Computers & Security 132 (2023) 103360

Contents lists available at ScienceDirect

Computers & Security

journal homepage: www.elsevier.com/locate/cose

# Efficient privacy preserving algorithms for hiding sensitive high utility itemsets

Mohamed Ashraf*, Sherine Rady, Tamer Abdelkader, Tarek F. Gharib

*Information Systems Department, Faculty of Computer and Information Sciences, Ain Shams University, Cairo 11566, Egypt*

ARTICLE INFO

*Article history:*
Received 12 March 2023
Revised 14 May 2023
Accepted 20 June 2023
Available online 21 June 2023

*Keywords:*
Privacy preserving
Data mining
Utility mining
Sensitive pattern
Privacy metrics
Data utility metrics,

ABSTRACT

Utility-driven mining is a powerful data mining technique that aims to extract significant and valuable knowledge from different kinds of datasets. Nevertheless, analyzing datasets with sensitive or private information may raise security and privacy concerns. To balance utility maximization and privacy preservation, Privacy Preserving Utility Mining (PPUM) has been presented. The primary objective of PPUM algorithms is to conceal the sensitive knowledge that can be found via the application of utility mining algorithms to sensitive data. However, the current PPUM literature shows a paucity of privacy preserving algorithms scalable and efficient enough to handle large and dense datasets. Based on this perspective, this paper proposes three heuristic-based algorithms, namely Selecting the Most Real item sensitive utility First (**SMRF**), Selecting the Least Real item sensitive utility First (**SLRF**), and Selecting the most Desirable Item First (**SDIF**), to efficiently conceal all Sensitive High utility Itemsets (SHIs) while mitigating the projected detrimental impact on the non-sensitive information. The proposed algorithms rely on a novel concept, called Real Item Sensitive Utility (RISU), to effectively select a definite victim item for each SHI during the whole sanitization process. Furthermore, a new sensitive dataset sorting technique is proposed to reduce the time needed to find suitable transactions for sanitization. Through comprehensive experimental evaluations with state-of-the-arts, the viability of the proposed three algorithms was testified. The acquired findings clearly demonstrate the efficacy of the proposed algorithms in terms of reducing the sanitization time and side effects.



## 1. Introduction

With the development of digital technologies in the retail domain, information about customers' behavior is becoming increasingly valuable. The ability to infer knowledge from customers' data can allow many entities to make critical decisions and sustain effective business performance. In this vein, pattern mining technologies (Fournier-Viger et al., 2022) have arisen as a reliable and robust way of analysis in business and industry. Over more than a decade, utility-driven mining (Zhang et al., 2020) has been broadly studied as a pattern mining technology to present new variants and forms of useful patterns. The term High Utility Itemset Mining (HUIM) (Gan et al., 2019) refers mainly to the data mining task that identifies the high-value itemsets in transactional datasets. Compared to Frequent Itemset Mining (FIM) (Agrawal et al., 1994), HUIM is commonly seen as more feasible for real-world applications as it considers critical attributes like the quantity and profit of items to discover more interesting and actionable patterns. In recent years, multiple utility mining algorithms have been introduced to take into account different data features in addition to specific factors and real-life situations. For instance, mining high utility patterns (HUPs) (Liu et al., 2005; Qu et al., 2020; Tseng et al., 2012; Zida et al., 2015), top-k HUPs (Ashraf et al., 2021), closed HUPs (Lin et al., 2022), on-shelf HUPs (Chen et al., 2022), diverse HUPs (Verma et al., 2021), and multi-level HUPs (Tung et al., 2021). The utility mining framework has also been utilized in different applications such as customer segmentation (Krishna and Ravi, 2021), event logs analysis (Fournier-Viger et al., 2020), biological data analysis (Segura-Delgado et al., 2022) and taxi operation data analysis (Liu and Guo, 2021).

There are several tangible benefits that utility-driven mining can offer. However, privacy concerns are another pertinent factor in utility mining literature, which would potentially impair companies' willingness to share their data with their partners for research purposes. Despite their practicality and usefulness, utility-driven techniques may lead to major security and privacy threats if their resultant knowledge became available to untrusted parties (Tran and Hu, 2019). For example, in market research businesses, corporations often acquire sensitive information about cus-

* Corresponding author.
*E-mail address:* Mohamed.a.a.h.is@gmail.com (M. Ashraf).

https://doi.org/10.1016/j.cose.2023.103360
0167-4048/

tomers, including their purchasing history, individual likes, and demographic details. Utility pattern mining can be applied to such sensitive data to uncover hidden patterns and relationships, which can be then used to build profiles of customers or develop an effective product recommendation system. Therefore, companies must assess the sensitivity of the data they are sharing and ensure that the sensitive high utility patterns and relationships are suppressed and can not be uncovered at a specific privacy threshold. Nevertheless, hiding the sensitive patterns is not a trivial task because the original database structure would be destroyed if the sensitive patterns were unwisely and unconsciously wiped, which would cause the utility and reliability of the data to drop drastically.

As a panacea to the aforementioned problems, Privacy Preserving Data Mining (PPDM) (Kenthapadi et al., 2019; Mendes and Vilela, 2017) has emerged. PPDM algorithms remove the sensitive and confidential knowledge from the data sources to, on one side, improve the data sharing and publishing possibilities and, on another side, enhance the utilization of data in the target domain. This in turn allows companies and organizations to control the information that can be elicited from their data and prevents adversaries from either exploiting the resultant information of mining to improve their profits or gaining business and financial advantages. Privacy Preserving Utility Mining (PPUM) (Dinh et al., 2019; Gan et al., 2018) is a branch of PPDM with the ultimate goal of masking all sensitive high utility itemsets (SHIs) while decreasing the adverse impacts of the sanitization process on the non-sensitive high utility itemsets (NHIs).

Thus far, some algorithms (Ashraf et al., 2022; Jangra and Toshniwal, 2022; Lin et al., 2016; Liu et al., 2020b; Yeh and Hsu, 2010) have been proposed to solve the problem of hiding the sensitive high utility patterns in transactional datasets. However, it is analyzed that almost all the existing algorithms only consider datasets of small and medium size, and their processing time and negative impact on the non-sensitive patterns tend to be notably high. In this paper, we precisely study and contribute to the literature of PPUM by presenting a new concept called Real Item Sensitive Utility (RISU) for efficiently sanitizing transactional datasets from sensitive high utility patterns. In particular, three novel heuristic-based algorithms are proposed to overcome the existing issues of past PPUM algorithms. Also, a novel sorting technique is proposed to speed up the sanitization process and reduce the loss in the non-sensitive knowledge. The proposed ideas were validated on four different benchmark datasets. The performance results were found to be quite promising in terms of reducing the sanitization time and side effects compared to the existing contenders.

The remaining content of this paper is arranged as follows. Section 2 reviews the related works on PPUM. Section 3 describes the preliminaries and key notions of HUIM and PPUM frameworks. The proposed perturbation algorithms are introduced in Section 4. Section 5 provides a demonstrative example for the proposed algorithms. Section 6 reports the results of the experimental evaluation. Eventually, Section 7 summarizes the present paper.

