An Improved Algorithm to Protect Sensitive High Utility Itemsets in Transaction Database

Nguyen Khac Chien[1(✉)] and Dang Thi Kim Trang[2]

[1] Faculty of Foreign Languages and Informatics, People's Police University, Ho Chi Minh City, Vietnam

[2] Posts and Telecommunications Institute of Technology Ho Chi Minh City, Ho Chi Minh City, Vietnam

Abstract. Privacy-Preserve Utility Mining is becoming a topic of interest to many researchers. The goal is to protect the sensitive-high utility itemsets in the transaction databases from being exploited by data mining techniques. This paper studies methods to hide sensitive high utility itemsets in transaction databases. There are some effective methods to deal with this problem, but these methods still cause undesirable side effects, such as: being missing hidden itemsets with non-sensitive high utility itemsets, the difference between the original database and the modified database, etc. This paper proposed an improved algorithm for hiding sensitive high utility itemsets, called IEHSHUI, focus on choosing the order to hide sensitive itemsets and selecting items to modify with minimal side effects. Experimental results show that the IEHSHUI proposed algorithm is more efficient than existing algorithms in terms of execution time.

Keywords: High utility itemset · Hiding utility itemset · Privacy-preserving utility mining

1 Introduction

Data mining is used to discover knowledge and decision-making information from huge databases [3]. In the business, data is shared between different companies for mutual benefit. However, this carries the risk of exposing sensitive information contained in that databases [10]. Sensitive information can be represented as a set of frequency patterns and high utility patterns with confidentiality. Therefore, the data owner wants to hide this sensitive information before sharing the database with a partner. To solve this problem, privacy-preserving data mining has been proposed and has become an important research direction [1]. Privacy-preserving data mining methods have been applied in various fields, such as in cloud computing, e-health, wireless sensor networks, and location services [9].

The method to hide sensitive information is to modify the original database into a modified one by modifying some items in the original database. Atallah et al. [2] have demonstrated that the problem of hiding sensitive information in the optimal problem is an NP-hard problem and the authors have proposed a data modification algorithm based on a heuristic strategy. There have much research works in this field. However,

P. Cong Vinh and N. Huu Nhan (Eds.): ICTCC 2021, LNICST 408, pp. 95–107, 2021.
https://doi.org/10.1007/978-3-030-92942-8_9

there is currently very little research on the method of hiding sensitive information to protect high utility itemsets in transaction databases based on the utility concept. Besides, the side effects base on non-sensitive information is also of little concern. Therefore, the database modification can protect non-sensitive information and the quality of the original database has been noticed.

Some works have been published to solve this problem such as works [4, 7, 8, 13–15]. Most of these algorithms use data modification methods to efficiently hide sensitive itemsets. However, they still cause unwanted side effects such as hiding non-sensitive information, database corruption before and after the hiding process. This paper will propose a strategy to improve the performance of the EHSHUI algorithm in [4], called the IEHSHUI algorithm. The proposed algorithm focuses on:

(i) Choose sensitive itemset S_j which has the maximal utility to hide.
(ii) Select item $i \in S_j$, which is among the most sensitive itemsets to modify. If many items that satisfy, then choose the item from the least number of non-sensitive itemsets to modify, called victim item i_{vic}.
(iii) The IEHSHUI proposed algorithm will use the coefficient α in [14] to calculate the ratio of the number of item i_{vic} in all sensitive transactions ST supporting the sensitive itemset $S_j \in ST$ to reduce processing time.

The rest of the paper organized as follows. Section 2 basic concepts and reviews related works. Section 3 proposes an improved hiding high utility item set algorithm. Section 4 experimental results and analysis. Finally, we present the conclusion of the paper in Sect. 5.

2 Basic Concepts and Related Work

2.1 Basic Concepts

In this paper, we use Table 1 and Table 2 as illustrative examples. In Table 1, the database contains nine transactions. In Table 2, the profit table of each item is contained in the transaction database.

Some concepts of high utility itemsets mining are presented in [5].

