Hindawi
Mathematical Problems in Engineering
Volume 2020, Article ID 7489045, 14 pages
https://doi.org/10.1155/2020/7489045

Research Article

An Improved Sanitization Algorithm in Privacy-Preserving Utility Mining

Xuan Liu,[1,2] **Genlang Chen**,[1,2] **Shiting Wen**,[1,2] **and Guanghui Song**[1,2]

[1]*Ningbo Institute of Technology, Zhejiang University, Ningbo 315100, China*
[2]*Zhejiang University Ningbo Research Institute, Ningbo 315100, China*

Correspondence should be addressed to Xuan Liu; liuxuan6105@126.com

Received 13 July 2019; Revised 2 March 2020; Accepted 9 March 2020; Published 25 April 2020

Academic Editor: Qiuye Sun

High-utility pattern mining is an effective technique that extracts significant information from varied types of databases. However, the analysis of data with sensitive private information may cause privacy concerns. To achieve better trade-off between utility maximizing and privacy preserving, privacy-preserving utility mining (PPUM) has become an important research topic in recent years. The MSICF algorithm is a sanitization algorithm for PPUM. It selects the item based on the conflict count and identifies the victim transaction based on the concept of utility. Although MSICF is effective, the heuristic selection strategy can be improved to obtain a lower ratio of side effects. In our paper, we propose an improved sanitization approach named the Improved Maximum Sensitive Itemsets Conflict First Algorithm (IMSICF) to address this issue. It dynamically calculates conflict counts of sensitive items in the sanitization process. In addition, IMSICF chooses the transaction with the minimum number of nonsensitive itemsets and the maximum utility in a sensitive itemset for modification. Extensive experiments have been conducted on various datasets to evaluate the effectiveness of our proposed algorithm. The results show that IMSICF outperforms other state-of-the-art algorithms in terms of minimizing side effects on nonsensitive information. Moreover, the influence of correlation among itemsets on various sanitization algorithms' performance is observed.

1. Introduction

Data mining is used to discover the decision-making knowledge and information from massive data [1–4]. In a cooperative project, data are shared among different companies for mutual benefits. However, this brings the risk of disclosing sensitive knowledge contained in a database [5]. Sensitive knowledge can be represented as a set of frequent patterns and high-utility patterns with security implication [6, 7]. Thus, data owner wants to hide sensitive information before a database is released. To solve the problem, privacy-preserving data mining (PPDM) has been proposed and become an important research direction [8]. PPDM methods have been applied in various fields, such as cloud computing, e-health, wireless sensor networks, and location-based services [9].

One way to conceal sensitive knowledge is to sanitize a database by modifying some items in it. Atallah et al. [10] first proved that the optimal sensitive-knowledge-hiding problem is NP-hard and proposed a sanitization algorithm based on heuristic strategy. After that, a lot of works have been completed. However, the existing hiding approaches for protecting high-utility itemsets sanitize a database on the basis of the concept of utility. Meanwhile, the side effects on nonsensitive information are not taken into account. Thus, the damage to nonsensitive knowledge is serious, and database quality is low when a database is modified. To address this problem, we propose an improved approach called the Improved Minimum Sensitive Itemsets Conflict First Algorithm (IMSICF) for hiding sensitive high-utility itemsets. This algorithm is based on the MSICF algorithm and makes the following improvements:

(1) For the victim item selection, the conflict count of each sensitive item is dynamically calculated in the sanitization process, which ensures that the item with the maximum conflict count is chosen to be sanitized.

(2) For the victim transaction selection, the transaction supporting the least number of nonsensitive itemsets

and having the maximal utility of a sensitive itemset is chosen to be modified, which effectively reduces undesired side effects produced by the sanitization process.

(3) The conflict degree is defined to reflect the correlation among sensitive itemsets. Moreover, the influence of conflict degree on sanitization performance is observed.

The rest of the paper is organized as follows. Section 2 reviews related works. In Section 3, preliminary knowledge of high-utility itemsets mining is introduced. Section 4 describes the hidden strategy of the proposed sanitization approach. Section 5 gives experimental results and analysis. Finally, conclusions are made in Section 6.

2. Related Works

In this section, related works on privacy-preserving data mining are reviewed.

Most of previous studies focused on hiding sensitive itemsets from frequent itemset mining approaches. Verykios et al. [11, 12] proposed five approaches for achieving PPDM. The first three algorithms are used to protect sensitive association rules. A sensitive rule is hidden by reducing its support or confidence. The last two algorithms are used to conceal sensitive itemsets. However, all of the five algorithms only hide rules that are supported by disjoint frequent itemsets. Oliveira and Zaïane [13] presented a one-scan algorithm that only needs to scan a database once. The disclosure threshold is introduced to balance privacy protection and knowledge disclosure. Amiri [14] developed three algorithms, namely, aggregate, disaggregate, and hybrid, for hiding sensitive frequent itemsets. Aggregate conceals sensitive itemsets by deleting transactions. Disaggregate sanitizes a database by removing some items. The hybrid algorithm is the combination of the previous two algorithms. In terms of execution time and side effects, the hybrid algorithm is recommended for database sanitization.

Gkoulalas-Divanis and Verykios [15] introduced a new approach for hiding sensitive itemsets by inserting some synthetic transactions into an original database. The hybrid database is generated based on the constraints on the border itemsets. Wu and Huang [16] described two greedy approaches, namely, greedy approximation algorithm and exhausted algorithm, for concealing sensitive association rules. Both algorithms include the sanitization procedure and exposed procedure. The greedy approximation algorithm always outperforms the other one because the cost is recalculated when items are modified. However, the greedy algorithm takes a lot of time to expose missing rules.

Hong et al. [17] used the technique of term frequency-inverse document frequency (TF-IDF) to sanitize a database. The transaction with more sensitive items and less influence on other transactions is selected for modification. However, the scalability of this approach is poor.

Le et al. [18, 19] applied the lattice theory to hide sensitive association rules, and two approaches called HCSRIL and AARHIL were proposed based on the intersection lattice of frequent itemsets. Both algorithms select the victim item with the least impacts on the generating set for sanitization. The AARHIL algorithm has better performance than HCSRIL in terms of missing cost since the victim transaction selection is improved. Shah et al. [20] adopted the genetic algorithm to hide sensitive association rules. The fitness function is used to evaluate whether to modify a transaction or not. The transaction with lower fitness will be modified with higher probability because it contains more sensitive items and minimal number of data items.

Cheng et al. [21–25] applied a multiobjective optimization algorithm into PPDM, such as NSGA-II and Hype. The algorithms in [21–24] conceal sensitive association rules by modifying some items. The sanitization approach in [25] hides sensitive itemsets by removing some items. The key issue in optimization algorithm is to design the objective functions, which are based on the side effects on a database. Besides, Cheng et al. [26] also proposed a greedy algorithm for hiding sensitive rules. The information on nonsensitive itemsets is considered in the selection of the victim transaction. Thus, the side effects are effectively reduced. Lin et al. [27] presented a multiobjective algorithm (NSGA2DT) for hiding sensitive itemsets with transaction deletion. A Fast SoRting strategy and the prelarge concept are utilized to accelerate the iterative process.

The above methods focus on protecting sensitive knowledge in frequent itemset mining. However, it is not suitable to modify the quantities of items in a transactional database. Recently, various methods are developed for protecting high-utility itemsets. Moreover, PPUM has become an important research issue. Yeh et al. [28, 29] presented the HHUIF and MSICF algorithms to conceal sensitive high-utility itemsets. Both algorithms sanitize the original database based on the concept of utility. However, the MSICF algorithm takes the conflict count of sensitive items into account. Rajalaxmi and Natarajan [30] identify the victim transaction with the maximum number of sensitive items. Then, the item with the maximal utility is selected for modification. However, the impact on nonsensitive itemsets is discarded. Lin et al. [31] proposed a GA-based algorithm for PPUM by inserting some appropriate transactions into an original database. A function with three factors is used to determine the transactions for insertion. However, this algorithm produces some spurious itemsets after the sanitization process.

Yun and Kim [32] presented an algorithm called FPUTT to improve the efficiency of the HHUIF algorithm. The tree structure is utilized to accelerate the sanitization process. However, the results of FPUTT in terms of side effects on nonsensitive knowledge are the same as those of HHUIF. Lin et al. [33, 34] then developed two approaches MSU_MAU and MSU_MIU for PPUM. For each sensitive itemset SH_i, the transaction with the maximum utility of SH_i is selected for modification. Then, the victim item is chosen based on the maximum utility or minimum utility. Lin et al. [35] designed a genetic algorithm to hide sensitive HUIs by transaction deletion. The prelarge concept is adopted to accelerate the evolution process. However, some spurious itemsets are produced by the sanitization process. Li et al.

[36] formulate the hiding process as a constraint satisfaction problem. Integer linear programming is adopted in the designed algorithm to obtain a lower ratio of side effects produced in the hiding process.