## 2. Related works

Over the past decade, there has been modest attention to the issue of security in utility mining. Generally speaking, the existing PPUM algorithms can be decomposed into two main groups. The first group is the item-based sanitization algorithms (Ashraf et al., 2022; Jangra and Toshniwal, 2022; Jisna and Salim, 2018; Lin et al., 2016; Liu et al., 2020a; 2020b; Selvaraj and Kuthadi, 2013; Yeh and Hsu, 2010; Yun and Kim, 2015), in which the utility of appropriate items is modified in each sensitive high utility itemset (SHI) until they are completely hidden. And the second is transaction-based sanitization algorithms (Lin et al., 2014; 2017), in which appropriate transactions are inserted or deleted from the original database by applying some meta-heuristics such as the genetic algorithm (Holland, 1992).

The problem of PPUM was first introduced by Yeh and Hsu (2010) who presented two algorithms, named HHUIF and MSCIF, and defined many basic concepts of PPUM. In both algorithms, the sensitive patterns are concealed by decreasing the quantities of their chosen victim items until the patterns' utilities fall below the minimum utility threshold. Conversely, the victim item selection criterion was different in both of them as the HHUIF algorithm prefers the item with the largest utility for sanitization, while the MSCIF algorithm prefers the item with the maximal occurrence frequency in the sensitive HUIs. In terms of results, HHUIF was found to be weak in reducing the database difference ratio before and after the sanitization process, whereas MSCIF was found to be weak in reducing the number of missing non-sensitive HUIs. For the sake of enhancing the hiding efficiency, Selvaraj and Kuthadi (2013) introduced an enhanced version of the HHUIF algorithm, called HHUIF*, which adopts an item selector strategy, named MHIS, to handle the case when there are multiple candidate victim items possessing similar utility values. HHUIF* was found to be superior compared to HHUIF. Nevertheless, a major drawback in the previous algorithms is that they require multiple database scans to achieve the perturbation operation, which; as could be expected, leads to performance and scalability issues. Motivated by this, Yun and Kim (2015) proposed FPUTT which extends the UP-Growth$^+$ (Tseng et al., 2012) algorithm to identify the sensitive itemsets and collect the needed information for the hiding process. Although its fine speed as it requires only three database scans, FPUTT follows the same hiding technique of HHUIF, and thus it suffers the same side effects.

Lin et al. (2016) introduced the concept of the maximum and minimum utility of sensitive items by proposing two new algorithms, MSU-MAU and MSU-MIU, to conceal SHIs with higher speed and reduce the reactions of the hiding process. Along with that, the authors proposed three new performance measures to better evaluate the hiding efficiency of PPUM algorithms. The proposed algorithms were compared against HHUIF and MSCIF and showed leading performance in reducing the execution time and adverse effects. To further improve the perturbation speed, Jisna and Salim (2018) proposed FPUFC which adopts a Sensitive Itemset Table (SIT) with a multi-set structure to save the required information for the purification procedure. The evaluation results showed that FPUFC has better runtime performance than FPUTT.

To this end, all the previously-mentioned algorithms have a great limitation in that they have no consideration for the unrestricted knowledge. Deleting sensitive itemsets in the dataset can lead to the loss of non-sensitive itemsets due to the shared items between them, which results in reducing the quality of the perturbed dataset. Therefore, many scholars began to pay more attention to the non-sensitive itemsets. Liu et al. (2020a) presented IMSCIF which is an improved version of MSCIF (Yeh and Hsu, 2010) with higher level of security. IMSCIF iteratively chooses victim items by evaluating their conflict count among the sensitive itemsets. As for the victim transaction, it selects the one that contains the minimum number of unrestricted itemsets and the maximum utility value for the sensitive itemset being sanitized. IMSCIF showed poor runtime performance, but promising results regarding reducing the missing cost compared to HHUIF, MSCIF, MSU-MAU, and MSU-MIU algorithms.

Li et al. (2019) have investigated the utilization of integer linear programming (ILP) to find the exact solution to the problem of privacy preserving utility mining. They redefined the hiding method as a constraint satisfaction problem along with a relaxation mechanism to guarantee minimizing the negative impacts produced by the masking operation to the maximum limit. Nonetheless, a major

challenge for their adopted technique is scalability; because finding the exact solution in NP-hard problems is a very computationally expensive task. In other words, finding the exact solution to a NP-hard problem often requires a large amount of computational resources, which can make it infeasible to solve such problems for large datasets in a reasonable amount of time. Therefore, most algorithms of the PPUM framework use heuristics or approximation methods to find good solutions within a reasonable amount of time, even though they cannot guarantee finding the exact solution.

Liu et al. (2020b) presented three data distortion algorithms, SMAU, SMIU, and SMSE, which perform the distortion operation by scanning the database twice. The three algorithms rely on a complex data structure that saves the required information about the sensitive and non-sensitive itemsets. The transaction possessing the least number of non-sensitive patterns is chosen for purification in the three algorithms. The core difference between the three algorithms lies in the way they select the victim item with. SMAU algorithm hides SHIs by reducing the utility of the item with the largest utility value. SMIU algorithm hides SHIs by reducing the utility of the item with the least utility value. SMSE algorithm determines the victim item such that it exists in the largest number of sensitive itemsets. The three algorithms showed competitive results in terms of execution time and side effects, though their high memory consumption.

Recently, Jangra and Toshniwal (2022) have proposed two novel algorithms, MinMax and Weighted, to achieve fast perturbation while limiting the data distortion ratio. Each algorithm adopts a different victim item selection strategy and relies on three dataset sorting techniques, in particular, *DoC_RoT*, *RoT_DoC*, and transaction length, to determine the victim transaction. Unlike previous item-based sanitization studies, a specific item is chosen in each sensitive itemset to be the victim item of this itemset during the whole sanitization process. The rationale behind this is that the chosen victim item will most likely cause the least negative impacts on the non-sensitive itemsets. In terms of findings, the double sorting technique, *RoT_DoC*, was found to be the most effective for the two algorithms. However, we argue that the double sorting technique brings relatively high computational cost, especially in large and highly sparse datasets. Moreover, in both algorithms, the utilized victim item selection techniques only consider the sensitive and non-sensitive itemsets yet neglect the influence of the chosen victim items on the sensitive transactions as a whole, which may consequently lead to unfavorable results.

More recently, Ashraf et al. (2022) developed SB2VF algorithm with the insight that highly efficient sanitization can be achieved by picking the victim items and transactions based on effective criteria. They suggested that two items should be chosen as candidate victim items in each sensitive itemset, which are the one that overlaps in most sensitive itemsets and the one that overlaps in most non-sensitive itemsets. These two items are modified interchangeably while masking their sensitive itemsets such that the one possessing a lower utility value in the processed transaction is more worthy of modification. They also suggested triple sorting the sensitive transactions to ensure that lower-side-effects transactions are sanitized first. By using three benchmark datasets, the efficacy of their ideas was testified. The same authors also presented HUP-Hiding algorithm (Ali et al., 2023), which opts for single or multiple victim items for each sensitive itemset based on a specific criterion and orders the sensitive transactions based on the utility of the sensitive itemsets.