Let $I = \{i_1, i_2, \ldots, i_m\}$ be a distinct set of m items, where each item $i_p \in I$ has a profit, called $eu(i_p), 1 \le p \le m$ and $D = \{T_1, T_2, \ldots, T_n\}$ represents a transaction database, where T_i is a subset of items contained in I. Each transaction is assigned a unique identifier *TID*. A set containing one or more items is called an itemset. A transaction T supports an itemset X if $X \subseteq I$. An itemset $X = \{i_1, i_2, \ldots, i_k\}$ contain k items is called a k-itemset. Each item i_p in the transaction T_q is associated with a number of item i_p in the transaction T_q, called $iu(i_p, T_q)$.

Table 1 contains a transaction database of a store which have six items *a, b, c, d, e, f*. There are nine transactions $T_1, T_2, \ldots, T_9$ in this transaction database. Each transaction contains the sold items along with their quantity. For example, T_1 transaction have three items a, b, and e have been sold with corresponding quantities $iu(a) = 10$, $iu(b) = 2$ and $iu(e) = 5$.

Definition 1. Profit of an item i_p represents the importance of an item i_p, denoted by $eu(i_p)$. For example, in Table 2, $eu(b) = 2$ and $eu(d) = 1$.

Definition 2. The utility of an item i_p in a transaction T_q, denoted by $u(i_p, , T_q)$, is calculated as follows:

$$\boldsymbol{u(i_p, T_q) = iu(i_p, T_q) * eu(i_p)} \tag{1}$$

For example, $u(b,\ T_8) = iu(b,\ T_8) * eu(b) = 15 * 2 = 30$.

Definition 3. The utility of an itemset X in a transaction T_q, denoted by $u(X, , T_q)$, is calculated as follows:

$$\boldsymbol{u(X, T_q) = \sum_{i_p \in X} u(i_p, T_q)} \tag{2}$$

For example, $u(bd, T_8) = u(b, T_8) + u(d, T_8) = 15 * 2 + 35 * 1 = 65$.

Table 1. Transaction database

TID	Transactions
T1	(a,10),(b,2),(e,5)
T2	(c,4), (d,2), (e,7), (f,15)
T3	(b,15), (c,15), (e,1), (f,1)
T4	(a,5), (b,4), (c,20), (d,2), (e,5)
T5	(b,25), (c,15)
T6	(a,15), (e,7), (f,15)
T7	(a,25), (c,15), (d,40)
T8	(b,15), (d,35), (e,3)
T9	(a,5),(b,10),(c,20),(d,30),(e,2),(f,3)

Table 2. Profit table

Item	a	b	c	d	e	f
Profit	7	2	1	1	5	10

Table 3. High utility itemsets (HUI) *minutil = 250*

HID	Itemset	Utility
1	ef	425
2	a	422
3	acd	372
4	aef	367
5	ae	342
6	f	340
7	af	322
8	ad	317
9	ac	300
10	cdef	281
11	cef	279
12	def	257

Definition 4. The utility of an itemset X, denoted by $u(X)$, is calculated as follows:

$$\boldsymbol{u(X) = \sum_{X \subseteq T_q \wedge T_q \in D} u(X, T_q)} \tag{3}$$

For example, $u(bd) = u(bd, T_4) + u(bd, T_8) + u(bd, T_9) = 10 + 65 + 50 = 125$.

Definition 5. The utility of a transaction T_q, denoted by $tu(T_q)$, is calculated as follows:

$$\boldsymbol{tu}(\boldsymbol{T_q}) = \sum_{\boldsymbol{i_p} \in \boldsymbol{T_q}} \boldsymbol{u}(\boldsymbol{i_p}, \boldsymbol{T_q}) \tag{4}$$

For example, $tu(T_8) = u(b, T_8) + u(d, T_8) + u(e, T_8) = 30 + 35 + 15 = 80$.

Definition 6. The problem of high utility itemset mining. An itemset X is said to be a high utility itemset if the utility of X is greater than or equal to the minimum utility threshold specified by the user, denoted *minutil*. Let *HUI* the set of high utility itemsets, we have $HUI = \{X | X \subseteq I \wedge u(X) \geq minutil\}$.

The transaction database is given in Tables 1, Table 2, and the minimum utility threshold $minutil = 250$, the set of high utility itemsets (HUI) are shown in Table 3.