Rajalaxmi and Natarajan [37] proposed two approaches named MSMU and MCRSU to hide the sensitive frequent and utility itemsets. Both algorithms conceal the itemsets until their support and utility fall below the given thresholds, respectively. Liu et al. [38] presented a novel sanitization algorithm called HUFI to conceal sensitive frequent and utility itemsets. The concept of maximum boundary value is introduced to determine the hidden strategy. Thus, the approach outperforms the other algorithms in minimizing the side effects. Besides the above works, Le et al. [39] proposed an efficient algorithm for hiding high-utility sequential patterns, which relies on a novel structure to enhance the sanitization process.

3. Preliminaries

Some preliminary definitions of high-utility itemsets mining are introduced in this section [40, 41]. In addition, the sanitization problem is described [42, 43].

3.1. Definitions. Let $I = \{I_1, I_2, \ldots, I_m\}$ be a set of distinct items. Let $D = \{T_1, T_2, \ldots, T_n\}$ be a transaction database, where each transaction T_i has a unique identifier TID and $T_i \subseteq I$. Each item has an external utility value, which reflects the importance of an item. An itemset is a subset of I, and it is called a k-itemset if it contains k items.

Definition 1. Each item i_t is assigned an external utility value, which is denoted as $\mathrm{eu}(i_t)$. For example, in Table 1, $\mathrm{eu}(b) = 3$.

Definition 2. Each item i_t in a transaction T is assigned an internal utility value, which is denoted as $\mathrm{iu}(i_t, T)$. For example, in Table 1, $\mathrm{iu}(b, T_1) = 1$.

Definition 3. The utility of item i_t in a transaction T is denoted as $u(i_t, T)$ and defined as $\mathrm{iu}(i_t, T) \times \mathrm{eu}(i_t)$. For example, in Table 1, $u(d, T_2) = 1 \times 6 = 6$.

Definition 4. The utility of itemset SH_i in a transaction T is denoted as $u(\mathrm{SH}_i, T)$ and defined as $\sum_{i_t \in \mathrm{SH}_i} u(i_t, T)$. For example, in Table 1, $u(\{b, c\}, T_3) = 3 + 4 = 7$.

Definition 5. The utility of itemset SH_i is denoted as $u(\mathrm{SH}_i)$ and defined as $\sum_{\mathrm{SH}_i \in T} u(\mathrm{SH}_i, T)$. For example, in Table 1, $u(\{b, c\}) = 7 + 16 + 19 = 42$.

Definition 6. The utility of a transaction T is denoted as $\mathrm{tu}(T)$ and defined as $\sum_{i_t \in T} u(i_t, T)$. For example, in Table 1, $u(T_2) = 2 + 6 = 8$.

Definition 7. The user-specified minimum utility threshold is denoted as *minutil*. A pattern X is a high-utility itemset if $u(X) \geq minutil$. Otherwise, it is a low-utility itemset. High-utility itemset mining is to discover the itemsets whose utility values are beyond *minutil*.

Table 1: A transaction database (left) External utility value (right).

TID	Transaction (item, iu)	Item	eu
T_1	(a, 2) (b, 1) (e, 3)	a	5
T_2	(c, 1) (d, 6)	b	3
T_3	(b, 1) (c, 2) (e, 1) (f, 1)	c	2
T_4	(a, 3) (b, 4) (c, 2) (d, 2) (e, 5)	d	1
T_5	(b, 3) (c, 5)	e	6
T_6	(a, 2) (e, 7) (f, 3)	f	10

3.2. Sanitization Problem Description. Given a transaction database D, the minimum utility threshold *minutil*, and the high-utility itemsets mined from D based on *minutil*. $\mathrm{SH} = \{\mathrm{SH}_1, \mathrm{SH}_2, \ldots, \mathrm{SH}_t\}$ is a set of sensitive high-utility itemsets, where SH_i is the itemset that needs to be hidden. A sensitive transaction refers to the transaction supporting at least one sensitive itemset. The sanitization problem is to transform an original database D to a sanitized database D' so that all sensitive itemsets are hidden, while at the same time, the side effects on the database and nonsensitive knowledge are minimized.

One way to sanitize a database is to modify some items in it. The modified item contained in D is the victim item, which is denoted as I_{vic}. The transaction supporting I_{vic} is the victim transaction, which is denoted as T_{vic}.

4. The Hiding Approach

In this section, a sanitization approach named the Improved Maximum Sensitive Itemsets Conflict First Algorithm (IMSICF) is presented in detail. The victim item is selected based on the conflict count, which is calculated dynamically. Moreover, the victim transaction is selected based on the side effects on nonsensitive knowledge, which effectively reduces the missing costs. To better illustrate how the IMSICF algorithm works, an example is given.

4.1. The Sanitization Process of Hiding Sensitive Itemsets

Definition 9. Let $\mathrm{SH} = \{\mathrm{SH}_1, \mathrm{SH}_2, \ldots, \mathrm{SH}_t\}$ be a set of sensitive high-utility itemsets. The conflict count of a sensitive item i_t in SH is denoted as $Icount(i_t)$ and defined as $|\{\mathrm{SH}_i \in \mathrm{SH} \mid i_t \in \mathrm{SH}_i\}|$; that is, $Icount(i_t)$ is the number of sensitive itemsets containing i_t. The conflict degree of SH is defined as

$$\frac{\sum_{i_t \in \mathrm{SH}} Icount(i_t)}{n_i}, \tag{1}$$

where n_i refers to the number of distinct items contained in SH. The conflict degree reflects the correlation among sensitive itemsets. The higher conflict degree indicates that sensitive itemsets have more common items.

For example, given a set of sensitive itemsets $\mathrm{SH} = \{\{b, e\}, \{e, f\}\}$, the conflict degree of SH is 4/3 because $Icount(b) = 1$, $Icount(e) = 2$, and $Icount(f) = 1$.

Definition 10. Let SH_i be a sensitive itemset and *minutil* be the minimum utility threshold. To hide SH_i, the minimum utility to be reduced is defined as $u(SH_i) - minutil + 1$ and denoted as *diffu*, where $u(SH_i)$ is the utility of SH_i.

Let D be a database and SH_i a sensitive itemset. To hide SH_i, the utility of SH_i should be reduced until it falls below the minimum utility threshold; namely, *diffu* of SH_i should be lower than or equal to zero. The strategy of hiding SH_i is to sanitize some items in the selected transactions in D. The sanitization process of hiding SH_i is shown in Figure 1. First, the victim item is identified. Then, a sensitive itemset supporting the victim item is determined. Next, the victim transaction is selected to be modified. After the victim item and transaction are determined, an original database is sanitized. In the following, the sanitization process is described in detail.

4.1.1. The Victim Item Selection. Let D be a database, $SH = \{SH_1, SH_2, \ldots, SH_t\}$ is a set of sensitive itemsets. In the MSICF algorithm, the items in SH are sorted according to the descending order of conflict counts. Then, the victim item is selected based on the sorted results. Because the order of sorted items is fixed, the victim item selection cannot be changed if an original database is modified. Thus, we improve the selection of the victim item. In our approach, the item with the maximum conflict count is selected to be sanitized, that is, $I_{vic} = \text{argmax}_{i_t \in SH} Icount(i_t)$. Once a sensitive itemset is hidden, the conflict count of each sensitive item is recalculated to prevent other sensitive itemsets from being hidden in the sanitization process. In this way, we can make sure that a victim item has the maximum conflict count in the sanitization process.

4.1.2. The Victim Transaction Selection. Let D be a database and SH_i a sensitive itemset. For the MSICF algorithm, the transaction with the maximum utility of a victim item is selected to be a victim transaction. However, the damage to nonsensitive knowledge is not taken into account. To reduce the side effects on nonsensitive information, the transaction that causes the minimum impact should be modified with priority. Thus, we assign a transaction weight to each sensitive transaction, which is used to determine the victim transaction. The transaction weight is computed as

$$tw(T) = \frac{u(SH_i, T)}{1 + NSHC(T)}, \quad (2)$$

where $u(SH_i, T)$ is the utility of SH_i in transaction T and $NSHC(T)$ is the number of nonsensitive itemsets supported by T. The transaction with the maximum utility of SH_i indicates that the deletion of a victim item will decrease more utility. Moreover, the transaction supporting the minimum number of nonsensitive itemsets indicates that the modification of it will generate less side effects. Thus, the transaction having the maximum transaction weight would be sanitized first. Because $NSHC(T)$ is zero when a transaction does not support any nonsensitive itemset, we set the denominator of Formula (2) to $NSHC(T) + 1$.

4.1.3. The Original Database Sanitization. Let SH_i be a sensitive itemset, I_{vic} a victim item, and T_{vic} a victim transaction. If *diffu* of SH_i is not greater than $u(I_{vic}, T_{vic}) - eu(I_{vic})$, it indicates that the victim item is not removed from a victim transaction. Then, $iu(I_{vic})$ is reduced to $iu(I_{vic}) - \lceil diffu/eu(I_{vic}) \rceil$, where $eu(I_{vic})$ is the external utility of I_{vic}. Correspondingly, *diffu* is decreased to $diffu - eu(I_{vic}) * \lceil diffu/eu(I_{vic}) \rceil$. Otherwise, if I_{vic} is removed from the victim transaction, *diffu* is updated to $diffu - u(SH_i, T_{vic})$ rather than $diffu - u(I_{vic}, T_{vic})$. The reason is that SH_i is not supported by T_{vic} if I_{vic} is removed from T_{vic}.