All the above-discussed algorithms adopted the item-based sanitization model to hide SHIs, which is the model this paper is most concerned about. We summarize the core advantages and disadvantages of the main item-based sanitization algorithms in Table 1. Other studies followed the transaction-based sanitization model for the task of PPUM. Lin et al. (2014) used the genetic algorithm and the pre-large concept to sanitize the dataset by inserting new transactions while reducing the time required to scan the database. The authors in (Lin et al. 2017) proposed a genetic algorithm-based approach, called PPUMGAT. Their proposed approach automatically deletes sensitive transactions to conceal SHIs and uses a fitness function to evaluate the fineness of the chromosomes (candidate solutions).

Meanwhile, few algorithms have been put forward to hide both sensitive utility and frequent itemsets at the same time. In (Rajalaxmi and Natarajan, 2012) two generic algorithms were introduced, namely MSMU and MCRSU. In the MSMU algorithm, the hiding process is performed by selecting the transaction containing the least number of unrestricted itemsets as the victim transaction, and the item having minimal support or maximal utility as the victim item. MCRSU algorithm adopts a similar approach but uses the ratio of unrestricted itemsets influenced by the perturbation process for the victim item selection. It was shown that the MCRSU algorithm is more efficient than MSMU in decreasing the missing cost and preserving the goodness of the purified dataset. Later on, Liu et al. (2018) developed HUFI algorithm to protect sensitive utility and frequent patterns. The developed HUFI applies the idea of maximum boundary to control the sanitization way.

Some works focused on applying new techniques to conduct the sanitization process. Bandil et al. (2018) applied the principle of differential privacy with Laplace transformations to obscure the sensitive utility itemsets. Trieu et al. (2018) introduced HHUARL algorithm to conceal the sensitive high utility association rules using the intersection lattice of itemsets (Grätzer, 2011) and the concept of transaction-weighted utility (Liu et al., 2005).

Besides the previously-mentioned algorithms, a number of studies investigated the problem of PPUM in sequential databases. For example, Le et al. (2018) proposed HUS-Hiding to address the problem of masking sensitive utility patterns in quantitative sequence datasets. In (Huynh et al. 2022) three multi-core parallel algorithms were introduced to increase the optimality of the hiding process.

Given the previous survey and the detailed comparison in Table 1, it appears clearly that the former research studies of the standard PPUM framework have paid little attention to maintaining the quality of the resultant transactional dataset; as many non-sensitive itemsets are lost during the perturbation process, and the difference between the original and purified dataset tend to be huge, leaving a room for improvement when it comes to reducing the required time for sanitization, maximizing the dataset utility, and minimizing the concomitant negative effects. To improve upon the prior algorithms, this research paper proposes three best-effort algorithms that are not only fast but also effective in lowering the expected negative consequences on the final sanitized dataset. In summary, the main contributions of this work are highlighted as next:

1. Three item-based sanitization algorithms with three different victim-item selection techniques are designed to maintain the sanitization performance while increasing the degree of privacy.
2. A proposal for the concept of Real Item Sensitive Utility (RISU), that can be leveraged to select an ideal victim item for each sensitive itemset in the initial stages of the sanitization process, resulting in faster sanitization.
3. A novel sensitive dataset sorting technique called Weighted Sorting is introduced. This technique is utilized for the victim transaction selection process for all the proposed algorithms to give priority for sanitization to transactions that may cause fewer side effects.
4. For the verification of the proposed algorithms efficiency, extensive comparisons are conducted on four diverse benchmark

**Table 1**
Comparison between the main PPUM item-based sanitization algorithms.

| Algorithm | Sanitization strategy | Advantages | Disadvantages |
|---|---|---|---|
| HHUIF (Yeh and Hsu, 2010) | -Select the item with the highest utility as victim. | -Simplicity of the design and implementation. | -Require several database scans. -Lose considerable non-sensitive information in the sanitized database. |
| MSICF (Yeh and Hsu, 2010) | -Select the item with the highest conflict degree among all the sensitive itemsets as victim. -Select the transaction that has the highest utility of the victim item. | -Simplicity of the design and implementation. | -Require several database scans. -Lose considerable non-sensitive information in the sanitized database. |
| FPUTT (Yun and Kim, 2015) | -Select the item with the highest utility as victim. | -Require only three database scans. | -Sanitization is still slow. -Lose considerable non-sensitive information in the sanitized database. |
| MSU-MAU (Lin et al., 2016) | -Select the transaction that has the highest utility of the sensitive itemset. -Select the item with the highest utility as victim. | -Avoid scanning the database multiple times by storing the sensitive transactions of each sensitive itemset in an index table. | -The loss in the non-sensitive information is still high. |
| MSU-MIU (Lin et al., 2016) | -Select the transaction that has the highest utility of the sensitive itemset. -Select the item with the lowest utility as victim. | -Avoid scanning the database multiple times by storing the sensitive transactions of each sensitive itemset in an index table. | -The loss in the non-sensitive information is still high. |
| SMAU (Liu et al., 2020b) | -Select the transaction that has the least number of non-sensitive itemsets. -Select the item with the highest utility as victim. | -Require only two database scans. -Can reduce the loss in the non-sensitive information. | -Suffer from scalability issues due to its expensive data structure. |
| SMIU (Liu et al., 2020b) | -Select the transaction that has the least number of non-sensitive itemsets. -Select the item with the lowest utility as victim. | -Require only two database scans. -Can reduce the loss in the non-sensitive information. | -Suffer from scalability issues due to its expensive data structure. |
| SMSE (Liu et al., 2020b) | -Select the transaction that has the least number of non-sensitive itemsets. -Select the item with the highest conflict degree among the sensitive itemsets and lowest conflict degree among the non-sensitive itemsets as victim. | -Require only two database scans. -Can reduce the loss in the non-sensitive information. | -Suffer from scalability issues due to its expensive data structure. |
| Weighted (Jangra and Toshniwal, 2022) | -The item with the lowest calculated weight is the ideal victim item of the sensitive itemset. -Transactions are selected based on predefined sorting criteria. | -No need to evaluate the victim items and transactions multiple times | -Double sorting the database can be costly. -Neglect the influence of the chosen victim items on the sensitive transactions as a whole. |
| MinMax (Jangra and Toshniwal, 2022) | -The item with the lowest conflict degree among the non-sensitive itemsets is the ideal victim item of the sensitive itemset. -Transactions are selected based on predefined sorting criteria. | -No need to evaluate the victim items and transactions multiple times | -Double sorting the database can be costly. -Neglect the influence of the chosen victim items on the sensitive transactions as a whole. |

**Table 2**
An example transactional dataset.

| Tid | Transaction (item, purchase quantity) |
|---|---|
| $T_1$ | (A, 1), (B, 3), (E, 2) |
| $T_2$ | (C, 4), (D, 5) |
| $T_3$ | (E, 2), (F, 3) |
| $T_4$ | (A, 5), (B, 4), (C, 2), (D, 2), (E, 4) |
| $T_5$ | (B, 5), (C, 4), (E, 6), (F, 1) |
| $T_6$ | (B, 1), (C, 3), (E, 6) |
| $T_7$ | (A, 2), (E, 3), (F, 4) |

**Table 3**
Unit profits of items.

| Item | Unit profit |
|---|---|
| A | 6 |
| B | 4 |
| C | 2 |
| D | 5 |
| E | 3 |
| F | 9 |

datasets using five performance measures. The obtained results indicate that the adopted ideas work very well in large and dense datasets and can diminish the loss in the non-sensitive itemsets by up to 40% compared to the current existing algorithms.