2.2 Research Problem

The research problem is stated as follows:

Given a set of sensitive high utility itemsets (referred to as sensitive itemsets) (*SHUI*) must be hidden, denoted by $SHUI = \{S_1, S_2, \ldots, S_s\}$, where $S_j \in HUI$, $1 \leq j \leq s$. The problem of hiding sensitive itemsets is to modify the original database D to a modified database D' such that the utility of all sensitive itemsets must be less than the minimum utility threshold specified by the user, i.e. $u(S_j) < minutil, vo'ij \in [1, s]$.

Definition 7. The set of sensitive itemsets, where si is a sensitive itemset that needs to be hidden before the database is released, we have $SHUI \subseteq HUI$. Let *NSHUI* a set of non-sensitive high utility itemsets (referred to as non-sensitive itemset), we have $SHUI \cup NSHUI = HUI$.

Definition 8. The set of sensitive transactions, each of which contains at least one sensitive itemset, is denoted by *ST*.

The item that is modified to hide the sensitive itemset is called the victim item (i_{vic}). The transaction containing the victim item is called the victim transaction (T_{vic}).

The data modification process of sensitive itemsets hiding consists of the following three steps:

(i) Apply high utility mining algorithms on transaction database D to get all high utility itemsets (*HUIs*);
(ii) Define the set of sensitive itemsets *SHUI* based on user requirements;
(iii) Apply algorithm to hide sensitive itemsets to generate modified database D'.

2.3 Related Work

In recent years, privacy-preserving utility mining methods have attracted the attention of many researchers. This problem becomes important because it considers both the quantity and profit of each item in the transaction database to hide sensitive high utility

itemsets. For privacy-preserving high utility mining to hide sensitive information (sensitive high utility itemsets) in the database, while ensuring other important information is still provided to the adversary, this problem is considered an optimization problem. Finding transactions and items to modify while optimally hiding sensitive high utility information is a difficult and impossible problem [2]. In 2010, Yeh et al. [15] were the first to propose two heuristic algorithms HHUIF and MSICF to hide sensitive high utility itemsets. The two algorithms select the item with the highest utility to modify for the hiding process. The HHUIF algorithm discards the items with the highest utility. The MSICF algorithm considers the number of conflicts in the hidden process.

Then, some works also proposed algorithms to improve the above two algorithms, such as Vo et al. (2013) [14] proposed an algorithm to improve the HHUIF algorithm in terms of time. Selvaraj et al. (2013) [13] propose an algorithm that improves MHIS in selecting the item in case their utility is the same. The results show that the MHIS algorithm is better than the HHUIF algorithm in terms of HF (hiding failure) and MC (missing cost) side effects. Yun and Kim (2015) [16] propose the FPUTT algorithm to improve the efficiency of the HHUIF algorithm by using a tree structure. The FPUTT algorithm is about 5 to 10 times faster than HHUIF. However, the side effects cause the same as HHUIF. Lin et al. (2015) [6] propose three similar measurements to use as a new standard for assessing side effects in privacy-preserving utility mining. Lin et al. (2016) [7] proposed two algorithms MSU-MAU and MSU-MUI to protect high utility itemsets. Both of these algorithms choose the transactions containing the maximal utility of sensitive itemsets to be hidden. These two algorithms apply the Max-Min property of utility to reduce side effects and increase the speed of data modification compared to HHUIF and MSICF algorithms. Furthermore, the MSU-MIU algorithm is better than the MSU-MAU algorithm due to using the optimal projection in the MSU-MIU. Xuan Liu et al. (2020) [8] propose three heuristic algorithms SMAU, SMIU, and SMSE to hide sensitive itemsets in transaction databases. Transactions that support the smallest number of non-sensitive itemsets are selected as modification transactions. These algorithms use two table structures T-table and HUI-table to reduce the number of database scans. However, these algorithms still cause unwanted side effects during hiding the sensitive high utility itemset.

Trieu et al. (2020) [4] proposed an EHSHUI algorithm to improve the HHUIF algorithm. EHSHUI select victim transaction which supports the sensitive itemset has the maximal utility, and the selection victim item that is in the sensitive itemset to be hidden has little effect on the number of non-sensitive itemsets being hidden by mistake or the item with the least utility. The results show that this algorithm is more efficient than HHUIF and MSICF in terms of side effects and execution time. However, the EHSHUI algorithm in [4] still scans the database many times and takes a long time to select the victim item (Algorithm 2 in [4]).