4.2. The Sketch of the IMSICF Algorithm. The pseudocode of the IMSICF algorithm is shown in Algorithm 1. Initially, the conflict count of each sensitive item in SH is calculated (Line 2). Then, the item with the maximum conflict count is selected to be victim item I_{vic} (Line 3). After the sanitized item is identified, the sensitive itemsets containing I_{vic} are hidden one by one. For a sensitive itemset SH_i, the minimum utility to be reduced is computed (Line 5). The utility of SH_i is reduced until *diffu* is less than or equal to zero. To hide SH_i, the sensitive transactions of SH_i are identified. Then, the transaction weight of each transaction is calculated according to Formula (2). The transaction with the maximum weight is selected to be victim transaction T_{vic}. This is reasonable since the minimum side effects on nonsensitive information are generated by modifying the selected T_{vic} (Line 7-8). Next, the victim item in T_{vic} is modified, and the database and itemsets are updated, respectively (Line 9–16). If *diffu* of SH_i is not greater than zero, the sensitive itemset is removed from SH (Line 18). The algorithm is terminated until all sensitive itemsets are hidden.

4.3. An Illustrative Example. For a given transaction database in Table 1, the high-utility itemsets derived at *minutil* = 60 are listed in Table 2. The user-specified sensitive itemsets are $\{b, e\}$ and$\{e, f\}$, which are identified in boldface in Table 2. The proposed algorithm (IMSICF) is applied to hide sensitive itemsets.

To hide the sensitive itemsets $SH = \{\{b, e\}, \{e, f\}\}$, the conflict count of each item contained in SH is calculated. The results are $Icount(b) = 1$, $Icount(e) = 2$, and $Icount(f) = 1$. Item e is selected to be the victim item because it has the maximum conflict count. Then, the sensitive itemset for hiding is randomly selected from among the ones containing the victim item. Let us assume that the selected itemset is $\{b, e\}$. The minimum utility to be reduced is $diffu = 72 - 60 + 1 = 13$, and the sensitive transactions are $ST = \{T_1, T_3, T_4\}$. The transaction weight of each transaction is assigned according to Formula (2). Because $tw(T_1) = 21/4$, $tw(T_3) = 9/2$, and $tw(T_4) = 42/6$, the transaction T_4 is chosen for modification. After the victim item and transaction are identified, the item is sanitized according to the original database sanitization method described in Section 3.1. Because $diffu < u(e, T_4) - eu(e)$, the internal utility of item e is updated to $iu(e, T_4) - \lceil diffu/eu(e) \rceil = 2$. The utility of $\{b, e\}$ is reduced

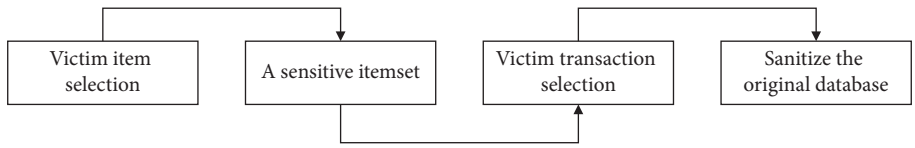

Figure 1: The sanitization process for hiding a sensitive itemset.

Input: a database D, a set of sensitive itemsets $\mathrm{SH} = \{\mathrm{SH}_1, \mathrm{SH}_2, \ldots, \mathrm{SH}_t\}$, the given utility threshold *minutil*.
Output: the sanitized database D'

```
(1)  while SH ≠ ∅
(2)    Calculate Icount(i_p) of each sensitive item i_p, i_p ∈ SH_i.
(3)    I_vic = argmax_{i_p ∈ SH} Icount(i_p)
(4)    for each SH_i ∈ SH ∧ I_vic ∈ SH_i
(5)      diffu = u(SH_i) − minutil + 1
(6)      while diffu > 0
(7)        Find the sensitive transactions ST, SH_i ⊆ ST
(8)        T_vic = argmax_{T∈ST} tw(T)
(9)        if diffu > u(I_vic, T_vic) − eu(I_vic)
(10)         Delete I_vic
(11)         diffu = diffu − u(SH_i, T_vic)
(12)       else
(13)         iu(I_vic) = iu(I_vic) − ⌈diffu/eu(I_vic)⌉
(14)         diffu = diffu − eu(I_vic) ∗ ⌈diffu/eu(I_vic)⌉
(15)       end if
(16)       Update the database and the itemsets
(17)     end while
(18)     SH = SH − SH_i
(19)   end for
(20) end while
```

Algorithm 1: The IMSICF algorithm.

Table 2: Derived high-utility itemsets.

HID	Itemsets	Utility
1	*e*	96
2	*a*, *e*	125
3	**b, e**	72
4	**e, f**	88
5	*a*, *b*, *e*	88
6	*a*, *e*, *f*	82
7	*a*, *b*, *c*, *e*	61
8	*a*, *b*, *c*, *d*, *e*	63

to 54. Thus, {*b*, *e*} is concealed, and the nonsensitive itemsets {*a*, *b*, *c*, *e*} and {*a*, *b*, *c*, *d*, *e*} are falsely hidden. Then, {*e*, *f*} is hidden by following the above steps. After all of the sensitive itemsets are hidden, four nonsensitive itemsets, namely, {*a*, *b*, *c*, *e*}, {*a*, *b*, *c*, *d*, *e*}, {*e*}, and {*a*, *e*, *f*}, are hidden.

5. Experimental Analysis

To evaluate the performance of the IMSICF algorithm, a series of experiments have been conducted on various real and synthetic datasets, in which IMSICF is compared with the state-of-the-art algorithms. Besides, the experimental results are discussed in this section.

5.1. Experimental Data. The experiments were conducted on a 2.8 GHz Intel Xeon E5-2360 processor with 8 GB RAM. To evaluate the performance of the proposed algorithm, four state-of-the-art sanitization algorithms, namely, HHUIF, MSICF, MSU-MIU, and MSU-MAU, were used for comparison. Because the PPUMGAT algorithm hides sensitive itemsets by transaction deletion, we do not compare the proposed algorithm with PPUMGAT. All of the algorithms were implemented in Java language. Four datasets [33] were used to run the programs. The characteristics of these datasets are displayed in Table 3. The density is measured as the average transaction length divided by the number of items. For each dataset, the external utility values were generated with the Gaussian normal distribution, and the internal utility values are the random numbers ranging from 1 to 10. The EFIM algorithm [44] was used to mine high-utility itemsets, and the minimum utility thresholds for mushroom, Foodmart, T25I10D10K, and T20I6D100K were set at 8.66%, 0.045%, 0.24%, and 0.17%, respectively. The sensitive itemsets were randomly selected from the mined itemsets.

The proposed algorithm identifies the victim item based on the conflict count of each sensitive item. Thus, the conflict degree of the sensitive itemsets is presented to observe how the correlation among the itemsets influences the performance of the sanitization algorithms. Besides, the sensitive

Table 3: The characteristics of various databases.

Dataset	No. of transactions	No. of items	Avg. trans. length	Density (%)
Mushroom	8124	119	23	19.3
Foodmart	4141	1559	4	0.25
T25I10D10K	10000	929	24.77	2.66
T20I6D100K	100000	893	19.9	2.22

percentage is used to evaluate the scalability of the sanitization approaches. Sensitive percentage is measured as the number of sensitive itemsets divided by the number of high-utility itemsets. This value of parameter ranges from 0.1% to 0.5%. Moreover, two-way ANOVA is used to evaluate the differences between the compared approaches. Two-way ANOVA is a comparison of means between groups that have been split on two independent variables (called factors). The P value is important because it indicates whether the difference between the sanitization algorithms is significant. If P value is below 0.05, it means that there is a significant difference between the compared approaches. Otherwise, it indicates that there is no significant difference between the sanitization approaches.

5.2. Performance Measurement. To evaluate the efficiency, the execution time of the sanitization process is measured and the data processing stages are discarded. On the other hand, five performance measures are used to evaluate the effectiveness and summarized as follows.

(1) Hiding failure (HF): it is the proportion of the sensitive itemsets that fail to be hidden, which is calculated as

$$\mathrm{HF} = \frac{\mathrm{SH}(D')}{\mathrm{SH}(D)}, \tag{3}$$

where $\mathrm{SH}(D')$ and $\mathrm{SH}(D)$ are the sensitive high-utility itemsets mined from a sanitized database D' and an original database D, respectively.

(2) Missing cost (MC): it is the proportion of the missing nonsensitive itemsets that are hidden by accident after sanitization, which is computed as

$$\mathrm{MC} = \frac{\mathrm{NSH}(D) - \mathrm{NSH}(D')}{\mathrm{NSH}(D)}, \tag{4}$$

where $\mathrm{NSH}(D')$ and $\mathrm{NSH}(D)$ are the nonsensitive itemsets discovered from the databases D' and D, respectively.

(3) Artificial cost (AC): it refers to the proportion of the artificial itemsets, which is computed as

$$\mathrm{AC} = \frac{H(D') - H(D) \cap H(D')}{H(D')}, \tag{5}$$

where $H(D')$ and $H(D)$ are the high-utility itemsets mined from the databases D' and D, respectively.