## 3. Preliminaries

Suppose that $D$ is a quantitative transaction dataset coming in the form of a set of transactions such that $D = \{T_1, T_2, ..., T_n\}$ where $T_i$ $(1 \leq i \leq n)$ is a transaction with its identifier $i$. For instance, Table 2 presents an example of a transactional dataset $D$ having seven transactions with their identifiers. These transactions may represent customers' purchase behavior or users' particular activities. Suppose $I = \{i_1, i_2, ..., i_m\}$ is a finite set of distinct items, each transaction $T_i$ entails $m$ quantitative items, each item $i_j \in I$ has an external utility value, $eu(i_j)$, which reflects its selling revenue or weight as shown in Table 3, and has an internal utility value $iu(i_j)$, which reflects its selling quantity or frequency in each transaction $T_i$. An itemset $X$ can be seen as a set of $k$ distinct items and can be referred to as $k$-itemset. A transaction $T_i$ can be regarded as a supportive transaction for $X$ if and only if $X \subseteq T_i$.

To better grasp the above preliminaries, lets inspect the example dataset $D$ given in Table 2. There are seven transactions in total, which are represented as $T_1, T_2, T_3, T_4, T_5, T_6$ and $T_7$. Also, there are six items in $D$, expressed as $A, B, C, D, E$ and $F$. The set of positive unit profits associated with selling these items is provided in Table 3. In principle, each transaction indicates selling specific items. For example, transaction $T_1$ in $D$ implies that items $A, B$ and $E$ were purchased in this transaction with quantities of 1, 3 and 2, respectively.

**Definition 1.** The utility of item $x \in T_i$ is computed as, $u(x, T_i) = iu(x, T_i) \times eu(x)$. E.g., in Table 2, $u(A, T_1) = iu(A, T_1) \times eu(A) = 1 \times 6 = 6$.

**Table 4**
The generated HUIs when minUtil = 70.

| Itemset | Utility |
|---|---|
| F | 72 |
| A, B | 72 |
| A, E | 87 |
| B, E | 102 |
| E, F | 105 |
| A, B, E | 90 |
| B, C, E | 106 |
| A, B, C, D, E | 72 |

**Table 5**
The sensitive HUIs and their transactions.

| Itemset | Transaction ID |
|---|---|
| A, B, E | $T_1, T_4$ |
| B, C, E | $T_4, T_5, T_6$ |
| A, B, C, D, E | $T_4$ |

**Definition 2.** For an itemset $X \subseteq T_i$, $u(X)$ in $T_i$ can be defined as, $u(X, T_i) = \sum_{x_j \in X \wedge X \subseteq T_i} u(x_j, T_i)$. E.g., in Table 2, $u(\{A, B\}, T_1) = u(A, T_1) + u(B, T_1) = 6 + 12 = 18$.

**Definition 3.** The gross utility of an itemset $X$ in the dataset $D$ is defined as: $u(X) = \sum_{X \subseteq T_i \wedge T_i \in D} u(X, T_i)$. E.g., in Table 2, $u(\{A, B\}) = u(\{A, B\}, T_1) + u(\{A, B\}, T_4) = 18 + 46 = 64$.

**Definition 4.** For any transaction $T_i \in D$, the transaction utility of $T_i$ can be calculated as, $TU(T_i) = \sum_{x \in T_i} u(x, T_i)$. E.g., in Table 2, $TU(T_3) = u(E, T_3) + u(F, T_3) = 6 + 27 = 33$.

**Definition 5.** A high utility itemset (HUI) is an itemset whose utility value is greater than or equal to a user-preferred minimum utility threshold (minUtil). The task of mining HUIs from transactional datasets is to find all the HUIs that satisfy the given minUtil. The generated high-utility itemsets from the example dataset $D$ when minUtil = 70 is provided in Table 4.

**Definition 6.** A sensitive high utility itemset (*SHI*) is a very valuable pattern that should be kept confidential and undetectable by any utility-driven mining technique because it can divulge secret knowledge or private information about the data owner. In contrast, a non-sensitive high utility itemset (*NHI*) should be kept detectable to ensure data reliability and validity. It is worth mentioning that what is considered sensitive information can change with the context and the domain of the data and the stakeholders involved in the analysis, for instance, in marketing engineering example, if a high utility pattern reveals the purchasing habits or behavior of customers from a certain gender or race, it could be seen as a sensitive pattern as it may raise ethical concerns. Considering the running example, the sensitive itemsets are outlined in Table 5.

**Definition 7.** Consider a sensitive high utility itemset $SHI_i$, a sensitive item $i_s$ and a transaction $T_j$ such that $i_s \in SHI_i$ and $SHI_i \subseteq T_j$. $i_s$ is said to be the victim item of $SHI_i$, more formally, $I_{vic}(SHI_i)$, if the utility of $i_s$ needs to be decreased to hide the sensitive itemset $SHI_i$. The hypothesis behind choosing item $i_s$ specifically among all the items in $SHI_i$ is that the non-sensitive high utility itemsets (NHIs) are not immune from masking, thus choosing this item based on effective criteria can greatly reduce their loss. In a similar vein, $T_j$ is said to be the victim transaction of $SHI_i$, to be specific, $T_{vic}(SHI_i)$, if it was chosen to modify the victim item. Considering the running example, according to Table 5, the sensitive items are $A, B, C, D, E$ and the sensitive transactions are $T_1, T_4, T_5, T_6$.

**Definition 8.** Sensitive cover of an item $x_i$, more formally $SC(x_i)$, refers to the number of SHIs containing the item $x_i$. Considering the running example, the sensitive covers of all the sensitive items are given in Table 6.

**Table 6**
The sensitive and non-sensitive cover of items.

| Sensitive item | A | B | C | D | E |
|---|---|---|---|---|---|
| SC | 2 | 3 | 2 | 1 | 3 |
| NSC | 2 | 2 | 0 | 0 | 3 |

**Table 7**
The sensitive and non-sensitive cover of transactions.

| Sensitive transaction Tid | $T_1$ | $T_4$ | $T_5$ | $T_6$ |
|---|---|---|---|---|
| SC | 1 | 3 | 1 | 1 |
| NSC | 3 | 3 | 3 | 1 |

**Definition 9.** Sensitive cover of a transaction $T_j$, more formally $SC(T_j)$, refers to the number of SHIs appeared in transaction $T_j$. Considering the running example, the sensitive covers of all the sensitive transactions are given in Table 7.

**Definition 10.** Non-sensitive cover of an item $x_i$, more formally $NSC(x_i)$, refers to the number of NHIs containing the item $x_i$. Considering the running example, the non-sensitive covers of all the sensitive items are given in Table 6.