Most of the above algorithms hide all sensitive itemsets but still cause unwanted side effects. Some algorithms may have to scan the database many times leading to a lot of execution time.

In this paper, an effective strategy of selecting the modified item and modified transaction will be proposed so that all sensitive itemsets can be hidden while minimal side effects on non-sensitive information and can reduce processing time.

3 Proposed Algorithm

3.1 Proposal Basics

The goal of hiding sensitive itemsets to protect privacy is not only to hide all sensitive itemsets but also to minimize side effects on non-sensitive information and the integrity of the original database. The algorithm proposed in this paper is also aimed at this goal.

The method of hiding sensitive itemsets is to modify the original database by deleting or reducing the number of some items so that the utility of the sensitive itemsets falls below the minimum utility threshold.

Most published works focus on identifying: Which transaction is selected to be modified (victim transaction - T_{vic})? and which item is selected (victim item – i_{vic}) for modification in the victim transaction (T_{vic})?

In this paper, we focus on: Firstly, when hiding sensitive itemsets, the order in which to choose which sensitive itemsets to hide first affects the hidden process and cause unwanted side effects. In this paper, we propose to choose which sensitive itemset has the maximal utility to be hidden first. Because when hiding these sensitive itemsets, it is possible to hide other sensitive itemsets, then we don't need to hide that sensitive itemset anymore. This is demonstrated in the illustrative example. Therefore, it is possible to increase the efficiency of the hiding process. Secondly, we select the victim item (i_{vic}) that is among the most sensitive itemsets to modify. If there are many items satisfied, we select the item from the least number of non-sensitive itemsets to modify. This minimizes the side effects of non-sensitive information. Thirdly, in most of the published algorithms like [4, 7, 8, 11–13, 15], they modify each transaction one by one. This may increase processing time. In this paper, we use the coefficient α in [14] to calculate the quantity reduction rate of item i_{vic} in all sensitive transactions that support Si need to hide. Then the proposed algorithm will modify all sensitive transactions at the same time. This reduces the number of database scans as well as the time it takes to hide the sensitive itemset.

Let S_j be the sensitive high utility itemset. To hide S_j, the utility of S_j must be reduced by at least:

$$\boldsymbol{diffu} = \boldsymbol{u}\left(\boldsymbol{S_j}\right) - \boldsymbol{minutil} + 1 \tag{5}$$

where $u\left(S_j\right)$ is the utility of sensitive itemset S_j and *minutil* is the minimal utility threshold.

The coefficient α is calculated as follows:

$$\boldsymbol{\alpha} = \boldsymbol{diu} \times \frac{\boldsymbol{eu}\left(\boldsymbol{i_p}\right)}{\boldsymbol{sum}\left(\boldsymbol{i_p}\right)} \tag{6}$$

where $diu = \left\lceil \frac{diffu}{eu(i_p)} \right\rceil$, $sum(i_p)$ is the total utility of item i_p all sensitive transactions supporting S_j.

Definition 9. Identify the victim item (i_{vic}): The item that is among the most sensitive itemsets. If there are many items satisfied, we select the item from the least number of non-sensitive itemsets.

The proposed algorithm is presented in Table 4. The algorithm is implemented as follows: First, S_j sensitive itemsets are sorted in descending order of its utility (line 1). Next, perform an iteration to hide the sensitive itemsets from lines 2–13. Calculate the utility to be reduced for each sensitive itemset (line 3). While $diffu > 0$, find the set of sensitive transactions ST. Identify the item that needs to be modified Ivic according to Definition 9 (line 6). Calculate the rate of reduction of the number of item i_{vic}: α (lines 7–8). Lines 9–12 proceed to modify the number of victim item Ivic in sensitive transactions ST. If $\alpha < 1$, the quantity of victim item i_{vic} in each ST transaction will be reduced by a ratio of α. If $\alpha \geq 1$, it will reduce the number of Ivic to 1, to avoid too much deviation in database structure after modification. Perform until all sensitive itemsets are hidden, then the algorithm ends. The proposed algorithm ensures that all sensitive itemsets are hidden.

