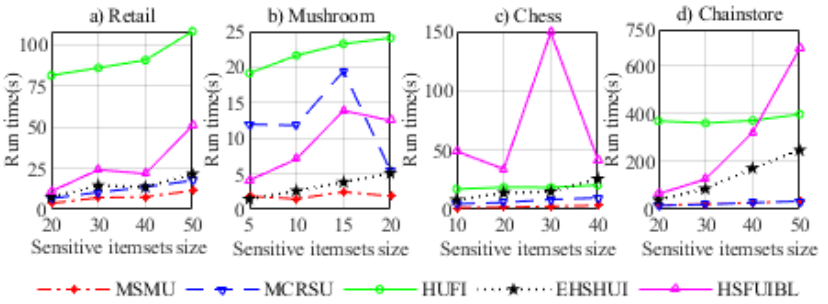

Fig. 16. Run time with various sensitive itemset

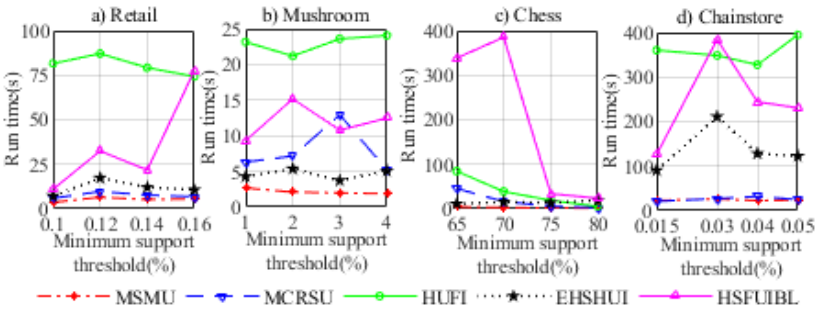

Fig. 17. Run time with various minimum utility thresholds

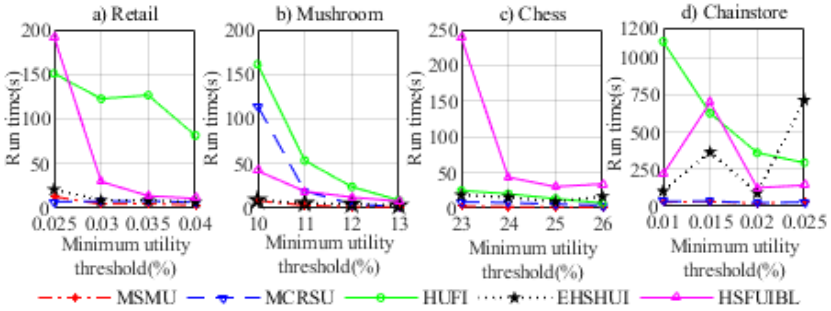

Fig. 18. Run time with various minimum utility thresholds

5. Conclusion

PPUIM is an emerging research area that concentrates on hiding sensitive information implicit in the database in such a way that this information cannot be discovered from the database by data mining techniques.

The target of the PPUIM algorithm is to completely hide sensitive information while minimizing side effects. This paper proposes a heuristic algorithm named HSUFIBL for hiding sensitive high utility and frequent itemsets based on constrained intersection lattice and maximal border theory to make a copy of the database by modifying the victim item from the victim transaction of the original database.

The main contribution of this paper is to propose two heuristic strategies to specify victim item and victim transaction for two cases of support reduction and utility reduction, respectively. By theoretically analyzing and experimentally computing the side effect for each case of data modification, the HSUFIBL algorithm selects the exact method for hiding all sensitive itemsets with minimal side effect. For the support reduction case, a constrained intersection lattice of high utility and frequent itemsets is defined. This theory is applied to propose a heuristic for finding the victim item. For the utility reduction case, the victim items and the victim transactions are specified in such a way that the utility of the victim item reaches maximal value at the victim transaction compared to utility of the other items. The experimental results show that the side effects caused by the HSUFIBL algorithm are mostly lower than the side effects caused by the HUFI, MSMU, MCRSU, EHSHUI algorithms.