**Definition 11.** Non-sensitive cover of a transaction $T_j$, more formally $NSC(T_j)$, refers to the number of NHIs appeared in transaction $T_j$. Considering the running example, the non-sensitive covers of all the sensitive transactions are given in Table 7.

**Definition 12.** The sensitive weight of a transaction $T_j$, $Wt(T_j)$, can be calculated using the following formula:

$$Wt(T_j) = \frac{SC(T_j)}{NSC(T_j) + 1} \tag{1}$$

Because a sensitive transaction may have no non-sensitive itemsets, we added 1 to the denominator of the above formula. In essence, the high Sensitive Cover (SC) of a transaction denotes that the removal of the victim item will most likely hide more sensitive itemsets, whereas the low Non-Sensitive Cover (NSC) denotes that the modification of the transaction will most likely produce fewer side effects. Hence, the transactions possessing high weight should be sanitized first.

**Definition 13.** To mitigate the information loss caused by transactions distortion, inspired by the double sorting technique in (Jangra and Toshniwal, 2022) we propose our Weighted Sorting technique. This technique implies that sensitive transactions are ordered according to their weight descending order. In other words, transactions with higher weight have higher priority for sanitization. Considering the running example, the sanitization order of the sensitive transactions is $T_4 \prec T_6 \prec T_1 \prec T_5$.

**Definition 14.** Let be a sensitive item, $x$, and the set of all sensitive transactions in the example dataset, $ST$. The Real Item Sensitive Utility of $x$, $RISU(x)$, can be calculated using the following formula:

$$RISU(x) = \sum_{x \in T_j \wedge T_j \in ST} u(x, T_j) \tag{2}$$

Finding the optimal victim item in a sensitive itemset is a delicate and rather complicated task. This is because modifying items in sensitive itemsets can lead to unintended side effects, especially if the modified items are overlapping with many sensitive transactions and non-sensitive itemsets. This is where the concept of Real Itemset Sensitive Utility (RISU) comes to display. The RISU

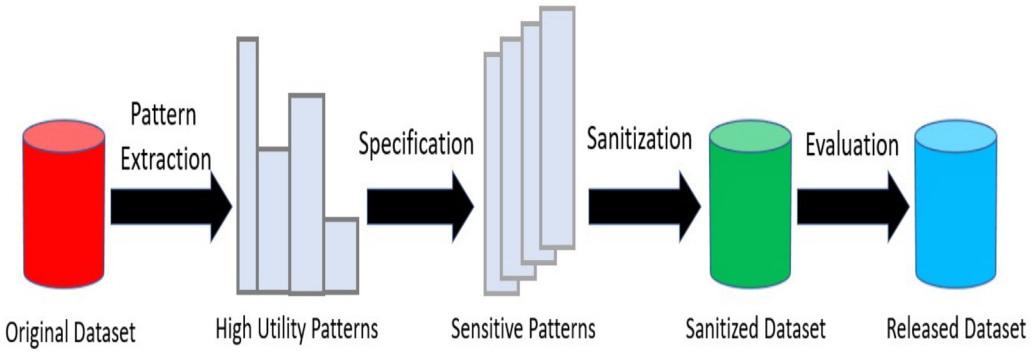


**Fig. 1.** The generic framework of the privacy preserving utility mining process.

**Table 8**
RISU values of items.

| Sensitive item | A | B | C | D | E |
|---|---|---|---|---|---|
| RISU | 48 | 48 | 18 | 10 | 54 |

values can be seen as a reference to the contribution of the sensitive items to the total utility of their sensitive transactions. Hence, these values can be used as an evaluator to find a proper victim item for each SHI, which can enhance the potential performance of the proposed algorithms for the sensitive knowledge removal task. Considering the running example, the RISU values of all the sensitive items are provided in Table 8.

Fig. 1 shows the generic framework for the PPUM process. On overall, given a transactional dataset *D*, a profitable table and a minimum privacy threshold (minUtil). The task of PPUM or so-called High Utility Pattern Hiding (HUPH) includes the next four major procedures: (1) a utility-driven algorithm is executed on *D* to produce all the HUIs at a specific minUtil (**Pattern Extraction**); (2) among the produced HUIs, the sensitive ones should be determined according to some business requirements, a privacy policy or the data owners preferences (**Specification**); (3) a sanitizing algorithm is executed for the sake of hiding all the predetermined sensitive itemsets with the least side effects (**Sanitization**); (4) The side effects of the sanitization process are assessed in relation to the sensitive and non-sensitive patterns indicated in the second stage (**Evaluation**). The distortion process of the PPUM algorithms that adopt the item-based sanitization approach involves selecting some items in the sensitive patterns to be sacrificed, whether by deleting them or reducing their internal utilities in some transactions until the patterns' utilities or reliability fall below the minUtil.

## 4. The proposed algorithms

Privacy Preserving Utility Mining (PPUM) is a definition that makes a promise to data owners that their private and sensitive information will not be exposed, adversely or otherwise, if they allowed sharing or publishing their data to be used in any utility-driven analytical operation. Aligning with this definition, we propose three heuristic algorithms for PPUM: (1) Selecting the Most Real item sensitive utility First (SMRF); (2) Selecting the Least Real item sensitive utility First (SLRF); and (3) Selecting the most Desirable Item First (SDIF).

Briefly, the efficiency and effectiveness of any PPUM algorithm depend fundamentally on three factors: (1) the selected victim items; (2) the selected victim transactions; (3) the processing and hiding order of the sensitive itemsets.

To select a victim transaction, the proposed algorithms utilize a dataset sorting technique, named Weighted Sorting, which arranges the sensitive transactions as per their estimated weight in descending order. This is because transactions with higher weight will most likely support higher number of sensitive itemsets and lower number of non-sensitive itemsets. This inevitably ensures that the non-sensitive patterns will be more resistant to distortion and many sensitive patterns will be affected or even concealed by modifying the victim transaction. For the selection of victim items, the proposed three algorithms make efficient use of the calculated RISU values of the sensitive items to specify a definite victim item for each sensitive itemset. This consequently expedites the whole perturbation process because the proposed algorithms will not need to select and evaluate multiple victim items for each SHI during the sanitization procedure. To optimize the hiding process, the sensitive itemsets are processed as per the descending order of the RISU values of their prespecified victim items. Based on three different criteria for choosing a single victim item for each sensitive itemset and one dataset scan, we propose three algorithms, SMRF, SLRF and SDIF. Fig. 2 demonstrates the overall working way of the proposed algorithms. The explanations of the sanitization processes of the proposed three algorithms are given as next:

### 4.1. The SMRF algorithm

The pseudo-code of the SMRF algorithm can be viewed in Algorithm 1. At first, the dataset is scanned once to determine the victim transactions and calculate the RISU values of the sensitive items using Eqn 2 (Line 1). Then, for each sensitive itemset, the item with the most RISU value is selected to be the victim item of this itemset (Lines 2–4). Subsequently, for each sensitive transaction, a sensitive weight is calculated using Eqn 1 (Lines 5–7). In line 8, the estimated weight values are then used to apply the Weighted Sorting technique. Line 9 sorts the sensitive itemsets as per the non-increasing order of the RISU values of their selected victim items. The sensitive patterns (SHIs) are then concealed in a one-by-one manner (Lines 10–31). The algorithm iteratively hides each SHI by computing the difference using the equation in Line 11. The difference value reflects the minimum utility needed to be reduced from the sensitive itemsets so that it becomes hidden. The algorithm then begins traversing the sorted sensitive transactions to identify a victim transaction and modify the victim item of the pattern being processed. If a supportive transaction was found and the difference value was still greater than zero, then the victim item in this transaction will be either deleted or modified (Lines 13–27). In case the difference value is greater than the utility of the victim item in the victim transaction, the victim item is deleted from the transaction (Lines 16–19). If this is not the case, the quantity of the victim item is reduced by the value calculated in line 21. In both previous cases, the utility of the remaining sensitive

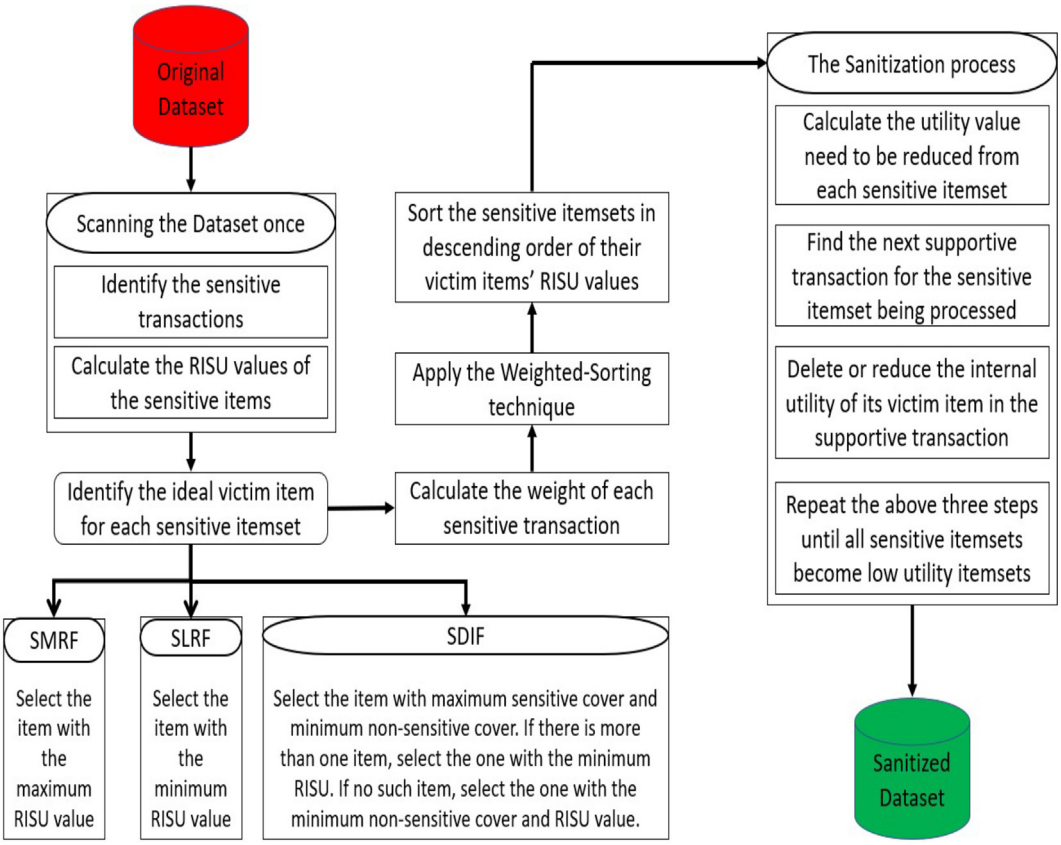


**Fig. 2.** The working way of the proposed algorithms.

patterns having this victim item and existing in the same victim transaction is also reduced. The modified sensitive transactions are then inserted into the original dataset when all the sensitive patterns are completely concealed, and a fully sanitized dataset is finally produced (Line 32–33).

### 4.2. The SLRF algorithm

The SLRF algorithm follows the same methodology as the SMRF algorithm, except that in the item selection process, the item with the least RISU value in the sensitive pattern is chosen to be the victim item of this pattern during the whole sanitization process. Algorithm 2 shows the sanitization process of the proposed SLRF. At first, a dataset scan is executed to identify the sensitive transaction and estimate the RISU values (Line 1). Then, we assign a victim item to each sensitive itemset by selecting the one with the least RISU value among the sensitive items of each itemset (Lines 2–4). In lines 5–8, the sensitive weight of each transaction is estimated and the Weighted Sorting technique is then applied to the sensitive transactions. Next, the sensitive itemset having the victim item with the most RISU value compared to the RISU values of chosen victim items of other sensitive itemsets, is chosen to be sanitized first. The algorithm then proceeds with the same hiding technique demonstrated previously in the SMRF algorithm.

### 4.3. The SDIF algorithm

Algorithm 3 describes the overall process of the SDIF algorithm. Unlike the two previous algorithms, which consider only the RISU values to select a victim item for each sensitive itemset, the SDIF algorithm adopts more advanced evaluation criteria. In lines(2–9), the victim item selection process puts into consideration three different factors; (1) the sensitive cover of each item; (2) the non-sensitive cover of each item; (3) the RISU value of each item. During the item evaluation process, the algorithm first tries to select the most desirable item, which is the item having the most sensitive cover and the least non-sensitive cover compared to the other items. If no desirable item was found, the algorithm selects the item with the least non-sensitive cover to be the victim item. Here, the RISU values are leveraged as a tie-breaker for both previous cases such that the algorithm always chooses the item with the least RISU value if there is more than one candidate victim item for the sensitive pattern. The sensitive transactions are then arranged according to their weights, whereas the sensitive itemsets are arranged according to their victim items' RISU values (Lines 10–14). In lines (15–36), the sanitization procedure is iteratively applied to the sensitive patterns until they are all hidden.

## 5. Demonstrative example

In this part, we provide a demonstrative example to better elucidate the sanitization process of the proposed three algorithms. Consider the dataset $D$ in Table 2 and the items profit values in Table 3, the high utility itemsets when the minUtil = 70 are depicted in Table 4 and the sensitive itemsets with their transactions are outlined in Table 5. The three algorithms start by scanning the original dataset once to determine the victim transactions and calculate the RISU values of the sensitive items. The RISU values of all sensitive items are provided in Table 8. Then, all the high utility itemsets are scanned once to estimate the sensitive and non-sensitive cover of items as shown in Table 6. Also, the sensitive transactions are scanned once to estimate the sensitive and non-sensitive cover of transactions as shown in Table 7. After that, each algorithm follows a different way to specify an ideal victim item

**Algorithm 1:** SMRF algorithm.

**Input**: $D^*$, the sensitive dataset; SHIs, the set of sensitive high utility itemsets; NHIs, the set of non-sensitive high utility itemsets; minUtil, the minimum utility threshold.
**Output**: $D'$, a sanitized dataset.