(4) Itemset utility similarity (IUS): it reveals the utility loss for the discovered itemsets by the sanitization process, which is calculated as

$$\mathrm{IUS} = \frac{\sum_{X \in \mathrm{HUIs}^{D'}} u(X)}{\sum_{X \in \mathrm{HUIs}^{D}} u(X)}, \tag{6}$$

where $\sum_{X \in \mathrm{HUIs}^{D'}} u(X)$ and $\sum_{X \in \mathrm{HUIs}^{D}} u(X)$ denote the utility of the high-utility itemsets discovered from the databases $D\prime$ and D, respectively.

(5) Database utility similarity (DUS): it reveals the utility loss for an original database by the sanitization process, which is calculated as

$$\mathrm{DUS} = \frac{\sum_{T_i \in D'} \mathrm{tu}(T_i)}{\sum_{T_i \in D} \mathrm{tu}(T_i)}, \tag{7}$$

where $\sum_{T_i \in D'} \mathrm{tu}(T_i)$ and $\sum_{T_i \in D} \mathrm{tu}(T_i)$ denote the utility of the databases D' and D, respectively.

5.3. Execution Time. The results of the execution times under various sensitive percentages are plotted in Figure 2. It is clear to see that the runtime is increased with the growth of the sensitive percentage. This is reasonable because the increasing number of sensitive itemsets requires more transactions for modification. From Figure 2, we can also observe that the proposed algorithm takes more time than the other algorithms. The reason is that HHUIF, MSICF, MSU_MIU, and MSU_MAU sanitize the database based on the concept of utility. However, IMSICF needs to calculate the number of nonsensitive itemsets supported by each sensitive transaction, which costs a lot of time. Besides, note that the runtime in mushroom is more than that in the other datasets. The reason is that mushroom is a much denser dataset.

Based on two-way ANOVA, there is a significant difference between the execution times of the sanitization algorithms for various datasets ($P = 3.15 * 10^{-13}$ in Figure 2(a), $P = 2.41 * 10^{-08}$ in Figure 2(b), $P = 2.15 * 10^{-15}$ in Figure 2(c), and $P = 6.41, * 10^{-18}$ in Figure 2(d)).

The results of the execution times under various conflict degrees for different databases are plotted in Figure 3. We can see that the runtime is decreased with the growth of the conflict degree in most cases. This is reasonable because the higher the conflict degree is, the more sensitive the itemsets will be concealed at the same time. From Figure 3, we also find that the proposed algorithm IMSICF costs more time than the other algorithms in most cases. This is because IMSICF takes a lot of time to calculate the number of nonsensitive itemsets supported by each sensitive

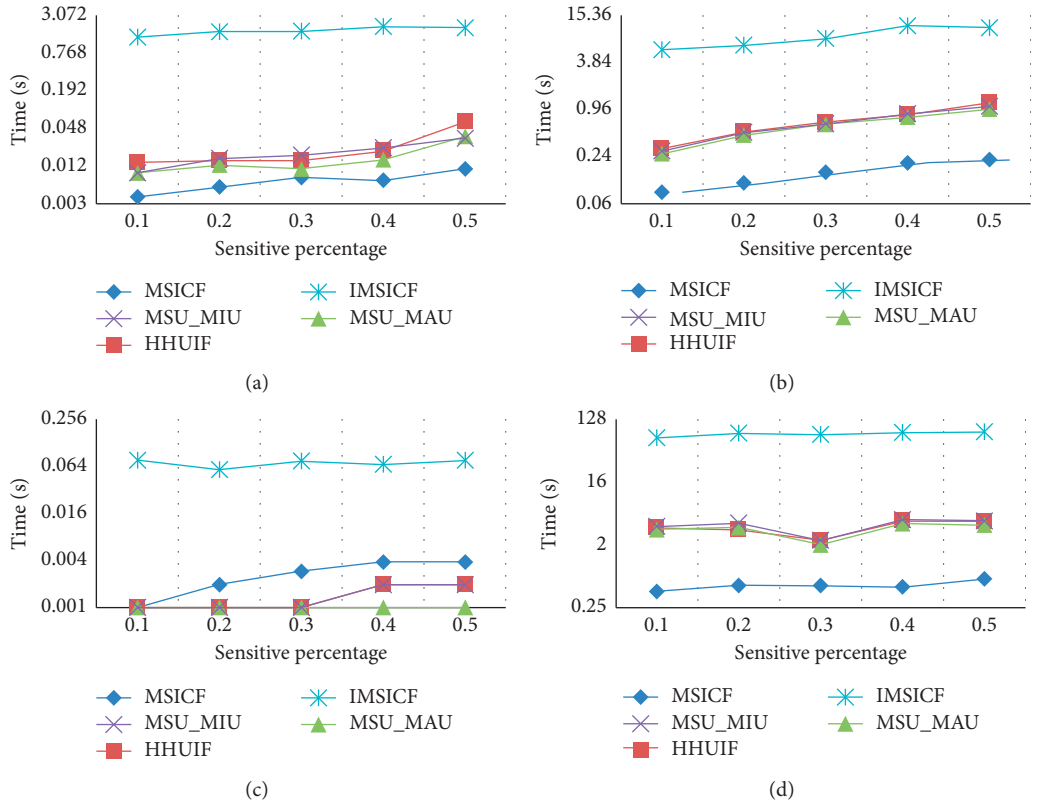

Figure 2: Execution times under various sensitive percentages: (a) T25I10D10K; (b) T20I6D100K; (c) Foodmart; (d) mushroom.

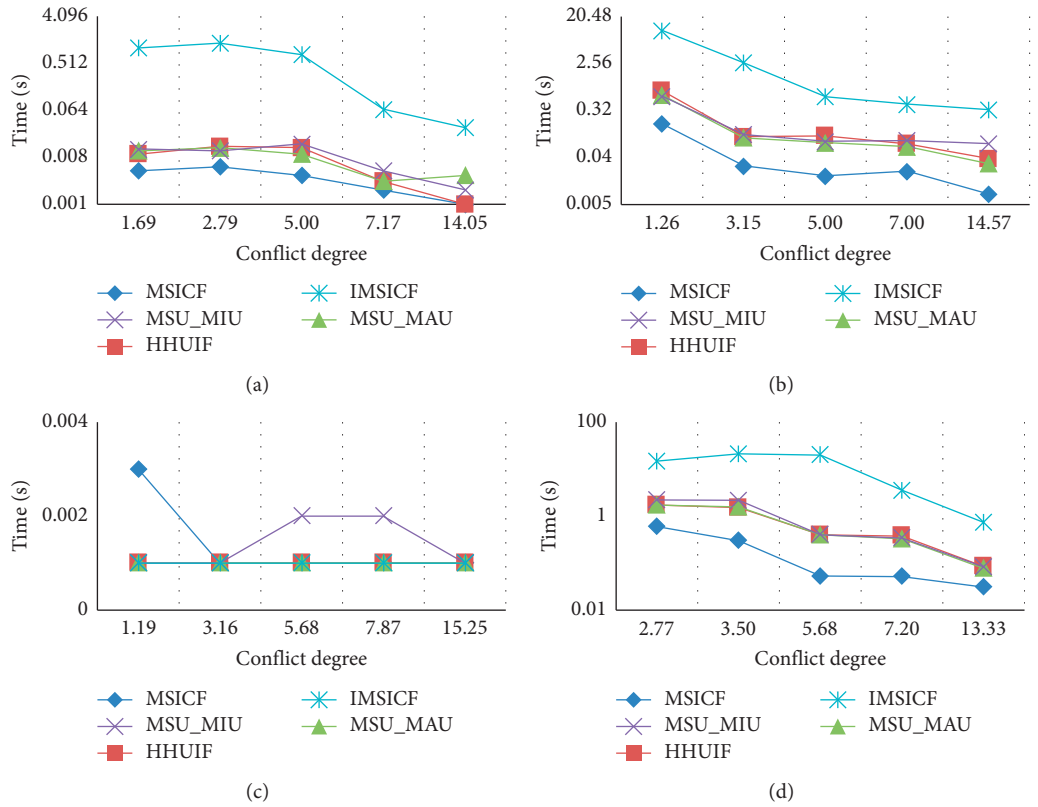

Figure 3: Execution times under various conflict degrees: (a) T25I10D10K; (b) T20I6D100K; (c) Foodmart; (d) mushroom.

transaction. However, IMSICF performs the best in Figure 3 because Foodmart is a very sparse dataset compared to the other datasets. Moreover, the sparse dataset indicates that the number of nonsensitive itemsets supported by each transaction is less. Thus, the execution time in Foodmart is less than that in other datasets.

Based on two-way ANOVA, there is a significant difference between the execution times of the sanitization algorithms for T25I10D10K and mushroom datasets ($P = 0.0037$ in Figure 2(a) and $P = 0.002$ in Figure 2(d)). For T20I6D100K and Foodmart datasets, there is no significant difference between the sanitization algorithms ($P = 0.13 > 0.05$ in Figure 2(b) and $P = 0.44$ in Figure 2(c)).