3.2 Illustration Example

With the database given in Tables 1 and 2, the minimum utility threshold: *minutil* = *250*, we can exploit the set of high utility itemsets HUI presented in Table 3.

Suppose the set of sensitive itemsets to be hidden is $SHUI = \{\langle ae\rangle, \langle ef\rangle, \langle aef\rangle\}$

Line 1: Sort $SHUI$ in decreasing order of $u(S_j)$:

$$SHUI = \{ef(425), aef(367), ae(342)\}$$

Line 2: Select $S1 = \langle ef\rangle$ to be hidden

Line 3: Calculate $diffu = u(ef) - minutil + 1 = 425 - 250 + 1 = 176$

Line 4: Find the set of sensitive transactions that support $S1 = \langle ef\rangle$ as:

$$ST = \{T2, T3, T6, T9\}$$

Line 5: diff > 0

Table 4. Improvement EHSHUI algorithm (Called IEHSHUI)

Input: Original database D; High utility itemset HUI;
Sensitive high utility itemsets $SHUI = \{S_1, S_2, \ldots, S_s\}$;
Minimal utility threshold $minutil$:
Output: Sanitized database D'.

1	Sort $SHUI$ in decreasing order of $u(S_i)$;
2	**for each** ($S_j \in SHUI$) **do**
3	$diffu = u(S_j) - minutil + 1$.
4	Find set of sensitive transaction ST support S_i.
5	**while** ($diffu > 0$) **do**
6	Find i_{vic} by *Definition 9.*
7	$diu = \left\lceil \frac{diffu}{eu(i_{vic})} \right\rceil$ //Number of item i_{vic} can be reduced to hide sensitive itemset $S_j \in SHUI$
8	Calculate factor $\alpha = diu \times \frac{eu(i_{vic})}{sum(i_{vic})}$, where $sum(i_{vic}) = \sum_{T_q \in ST} u(i_{vic}, T_q)$
9	**for each** $T_q \in ST$ **do**
10	Modify the quantity of i_{vic}.
11	$iu(i_{vic}) = \begin{cases} iu(i_{vic}) - iu(i_{vic}) \times \alpha & if\ \alpha < 1 \\ 1 & if\ \alpha \geq 1 \end{cases}$
12	Modify $diffu$.
13	Update (D);

Line 6: Find the victim item i_{vic} to modify: There are 2 items e and f. Item e is present in the sensitive itemsets $\langle ae \rangle$, $\langle ef \rangle$, and $\langle aef \rangle$. Item f is present in $\langle ef \rangle$ and $\langle aef \rangle$ sensitive itemsets. So choose item e to modify because it is among the most number of sensitive itemsets.

Line 7: Calculate the items e that must be reduced to hide the sensitive itemset $S1 = \langle ef \rangle$ as : $diu = \left\lceil \frac{diffu}{eu(E)} \right\rceil = \left\lceil \frac{176}{5} \right\rceil = 36$

Line 8: Calculate the coefficient α for item e: $sum(e) = u(e, T2) + u(e, T3) + u(e, T6) + u(e, T9) = 35 + 5 + 35 + 10 = 85$

$$\alpha = diu \times \frac{eu(f)}{sum(f)} = 12 \times \frac{10}{340} = 0.35$$

Line 11: Since $\alpha > 1$, the IEHSHUI algorithm modifies the number of item e in transactions T2,T3,T6,T9 to the value 1.

Line 12: And update the values: The utility of the sensitive itemset $S1 = \langle ef \rangle$, reduced to: $u(\langle ef \rangle) = 425 - 65 = 360$.

$$diffu = 360 - 250 + 1 = 111$$

Line 13: Update the database.

Since $diffu > 0$ keep going back to lines 5, 6. IEHSHUI Algorithm selects item f to modify.