Scan the dataset once to determine the sensitive transactions and compute the RISU values of the sensitive items using Eq. (2)
**foreach** $S_i \in SHIs$ **do**
  $I_{vic}(S_i) \leftarrow x_i$ such that $x_i \in S_i$ and $\forall x_j \in S_i$ $RISU(x_i) \geq RISU(x_j)$
**end foreach**
**foreach** *transaction* $T_i \in D^*$ **do**
  Calculate the weight of $T_i$ using Eq. (1)
**end foreach**
Sort transactions $T_i \in D^*$ in descending order of their weight (Weighted Sorting)
Sort itemsets $S_i \in SHIs$ according to decreasing order of the RISU values of their already selected victim items $I_{vic}(S_i)$
**foreach** $S_i$ *in sorted SHIs* **do**
  $diff = u(S_i) - minUtil + 1$
  **foreach** *transaction* $T_i \in D^*$ **do**
    **if** $diff > 0$ **then**
      **if** $S_i \subseteq T_i$ **then**
        $T_{vic}(S_i) = T_i$
        **if** $diff \geq u(I_{vic}(S_i), T_{vic}(S_i))$ **then**
          delete $I_{vic}$ from $T_{vic}$
          $diff = diff - u(I_{vic}(S_i), T_{vic}(S_i))$
          Update $S_j \in SHIs$ such that $S_j \subseteq T_{vic}$ and $I_{vic}(S_i) \in S_j$
        **end if**
        **else**
          $diu = \lceil diff/eu(I_{vic}(S_i)) \rceil$
          $iu(I_{vic}(S_i), T_{vic}(S_i)) = iu(I_{vic}(S_i), T_{vic}(S_i)) - diu$
          Update $S_j \in SHIs$ such that $S_j \subseteq T_{vic}$ and $I_{vic}(S_i) \in S_j$
          $diff = 0$
        **end if**
      **end if**
    **end if**
    **else**
      break
    **end if**
  **end foreach**
**end foreach**
$D' \leftarrow D^*$
**return** the sanitized dataset $D'$

**Algorithm 2:** SLRF algorithm.

**Input**: $D^*$, the sensitive dataset; SHIs, the set of sensitive high utility itemsets; NHIs, the set of non-sensitive high utility itemsets; minUtil, the minimum utility threshold.
**Output**: $D'$, a sanitized dataset.

Scan the dataset once to determine the sensitive transactions and compute the RISU values of the sensitive items using Eq. (2)
**foreach** $S_i \in SHIs$ **do**
  $I_{vic}(S_i) \leftarrow x_i$ such that $x_i \in S_i$ and $\forall x_j \in S_i$ $RISU(x_i) \leq RISU(x_j)$
**end foreach**
**foreach** *transaction* $T_i \in D^*$ **do**
  Calculate the weight of $T_i$ using Eq. (1)
**end foreach**
Sort transactions $T_i \in D^*$ in descending order of their weight (Weighted Sorting)
Sort itemsets $S_i \in SHIs$ according to decreasing order of the RISU values of their already selected victim items $I_{vic}(S_i)$
**foreach** $S_i$ *in sorted SHIs* **do**
  $diff = u(S_i) - minUtil + 1$
  **foreach** *transaction* $T_i \in D^*$ **do**
    **if** $diff > 0$ **then**
      **if** $S_i \subseteq T_i$ **then**
        $T_{vic}(S_i) = T_i$
        **if** $diff \geq u(I_{vic}(S_i), T_{vic}(S_i))$ **then**
          delete $I_{vic}$ from $T_{vic}$
          $diff = diff - u(I_{vic}(S_i), T_{vic}(S_i))$
          Update $S_j \in SHIs$ such that $S_j \subseteq T_{vic}$ and $I_{vic}(S_i) \in S_j$
        **end if**
        **else**
          $diu = \lceil diff/eu(I_{vic}(S_i)) \rceil$
          $iu(I_{vic}(S_i), T_{vic}(S_i)) = iu(I_{vic}(S_i), T_{vic}(S_i)) - diu$
          Update $S_j \in SHIs$ such that $S_j \subseteq T_{vic}$ and $I_{vic}(S_i) \in S_j$
          $diff = 0$
        **end if**
      **end if**
    **end if**
    **else**
      break
    **end if**
  **end foreach**
**end foreach**
$D' \leftarrow D^*$
**return** the sanitized dataset $D'$

for each sensitive itemset. Considering the current example, the sensitive itemsets are {A, B, E}, {B, C, E} and {A, B, C, D, E}. As for the SMRF algorithm, the item with the maximum RISU value in each sensitive itemset will be selected as the victim item during the whole sanitization process. For the itemset {A, B, E}, item E is chosen to represent the victim item of this itemset. This is because RISU(A) = 48, RISU(B) = 48 and RISU(E) = 54. Similarly, item E is chosen in {B, C, E} and {A, B, C, D, E} itemsets. For the SLRF algorithm, the item with the minimum RISU value will be the victim item. For the itemset {A, B, E}, item A and item B have the least RISU value, thus any one of them can be selected. For the itemsets {B, C, E} and {A, B, C, D, E}, items C and D are selected as victim items for both itemsets respectively. This is because RISU(C) = 18 and RISU(D) = 10. The SDIF algorithm considers the Sensitive Cover SC and Non-Sensitive Cover NSC of items during the evaluation process. The most desirable item, which has the maximum SC and the minimum NSC, is selected in each itemset. For itemset {A, B, E}, the maximum SC = 3 and the minimum NSC = 2. Therefore, item B is chosen as the victim item of the previous itemset. For itemset {B, C, E}, the maximum SC = 3 and the minimum NSC = 0. Thus, no desirable item is found in this itemset. In such a case, item C is chosen as the victim item since it has the minimum NSC value. For the itemset {A, B, C, D, E}, no desirable item is found and there are two items, C and D, that have the minimum NSC value. For this case, the item with the least RISU value is chosen, which is item D. Subsequently, after the victim items are determined, the Weighted Sorting technique is applied to the sensitive transactions. Since the sensitive transactions are , the sanitization order will be

**Algorithm 3:** SDIF algorithm.

**Input**: $D^*$, the sensitive dataset; SHIs, the set of sensitive high utility itemsets; NHIs, the set of non-sensitive high utility itemsets; minUtil, the minimum utility threshold.
**Output**: $D'$, a sanitized dataset.