Although achieving better results than HUFI, MSMU, MCRSU, and EHSHUI algorithms in almost all criteria of the experiment, the HSUFIBL algorithm still requires more time when running with the dataset which has long transactions. This limitation is caused by the victim item specification step that waists more time to find the maximal itemset and the set of lowest-support itemsets. Our near future work is to improve this drawback for the better algorithm.

References

1. A g a r w a l, R., R. S r i k a n t. Fast Algorithms for Mining Association Rules. – In: Proc. of 20th VLDB Conference, 1994. p. 499.

2. A g r a w a l, R., R. S r i k a n t. Privacy-Preserving Data Mining. – In: Proc. of 2000 ACM SIGMOD International Conference on Management of Data, 2000, pp. 439-450.
3. C h e n g, P., et al. Hide Association Rules with Fewer Side Effects. – IEICE TRANSACTIONS on Information, Vol. **98**, 2015, No 10, pp. 1788-1798.
4. F o u r n i e r-V i g e r, P. 2021 [cited 2021 01/01/2021]. **http://www.philippe-fournier-viger.com/spmf/index.php?link=datasets.php**
5. G a n, W., et al. Privacy Preserving Utility Mining: A Survey. – In: Proc. of 2018 IEEE International Conference on Big Data (Big Data), 2018, IEEE, pp. 2617-2626.
6. G r C a t z e r, G. Lattice Theory: Foundation. 2011. Springer Science & Business Media.
7. H u y n h T r i e u, V., H. L e Q u o c, C. T r u o n g N g o c. An Efficient Algorithm for Hiding Sensitive-High Utility Itemsets. – Intelligent Data Analysis, Vol. **24**, 2020, No 4, pp. 831-845.
8. K i r a n, R. U., et al. Efficiently Finding High Utility-Frequent Itemsets Using Cutoff and Suffix Utility. – In: Proc. of Pacific-Asia Conference on Knowledge Discovery and Data Mining, 2019, Springer, pp. 191-203.
9. L e, H. Q. et al. Association Rule Hiding in Risk Management for Retail Supply Chain Collaboration. – Computers in Industry, Vol. **64**, 2013, No 7, pp. 776-784.
10. L i n, C.-W., et al. A GA-Based Approach to Hide Sensitive High Utility Itemsets. – The Scientific World Journal, Vol. **2014**, 2014.
11. L i n, J. C.-W., et al. Fast Algorithms for Hiding Sensitive High-Utility Itemsets in Privacy-Preserving Utility Mining. – Engineering Applications of Artificial Intelligence, Vol. **55**, 2016, pp. 269-284.
12. L i u, X., F. X u, X. L v. A Novel Approach for Hiding Sensitive Utility and Frequent Itemsets. – Intelligent Data Analysis, Vol. **22**, 2018, No 6, pp. 1259-1278.
13. Q u o c L e, H., S. A r c h-I n t, N. A r c h-I n t. Association Rule Hiding Based on Intersection Lattice. – Mathematical Problems in Engineering, Vol. **2013**, 2013.
14. R a j a l a x m i, R., A. N a t a r a j a n. Effective Sanitization Approaches to Hide Sensitive Utility and Frequent Itemsets. – Intelligent Data Analysis, Vol. **16**, 2012, No 6, pp. 933-951.
15. Y a o, H., H. J. H a m i l t o n, C. J. B u t z. A Foundational Approach to Mining Itemset Utilities from Databases. – In: Proc. of 2004 SIAM International Conference on Data Mining, 2004. SIAM, pp. 482-486.
16. Y e h, J.-S., P.-C. H s u. HHUIF and MSICF: Novel Algorithms for Privacy Preserving Utility Mining. – Expert Systems with Applications, Vol. **37**, 2010, No 7, pp. 4779-4786.

Received: 11.05.2021; Second Version: 06.11.2021; Accepted: 11.01.2022