5.4. Missing Costs. The results of the missing costs under various sensitive percentages are shown in Figure 4. It can be observed that the missing costs are increased with the rise of the sensitive percentage due to the increasing number of modified transactions. From the results of Figure 4, it is also clear to see that the proposed algorithm prevents more nonsensitive itemsets from being overhidden. An important reason is that the transaction with the minimum number of nonsensitive itemsets and the maximum utility of sensitive itemset is chosen for modification. The second reason is that the conflict count of each item contained in sensitive itemsets is recalculated once a sensitive itemset is hidden. However, the algorithms MSU_MAU, MSU_MIU, HHUIF, and MSICF select the victim item based on the value of utility. Thus, the side effects on nonsensitive information are discarded in the sanitization process. Moreover, based on two-way ANOVA, there is a significant difference between the missing costs of the sanitization algorithms for various datasets ($P = 1.46 * 10^{-8}$ in Figure 2(a), $P = 9.43 * 10^{-6}$ in Figure 2(b), $P = 4.24 * 10^{-7}$ in Figure 2(c), and $P = 0.0006$ in Figure 2(d)).

The results of the missing costs under various conflict degrees are shown in Figure 5. From the results of Figure 5, it can be observed that the missing costs decrease as the conflict degree is increased. The reason is that the higher the conflict degree is, the more sensitive the itemsets will be hidden by modifying a victim item. Correspondingly, the number of nonsensitive itemsets concealed by mistake is decreased. From Figure 5, we can also find that the proposed algorithm outperforms other algorithms. This is caused by the sanitization strategy of IMSICF. Besides, it is noted that the number of missing nonsensitive itemsets in mushroom is more than that in other datasets. The reason is that mushroom is much denser compared to the other datasets, which indicates that the modification of the dataset will cause more side effects on nonsensitive itemsets. Moreover, based on two-way ANOVA, there is a significant difference between the missing costs of the sanitization algorithms for various datasets ($P = 0.004$ in Figure 2(a), $P = 0.021$ in Figure 2(b), $P = 2.04 * 10^{-7}$ in Figure 2(c), and $P = 0.02$ in Figure 2(d)).

5.5. Itemset Utility Similarity. The results of the itemset utility similarity under various sensitive percentages for different datasets are plotted in Figure 6. We can find that the IUS values are decreased with the growth of the sensitive percentage. This is reasonable because the increase on the number of sensitive itemsets will cause more transactions to be sanitized. Thus, the damage to the nonsensitive itemsets is correspondingly increased. From the results of Figure 6, we also observe that the proposed algorithm outperforms the other four algorithms in most cases. The reason is that IMSICF takes the side effects on nonsensitive knowledge into account when identifying the victim transaction. However, the other algorithms select the victim transaction based on the value of utility, without considering the damage to nonsensitive information by the sanitization process. Besides, it is interesting to see that the IUS values of mushroom are lower than those of the other datasets since mushroom is a very dense dataset.

Based on two-way ANOVA, there is a significant difference between the IUS values of the sanitization algorithms for various datasets ($P = 2.36 * 10^{-8}$ in Figure 2(a), $P = 0.0003$ in Figure 2(b), $P = 8.44 * 10^{-7}$ in Figure 2(c), and $P = 0.0009$ in Figure 2(d)).

The results of the itemset utility similarity under various conflict degrees for different datasets are plotted in Figure 7. It can be observed that the conflict degree has a great impact on the itemset utility similarity. With the growth of the correlation among the sensitive itemsets, more sensitive itemsets are hidden when a sensitive itemset is sanitized. Thus, less nonsensitive itemsets are concealed in error after the database sanitization, and the IUS values are correspondingly increased. From Figure 7, it is also clear to see that the proposed algorithm outperforms the other algorithms under various conflict degrees. The reason is that the impact on nonsensitive information is considered in the IMSICF algorithm. Moreover, the conflict count of each sensitive item is dynamically calculated. However, the other algorithms only take the concept of utility into account in the sanitization process.

Based on two-way ANOVA, there is a significant difference between the IUS of the sanitization algorithms for various datasets ($P = 0.0046$ in Figure 2(a), $P = 0.025$ in Figure 2(b), $P = 1.25 * 10^{-7}$ in Figure 2(c), and $P = 0.019$ in Figure 2(d)).

5.6. Database Utility Similarity. The results of the database utility similarity under various sensitive percentages are shown in Figure 8. It can be seen that the DUS values are decreased with the growth of the sensitive percentage because more items are modified for hiding more sensitive itemsets. As shown in Figure 8, we also observe that the IMSICF algorithm outperforms the other approaches except the MSU_MIU algorithm. The reason is that MSU_MIU identifies the victim transaction T_{vic} with the maximum utility of a sensitive itemset X, and the item contained in X with the minimum utility is selected to be a victim item I_{vic}. In addition, the utility of X is reduced by $u(X, T_{\text{vic}})$ when I_{vic} is removed from T_{vic}. Thus, the database utility similarity of MSU_MIU is higher than that of the other algorithms. However, the proposed algorithm IMSICF selects the

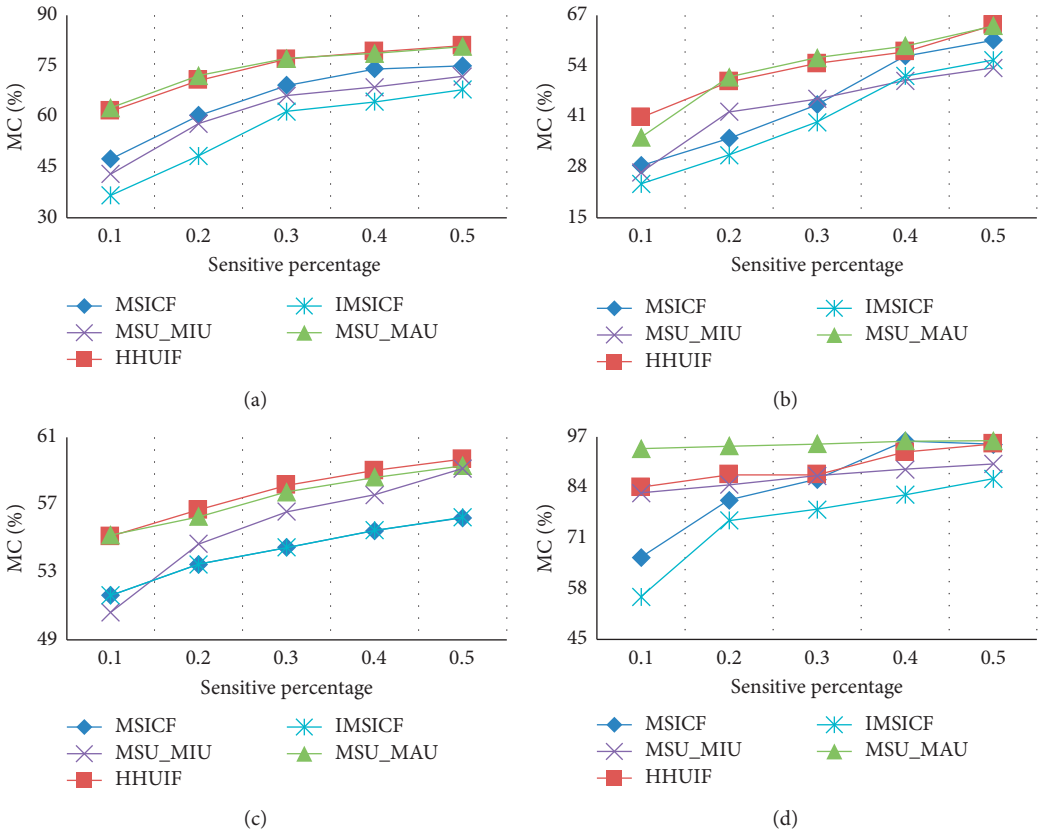

Figure 4: MC values under various sensitive percentages: (a) T25I10D10K; (b) T20I6D100K; (c) Foodmart; (d) mushroom.