Line 7: Calculate the number of items f that need to be reduced to hide the sensitive itemset $S1 = \langle ef \rangle$ as:

$$diu = \left\lceil \frac{diffu}{eu(F)} \right\rceil = \left\lceil \frac{111}{10} \right\rceil = 12$$

Line 8: Calculate the coefficient α for item f.

$$sum\ (f) = u(f, T2)\ + u(f, T3)\ +\ u(f,\ T6) + u(f,\ T9)\ =\ 150\ +\ 10\ +\ 150\ +\ 30\ =\ 340$$

$$\alpha = diu \times \frac{eu(f)}{sum(f)} = 12 \times \frac{10}{340} = 0.35$$

So $\alpha = 0.35 < 1$. Calculate the number of items f that must be reduced in transactions $T2, T3, T6, T9$ as:

The item f that must be reduced in $T2$ is $15 * 0.35 = 5$. The number of item f that must be reduced in $T6$ is $15 * 0.35 = 5$. The item f that must be reduced in $T9$ is $3 * 0.35 = 1$. The number of item f that must be reduced in $T3$ is $12 - 5 - 5 - 1 = 1$. But the number of item f in $T3$ is only 1, so when reducing one item f from transaction $T3$, it is considered to remove item f from transaction $T3$. As a result, $T3$ will no support the sensitive itemset $\langle ef \rangle$. The utility of itemset $\langle ef \rangle$ must be reduced by $u(\langle ef \rangle,\ T3)$ when f is removed from transaction $T3$.

Update the values again: $u(<ef>) = 360 - 5 * 10 - 5 * 10 - 1 * 10 - u(<ef>.T3) = 360 - 110 - 15 = 135 < minutil = 250$. So the item set $S1 = <ef>$ has been hidden successfully.

$diff = -114 < 0$.

Line 13: Update the database.

Do the same to hide sensitive itemsets $S2 = <ae>$ and $S3 = <aef>$. Finally, the IEHSHUI proposed algorithm hides all sensitive itemsets and mistakenly hides five non-sensitive itemsets, which are $\langle f \rangle, \langle af \rangle, \langle cdef \rangle, \langle cef \rangle$ *and* $\langle def \rangle$.

Thus, in the IEHSHUI proposed algorithm, it is possible to modify many transactions at the same time and quickly hide the sensitive itemset. In part 4, the experiment will compare and evaluate the IEHSHUI proposed algorithm with the EHSHUI algorithm in [4].

4 Experimental Results

The experiments were performed on an Intel ® Core™ i7 computer with 2.00 GHz CPU, 8 GB RAM running on Windows 10. Algorithms are implemented in Java language. The experimental database obtained on the website http://www.philippe-fournier-viger.com/spmf/index.php?link=datasets.php has the following characteristics in Table 5:

Table 5. Experimental datasets

Dataset	Number of transaction	Number of item
Chess	3196	75
Mushroom	8124	120

We add the number of item in each transaction at random the values in the range using uniform distribution and the profit values of each item in the database are also generated randomly.

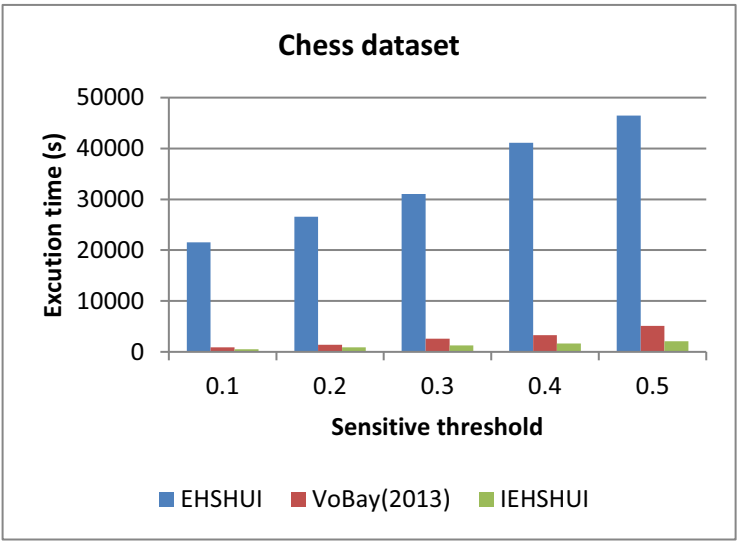

Fig. 1. Comparison of execution time on Chess dataset

In this paper, we compare the IEHSHUI proposed algorithm with the EHSHUI algorithms [4] and the algorithm (VoBay2013) [14] in terms of execution time and memory usage. The experiment was performed fifty times, then we have taken the average value. Experiment on the number of randomly selected sensitive itemsets, 0.1, 0.2, 0.3, 0.4 and 0.5 on the number of high utility itemsets (HUI) respectively. The results are shown in Fig. 1 and Fig. 2.