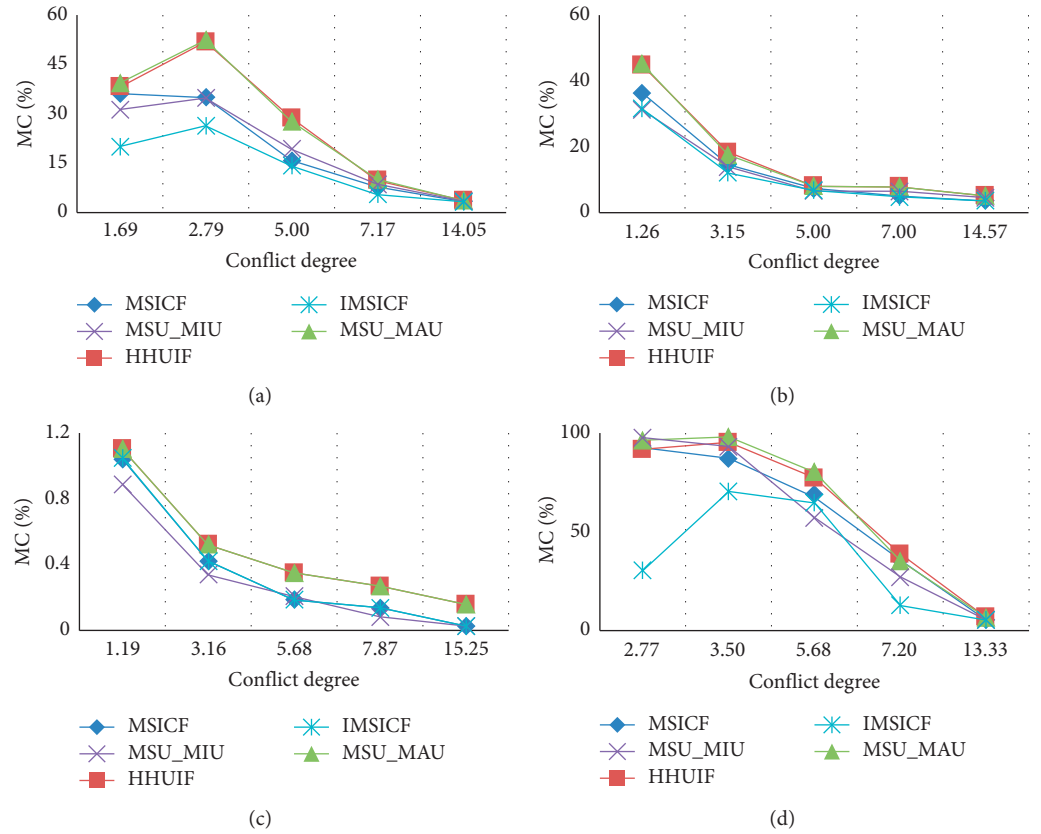

Figure 5: MC values under various conflict degrees: (a) T25I10D10K; (b) T20I6D100K; (c) Foodmart; (d) mushroom.

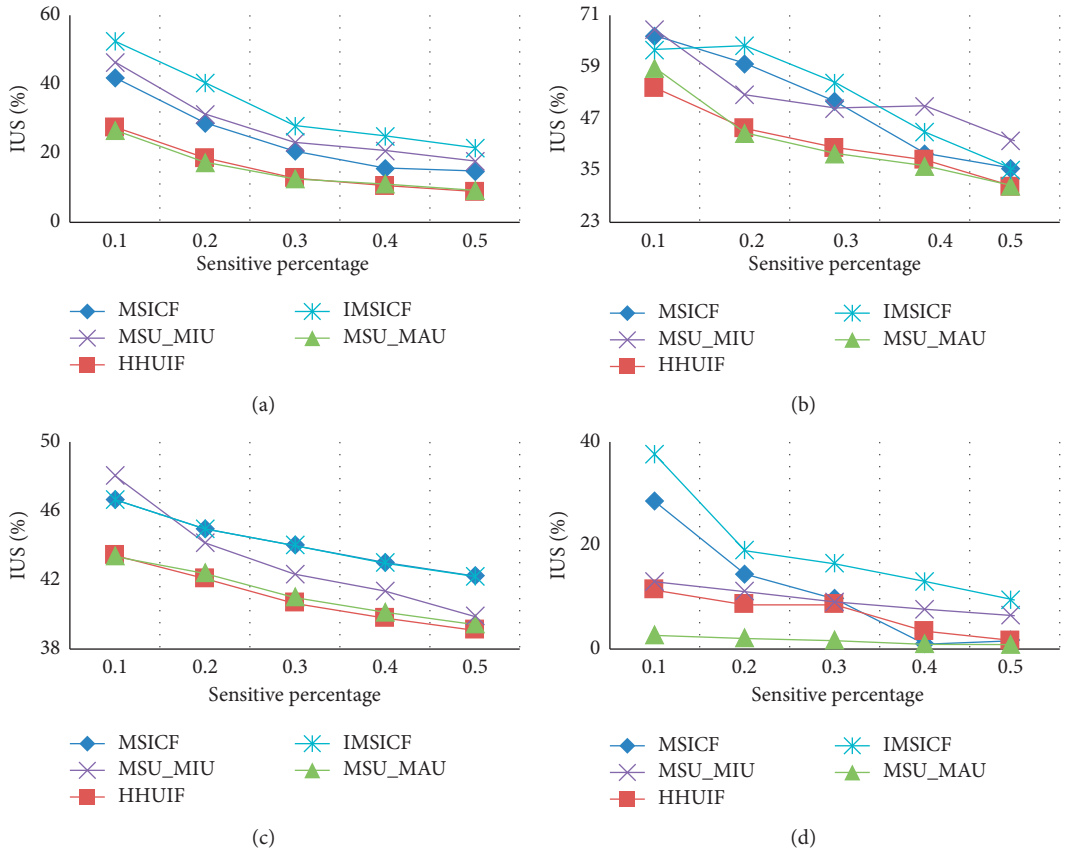

Figure 6: IUS values under various sensitive percentages: (a) T25I10D10K; (b) T20I6D100K; (c) Foodmart; (d) mushroom.

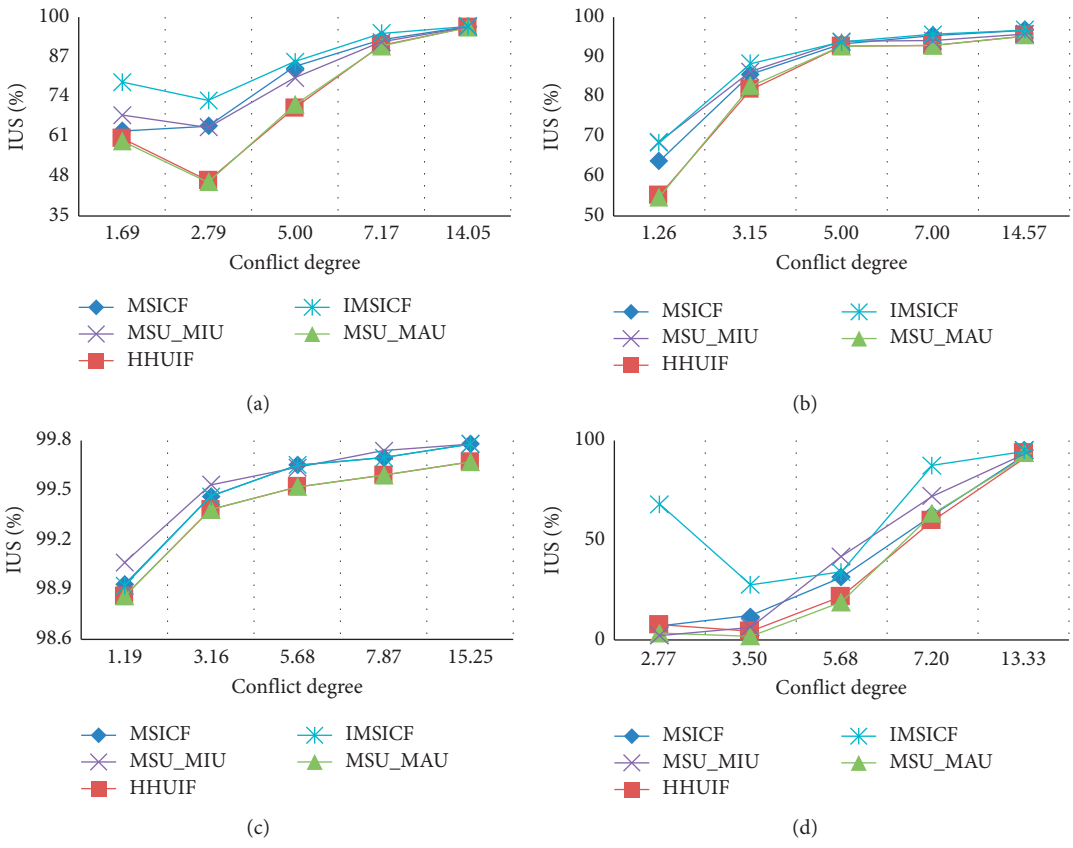

Figure 7: IUS values under various conflict degrees: (a) T25I10D10K; (b) T20I6D100K; (c) Foodmart; (d) mushroom.

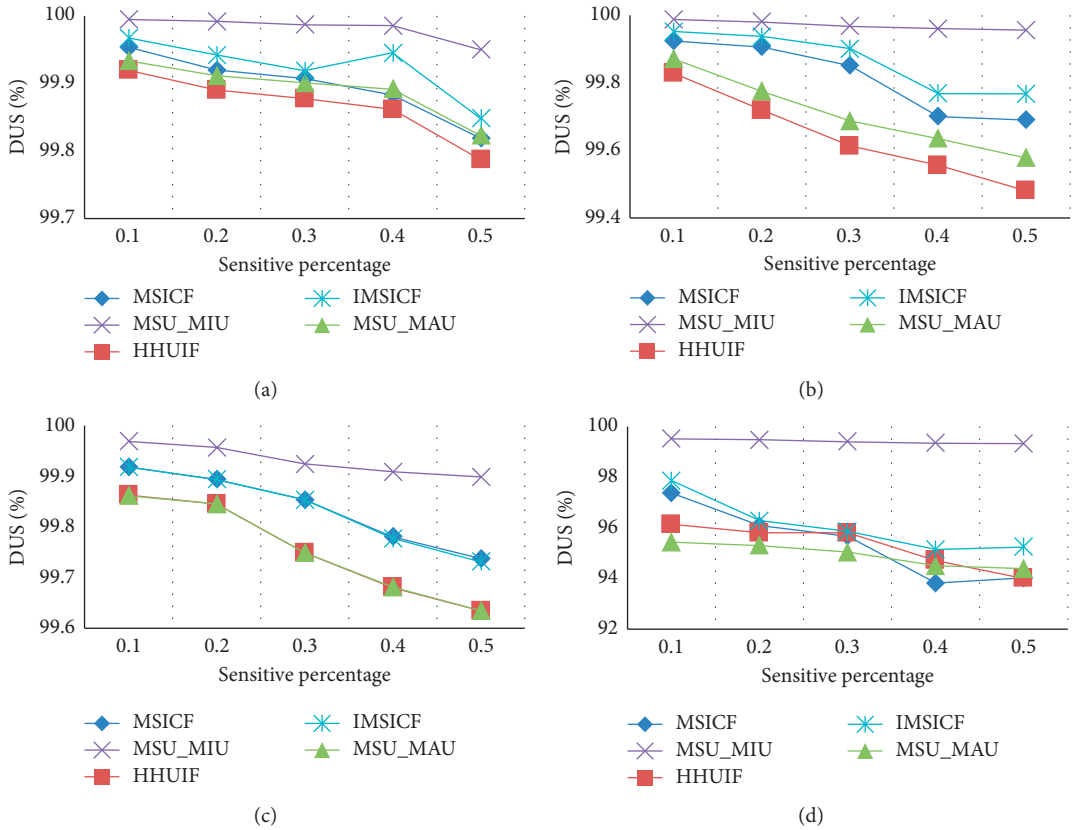

Figure 8: DUS values under various sensitive percentages: (a) T25I10D10K; (b) T20I6D100K; (c) Foodmart; (d) mushroom.

sanitized item based on the side effects on nonsensitive information. Thus, IMSICF performs worse than MSU_MIU in terms of DUS. Moreover, HHUIF, MSICF, and MSU_MAU algorithms select the victim item with the maximum utility. Hence, these algorithms perform worse than the previous two algorithms.

Based on two-way ANOVA, there is a significant difference between the DUS of the sanitization algorithms for various datasets ($P = 3.27 * 10^{-8}$ in Figure 2(a), $P = 2.55 * 10^{-7}$ in Figure 2(b), $P = 2.43 * 10^{-7}$ in Figure 2(c), and $P = 1.11 * 10^{-8}$ in Figure 2(d)).

The results of the database utility similarity under various conflict degrees for different datasets are shown in Figure 9. The DUS values are increased as the conflict degree increases. This is reasonable because few items are sanitized when the sensitive itemsets have more common items. In Figure 9, we also find that the MSU_MIU algorithm performs the best in terms of DUS under various conflict degrees. The reason is that the item with the minimal utility is chosen for modification. The proposed algorithm IMSICF has better performance than MSU_MAU, HHUIF, and MSICF because these algorithms identify the victim item with the maximum utility. Besides, note that DUS of mushroom is lower than that of other datasets. This is because the number of items supported by a transaction in mushroom is much higher compared to the other datasets. Moreover, based on two-way ANOVA, there is a significant difference between the DUS of the sanitization algorithms for various datasets ($P = 0.0016$ in Figure 2(a), $P = 0.039$ in Figure 2(b), $P = 1.13 * 10^{-6}$ in Figure 2(c), and $P = 0.0015$ in Figure 2(d)).

The above experimental results demonstrate that the proposed algorithm IMSICF outperforms the other state-of-the-art algorithms in terms of MC and IUS. The reason is that IMSICF selects the victim transaction based on the side effects on nonsensitive itemsets. Besides, we can find that the MSU_MIU algorithm performs better than other algorithms in DUS. This is reasonable because the victim item with the minimum utility is chosen for modification, and the utility of a sensitive itemset is reduced by the utility of the itemset in the identified transaction when a victim item is removed. In addition, it can be observed that the density of a dataset affects the performance of the itemset hiding.

6. Conclusions

In this paper, an improved sanitization algorithm called IMSICF is proposed for privacy-preserving utility mining. This algorithm identifies the victim item with the maximum conflict count, which is dynamically computed in the sanitization process. Then, the sensitive itemset containing the victim item is selected to be hidden. The transaction with the maximum utility of the currently hidden sensitive itemset and the minimum count of nonsensitive itemsets is chosen for modification. Hence, the side effects on nonsensitive knowledge are effectively reduced. In our experiments, real

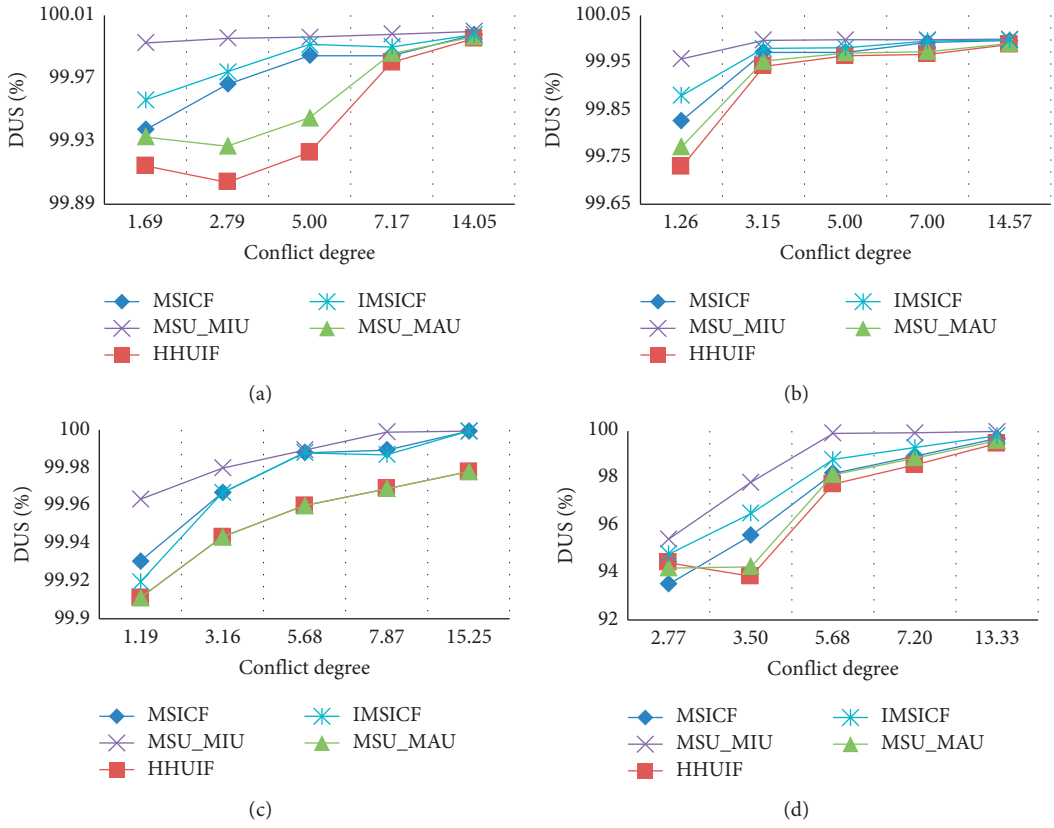

Figure 9: DUS values under various conflict degrees: (a) T25I10D10K; (b) T20I6D100K; (c) Foodmart; (d) mushroom.

and synthetic datasets are used to evaluate the performance of the proposed algorithm. The experimental results show that IMSICF outperforms the state-of-the-art algorithms in missing cost and itemset utility similarity, at the expense of a degradation on efficiency. Besides, it is observed that the conflict degree of sensitive itemsets has a great impact on the performance of the sanitization algorithms.

For future work, we will focus on preserving other forms of sensitive knowledge, such as frequent and utility itemset.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This research was supported by the National Natural Science Foundation of China (Grant no. 61802344), Zhejiang Provincial Natural Science Foundation of China (Grant no. LY16F030012), Ningbo Natural Science Foundation of China (Grant no. 2017A610118), General Scientific Research Projects of Zhejiang Education Department (Grant no. Y201534788), and Youth Foundation for Humanities and Social Sciences Research of Ministry of Education of China (Grant no. 16YJCZH112).

References

[1] P. Fournier-Viger, J. C. W. Lin, B. Vo et al., "A survey of itemset mining," *Wiley Interdisciplinary Reviews-Data Mining and Knowledge Discovery*, vol. 7, no. 4, p. e1207, 2017.

[2] W. Gan, J. C. W. Lin, P. Fournier-Viger et al., "A survey of incremental high-utility itemset mining," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 8, no. 2, p. e1242, 2018.

[3] J. Wang, F. Liu, and C. Jin, "PHUIMUS: a potential high utility itemsets mining algorithm based on stream data with uncertainty," *Mathematical Problems in Engineering*, vol. 2017, Article ID 8576829, 13 pages, 2017.

[4] R. Wang, Q. Sun, D. Ma, and Z. Liu, "The small-signal stability analysis of the droop-controlled converter in electromagnetic timescale," *IEEE Transactions on Sustainable Energy*, vol. 10, no. 3, pp. 1459–1469, 2019.

[5] D. E. O'Leary, "Knowledge discovery as a threat to database security," in *Proceedings of the 1st International Conference in Knowledge Discovery and Database*, pp. 507–516, Menlo Park, CA, USA, 1991.