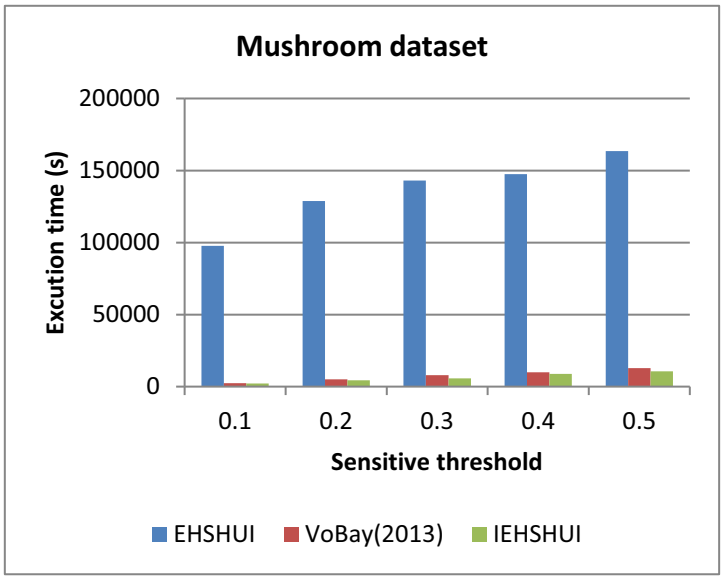

Fig. 2. Comparison of execution time on Mushroom dataset

Figure 1 and Fig. 2 show that the IEHSHUI proposed algorithm is the most efficient in terms of execution time on both Chess and Mushroom dataset. The IEHSHUI algorithm is faster than the EHSHUI algorithm in [4] many times because the IEHSHUI algorithm modifies multiple transactions at the same time to hide sensitive

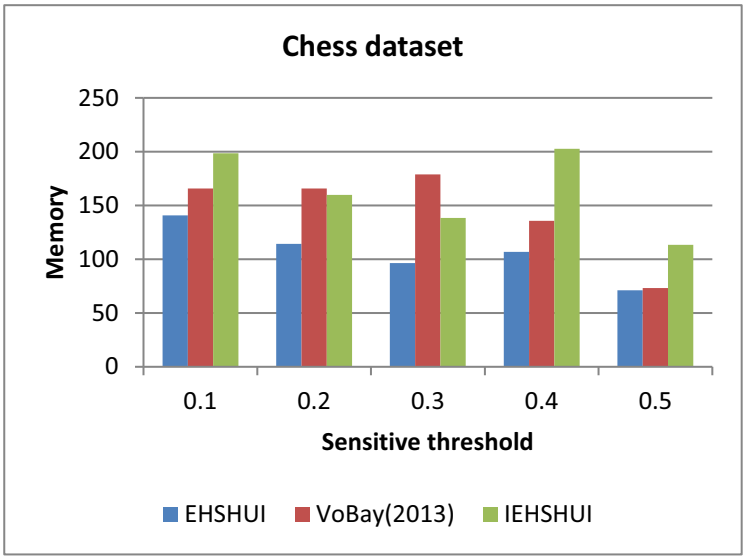

Fig. 3. Comparison of memory usage on Chess dataset

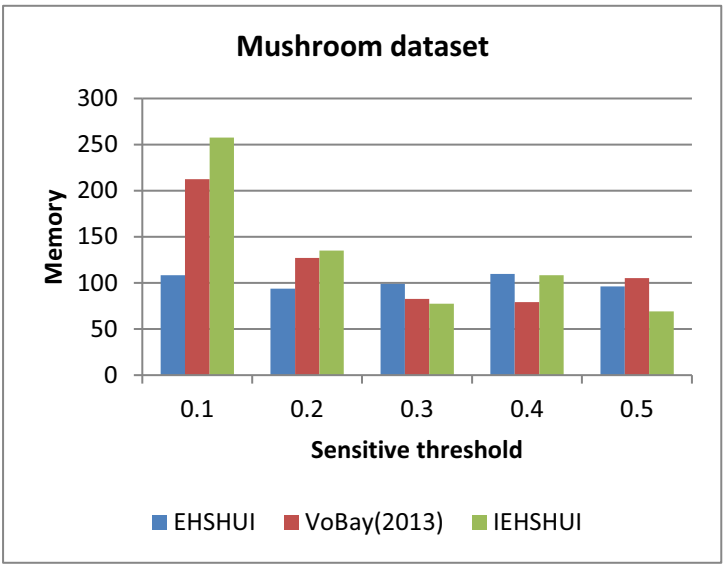

Fig. 4. Comparison of memory usage on Mushroom dataset

information. The EHSHUI algorithm in [4] modifies each transaction one by one. Figure 3 and Fig. 4 show that the memory usage of the IEHSHUI proposed algorithm is more than other algorithms.

5 Conclusion and Future Works

This paper has proposed an algorithm IEHSHUI to protect sensitive itemsets effectively based on the strategy of selecting reasonable sensitive itemsets and victim items. Experiment results show that the IEHSHUI algorithm is more efficient than the EHSHUI [4] and algorithm [14] in terms of execution time.

In the future, we continue to improve the algorithm and test the proposed algorithm on other transaction databases and compare with other algorithms to evaluate the effectiveness and performance on other measurements.

References

1. Agrawal, R., Srikant, R.: Privacy-preserving data mining. In: Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data (2000)
2. Atallah, M., et al.: Disclosure limitation of sensitive rules. In: Proceedings 1999 Workshop on Knowledge and Data Engineering Exchange (KDEX1999) (Cat. No. PR00453). IEEE (1999)
3. Fournier-Viger, P., et al.: A survey of itemset mining. Wiley Interdiscipl. Rev. Data Min. Knowl. Discov. **7**(4), e1207 (2017)

4. Huynh Trieu, V., Le Quoc, H., Truong Ngoc, C.: An efficient algorithm for hiding sensitive-high utility itemsets. Intell. Data Anal. **24**(4), 831–845 (2020)
5. Krishnamoorthy, S.: Pruning strategies for mining high utility itemsets. Expert Syst. Appl. **42**(5), 2371–2381 (2015)
6. Lin, C.-W., et al.: A GA-based approach to hide sensitive high utility itemsets. Sci. World J. **2014** (2014)
7. Lin, J.C.-W., et al.: Fast algorithms for hiding sensitive high-utility itemsets in privacy-preserving utility mining. Eng. Appl. Artif. Intell. **55**, 269–284 (2016)
8. Liu, X., Wen, S., Zuo, W.: Effective sanitization approaches to protect sensitive knowledge in high-utility itemset mining. Appl. Intell. **50**(1), 169–191 (2019). https://doi.org/10.1007/s10489-019-01524-2
9. Mendes, R., Vilela, J.P.: Privacy-preserving data mining: methods, metrics, and applications. IEEE Access **5**, 10562–10582 (2017)
10. O'Leary, D.E.: Knowledge discovery as a threat to database security. Knowl. Discov. Database **9**, 507–516 (1991)
11. Rajalaxmi, R., Natarajan, A.: Effective sanitization approaches to hide sensitive utility and frequent itemsets. Intell. Data Anal. **16**(6), 933–951 (2012)
12. Saravanabhavan, C., Parvathi, R.: Privacy preserving sensitive utility pattern mining. J. Theor. Appl. Inf. Technol. **49**(2) (2013)
13. Selvaraj, R., Kuthadi, V.M.: A modified hiding high utility item first algorithm (HHUIF) with item selector (MHIS) for hiding sensitive itemsets. Int. J. Innov. Comput. Inf. Contrl. **9**, 4851–4862 (2013)
14. Vo, B., et al.: An efficient method for hiding high utility itemsets. In: KES-AMSTA (2013)
15. Yeh, J.-S., Hsu, P.-C.: HHUIF and MSICF: Novel algorithms for privacy preserving utility mining. Expert Syst. Appl. **37**(7), 4779–4786 (2010)
16. Yun, U., Kim, J.: A fast perturbation algorithm using tree structure for privacy preserving utility mining. Expert Syst. Appl. **42**(3), 1149–1165 (2015)