[6] U. Yun, H. Ryang, G. Lee, and H. Fujita, "An efficient algorithm for mining high utility patterns from incremental databases with one database scan," *Knowledge-Based Systems*, vol. 124, pp. 188–206, 2017.

[7] J. Lee, U. Yun, G. Lee, and E. Yoon, "Efficient incremental high utility pattern mining based on pre-large concept,"

Engineering Applications of Artificial Intelligence, vol. 72, pp. 111–123, 2018.

[8] R. Agrawal and R. Srikant, "Privacy-preserving data mining," *ACM SIGMOD Record*, vol. 29, no. 2, pp. 439–450, 2000.

[9] R. Mendes and J. P. Vilela, "Privacy-preserving data mining: methods, metrics, and applications," *IEEE Access*, vol. 5, pp. 10562–10582, 2017.

[10] M. Atallah, E. Bertino, A. Elmagarmid et al., "Disclosure limitation of sensitive rules," in *Proceedings of the 1999 Workshop on Knowledge and Data Engineering Exchange*, pp. 45–52, Chicago, IL, USA, November 1999.

[11] E. Dasseni, V. S. Verykios, A. K. Elmagarmid, and E. Bertino, "Hiding association rules by using confidence and support," *Information Hiding*, Springer, Berlin, Germany, pp. 369–383, 2001.

[12] V. S. Verykios, A. K. Elmagarmid, E. Bertino, Y. Saygin, and E. Dasseni, "Association rule hiding," *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, no. 4, pp. 434–447, 2004.

[13] S. R. M. Oliveira and O. R. Zaïane, "Protecting sensitive knowledge by data sanitization," in *Proceedings of the 3rd International Conference on Data Mining*, pp. 613–616, Leipzig, Germany, July 2003.

[14] A. Amiri, "Dare to share: protecting sensitive knowledge with data sanitization," *Decision Support Systems*, vol. 43, no. 1, pp. 181–191, 2007.

[15] A. Gkoulalas-Divanis and V. S. Verykios, "Exact knowledge hiding through database extension," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 5, pp. 699–713, 2009.

[16] C.-M. Wu and Y.-F. Huang, "A cost-efficient and versatile sanitizing algorithm by using a greedy approach," *Soft Computing*, vol. 15, no. 5, pp. 939–952, 2011.

[17] T.-P. Hong, C.-W. Lin, K.-T. Yang, and S.-L. Wang, "Using TF-IDF to hide sensitive itemsets," *Applied Intelligence*, vol. 38, no. 4, pp. 502–510, 2013.

[18] H. Q. Le, S. Arch-int, and N. Arch-int, "Association rule hiding based on intersection lattice," *Mathematical Problems in Engineering*, vol. 2013, Article ID 210405, 11 pages, 2013.

[19] H. Q. Le, S. Arch-int, H. X. Nguyen, and N. Arch-int, "Association rule hiding in risk management for retail supply chain collaboration," *Computers in Industry*, vol. 64, no. 7, pp. 776–784, 2013.

[20] R. A. Shah and S. Asghar, "Privacy preserving in association rules using a genetic algorithm," *Turkish Journal of Electrical Engineering & Computer Sciences*, vol. 22, no. 2, pp. 434–450, 2014.

[21] P. Cheng and J.-S. Pan, "Use EMO to protect sensitive knowledge in association rule mining by adding items," in *Proceedings of the Companion Publication of the 2014 Annual Conference on Genetic and Evolutionary Computation*, pp. 65-66, Vancouver, Canada, June 2014.

[22] P. Cheng, J.-S. Pan, and C.-W. Lin, "Privacy preserving association rule mining using binary encoded NSGA-II," *Lecture Notes in Computer Science*, Springer, Berlin, Germany, pp. 87–99, 2014.

[23] P. Cheng, J.-S. Pan, and C.-W. L. Harbin, "Use EMO to protect sensitive knowledge in association rule mining by removing items," in *Proceedings of the 2014 Congress on Evolutionary Computation*, pp. 1108–1115, Beijing, China, June 2014.

[24] P. Cheng, C. W. Lin, and J. S. Pan, "Use HypE to hide association rules by adding items," *PLoS One*, vol. 10, no. 6, Article ID e0127834, 2015.

[25] P. Cheng, C.-W. Lin, J.-S. Pan, and I. Lee, "Manage the tradeoff in data sanitization," *IEICE Transactions on Information and Systems*, vol. E98.D, no. 10, pp. 1856–1860, 2015.

[26] P. Cheng, J. F. Roddick, S.-C. Chu, and C.-W. Lin, "Privacy preservation through a greedy, distortion-based rule-hiding method," *Applied Intelligence*, vol. 44, no. 2, pp. 295–306, 2016.

[27] J. C.-W. Lin, Y. Zhang, B. Zhang et al., "Hiding sensitive itemsets with multiple objective optimization," *Soft Computing*, vol. 23, no. 23, pp. 12779–12797, 2019.

[28] J.-S. Yeh and P.-C. Hsu, "HHUIF and MSICF: novel algorithms for privacy preserving utility mining," *Expert Systems with Applications*, vol. 37, no. 7, pp. 4779–4786, 2010.

[29] J. S. Yeh, P. C. Hsu, and M. H. Wen, "Novel algorithms for privacy preserving utility mining," in *Proceedings of the 8th International Conference on Intelligent Systems Design and Applications*, pp. 291–296, Kaohsiung, Taiwan, November 2008.

[30] R. R. Rajalaxmi and A. M. Natarajan, "A novel sanitization approach for privacy preserving utility itemset mining," *Computer and Information Science*, vol. 1, no. 3, pp. 77–82, 2009.

[31] C.-W. Lin, T.-P. Hong, H.-C. Hsu et al., "Reducing side effects of hiding sensitive itemsets in privacy preserving data mining," *The Scientific World Journal*, vol. 2014, Article ID 398269, 13 pages, 2014.

[32] U. Yun and J. Kim, "A fast perturbation algorithm using tree structure for privacy preserving utility mining," *Expert Systems with Applications*, vol. 42, no. 3, pp. 1149–1165, 2015.

[33] J. C.-W. Lin, T.-Y. Wu, P. Fournier-Viger, G. Lin, J. Zhan, and M. Voznak, "Fast algorithms for hiding sensitive high-utility itemsets in privacy-preserving utility mining," *Engineering Applications of Artificial Intelligence*, vol. 55, no. C, pp. 269–284, 2016.

[34] J. C.-W. Lin, T.-Y. Wu, P. Fournier-Viger, G. Lin, T.-P. Hong, and J.-S. Pan, "A sanitization approach of privacy preserving utility mining," *Advances in Intelligent Systems and Computing*, Springer, Berlin, Germany, pp. 47–57, 2016.

[35] J. C.-W. Lin, T.-P. Hong, P. Fournier-Viger, Q. Liu, J.-W. Wong, and J. Zhan, "Efficient hiding of confidential high-utility itemsets with minimal side effects," *Journal of Experimental & Theoretical Artificial Intelligence*, vol. 29, no. 6, pp. 1225–1245, 2017.

[36] S. Li, N. Mu, J. Le, and X. Liao, "A novel algorithm for privacy preserving utility mining based on integer linear programming," *Engineering Applications of Artificial Intelligence*, vol. 81, pp. 300–312, 2019.

[37] R. R. Rajalaxmi and A. M. Natarajan, "Effective sanitization approaches to hide sensitive utility and frequent itemsets," *Intelligent Data Analysis*, vol. 16, no. 6, pp. 933–951, 2012.

[38] X. Liu, F. Xu, and X. Lv, "A novel approach for hiding sensitive utility and frequent itemsets," *Intelligent Data Analysis*, vol. 22, no. 6, pp. 1259–1278, 2018.

[39] B. Le, D.-T. Dinh, V.-N. Huynh, Q.-M. Nguyen, and P. Fournier-Viger, "An efficient algorithm for hiding high utility sequential patterns," *International Journal of Approximate Reasoning*, vol. 95, pp. 77–92, 2018.

[40] L. T. T. Nguyen, P. Nguyen, T. D. D. Nguyen, B. Vo, P. Fournier-Viger, and V. S. Tseng, "Mining high-utility itemsets in dynamic profit databases," *Knowledge-Based Systems*, vol. 175, pp. 130–144, 2019.

[41] S. Krishnamoorthy, "HMiner: efficiently mining high utility itemsets," *Expert Systems with Applications*, vol. 90, pp. 168–183, 2017.

[42] A. Telikani and A. Shahbahrami, "Data sanitization in association rule mining: an analytical review," *Expert Systems with Applications*, vol. 96, pp. 406–426, 2018.

[43] U. Yun and D. Kim, "Analysis of privacy preserving approaches in high utility pattern mining," *Advances in Computer Science and Ubiquitous Computing*, Springer, Singapore, pp. 883–887, 2016.

[44] S. Zida, P. Fournier-Viger, J. C.-W. Lin, C.-W. Wu, and V. S. Tseng, "EFIM: a highly efficient algorithm for high-utility itemset mining," *Lecture Notes in Computer Science*, Springer, Cham, Switzerland, pp. 530–546, 2015.