Scan the dataset once to determine the sensitive transactions and compute the RISU values of the sensitive items using Eq. (2)
**foreach** $S_i \in SHIs$ **do**
  $CandI_{vic} \leftarrow x_i$ such that $NSC(x_i) \leq NSC(x_j)$ and $SC(x_i) \geq SC(x_j)\ \forall x_j \in S_i$
  **if** $|CandI_{vic}| \geq 1$ **then**
    $I_{vic}(S_i) \leftarrow x_i$ such that $RISU(x_i) < RISU(x_j)$ $\forall x_j \in CandI_{vic}$
  **end if**
  **else**
    $I_{vic}(S_i) \leftarrow x_i$ such that $NSC(x_i) < NSC(x_j)$ and $RISU(x_i) < RISU(x_j)\ \forall x_j \in S_i$
  **end if**
**end foreach**
**foreach** *transaction* $T_i \in D^*$ **do**
  Calculate the weight of $T_i$ using Eq. (1)
**end foreach**
Sort transactions $T_i \in D^*$ in descending order of their weight (Weighted Sorting)
Sort itemsets $S_i \in SHIs$ according to decreasing order of the RISU values of their already selected victim items $I_{vic}(S_i)$
**foreach** $S_i$ *in sorted SHIs* **do**
  $diff = u(S_i) - minUtil + 1$
  **foreach** *transaction* $T_i \in D^*$ **do**
    **if** $diff > 0$ **then**
      **if** $S_i \subseteq T_i$ **then**
        $T_{vic}(S_i) = T_i$
        **if** $diff \geq u(I_{vic}(S_i), T_{vic}(S_i))$ **then**
          delete $I_{vic}$ from $T_{vic}$
          $diff = diff - u(I_{vic}(S_i), T_{vic}(S_i))$
          Update $S_j \in SHIs$ such that $S_j \subseteq T_{vic}$ and $I_{vic}(S_i) \in S_j$
        **end if**
        **else**
          $diu = \lceil diff/eu(I_{vic}(S_i)) \rceil$
          $iu(I_{vic}(S_i), T_{vic}(S_i)) = iu(I_{vic}(S_i), T_{vic}(S_i)) - diu$
          Update $S_j \in SHIs$ such that $S_j \subseteq T_{vic}$ and $I_{vic}(S_i) \in S_j$
          $diff = 0$
        **end if**
      **end if**
    **end if**
    **else**
      break
    **end if**
  **end foreach**
**end foreach**
$D' \leftarrow D^*$; **return** the sanitized dataset $D'$

$T_4 \prec T_6 \prec T_1 \prec T_5$. In the next step, we order the sensitive itemsets based on the RISU value of their victim items in descending order. Following this, each algorithm operates as next.

In the SMRF algorithm, the processing order of sensitive itemsets is {B, C, E} - {A, B, C, D E} - {A, B, E}. For the itemset {B, C, E}, the difference value ($diff$) that need to be reduced to hide this itemset is $diff = 106 - 70 + 1 = 37$. For the three transactions $T_4$, $T_5$ and $T_6$, which contain the itemset being processed, transaction $T_4$ is first chosen as the victim transaction because it has a higher weight than the other two transactions. Since the utility of the victim item E in $T_4$ is 12, which is less than the value of $diff$, then item E is deleted from transaction $T_4$ and the utility of {B, C, E} is decreased to 74. Because both the sensitive itemsets {A, B, E} and {A, B, C, D, E} appear in the current transaction $T_4$ and contain item E, their utilities are also decreased to 32 and 0 respectively. Next, {B, C, E} continues to be hidden as transaction $T_6$ is selected for modification. Since the utility of E in $T_6$ is less than $diff$, the internal utility of E is reduced from 6 to 4 and the utility of the current itemset is reduced to 68, which is lower than the given minUtil. As a result, the itemset {B, C, E} is completely hidden, and the other remaining sensitive itemsets, {A, B, E} and {A, B, C, D, E}, do not need to be processed as they are also hidden. Yet, the non-sensitive itemsets, {A, E} and {B, E}, are lost in error.

For the SLRF algorithm, the processing order of sensitive itemsets is {A, B, E} - {B, C, E} - {A, B, C, D E}. For the itemset {A, B, E}, the internal utility of its victim item A is decreased from 5 to 1 in the victim transaction $T_4$. As a result, both {A, B, E} and {A, B, C, D E} itemsets are masked. In order to hide itemset {B, C, E}, its victim item C is removed from the victim transactions $T_4$ and $T_6$, respectively. As a result, to the sanitizing procedure, the non-sensitive itemsets, {A, E} and {A, B}, are also hidden.

In the SDIF algorithm, the processing order of sensitive itemsets is {B, C, E} - {A, B, E} - {A, B, C, D E}. During processing the itemset {B, C, E}, transactions $T_4$ and $T_6$ are chosen as the victim transactions, respectively. Meanwhile, the victim item C is deleted from both transactions to hide the current sensitive itemset, whereas the utility of itemset {A, B, C, D E} is reduced to 0 as it no longer exists in transaction $T_4$. In the same manner, itemset {A, B, E} is concealed by removing its victim item B from the victim transaction $T_4$. After the sanitization, only the non-sensitive itemset {A, B} is over-hidden.

## 6. Experimental evaluation

To assess the efficiency of the proposed SMRF, SLRF and SDIF algorithms, multiple intensive experiments were conducted on four benchmark datasets, which are commonly used for performance evaluation in the pattern mining field. These datasets were downloaded from the SPMF[1] website (Fournier-Viger et al., 2014), which is an open-source data mining library. Table 9 summarizes the features of the adopted datasets. These datasets were picked specifically since they have different natures (dense, sparse, large and long transactions) and hence represent well the major types of data seen in real-world applications. The features in Table 9 include for every dataset: the dataset name, the number of transactions, the number of distinct items, the average number of items in each transaction, the maximum length of transactions, the type, the density percentage, and a short remark, respectively. To recognize the dense and sparse datasets, the density was calculated by dividing the average transaction length of the dataset by the number of distinct items in it. Datasets having a density percentage greater than 5% are classified as dense. Chess is a very dense dataset with almost 50% density. Mushroom is also dense with more distinct items than chess. Accidents dataset is the largest dense dataset as it contains 340,183 transactions with an average of 33.8 items per transaction. On the contrary, retail dataset has a very small density percentage since it is composed of a relatively large number of items and short transactions.

The overall performance of the proposed algorithms was evaluated against the following baseline and state-of-the-art algorithms:

[1] https://www.philippe-fournier-viger.com/spmf/

**Table 9**
The characteristics of the studied datasets.

| Dataset | #Trans | #Items | Avg. length | Max. length | Type | Density(%) | Remarks |
|---|---|---|---|---|---|---|---|
| chess | 3196 | 75 | 37 | 37 | Dense | 49.3 | Legal moves of a chess game |
| mushroom | 8124 | 119 | 23 | 23 | Dense | 19.3 | Information about various mushrooms species |
| accidents | 340,183 | 468 | 33.8 | 51 | Large | 7.2 | Anonymized traffic-accident data for the period 19912000 |
| retail | 88,162 | 16,470 | 10.3 | 76 | Sparse | 0.062 | Customer transactions from an anonymous Belgian retail store |

**Table 10**
The parameter settings of the benchmark datasets.

| chess | | mushroom | | accidents | | retail | |
|---|---|---|---|---|---|---|---|
| #SHIs | MU(%) | #SHIs | MU(%) | #SHIs | MU(%) | #SHIs | MU(%) |
| varied | 21 | varied | 7 | varied | 1.2 | varied | 0.025 |
| 100 | varied | 50 | varied | 15 | varied | 50 | varied |

HHUIF (Yeh and Hsu, 2010), MSICF (Yeh and Hsu, 2010), MSU-MIU (Lin et al., 2016), MSU-MAU (Lin et al., 2016), SMAU (Liu et al., 2020b), and *Weighted_RoT_DoC* (Jangra and Toshniwal, 2022), including their performance when varying (1) the minUtil value and (2) the number of sensitive itemsets. During the experiments, the high utility itemsets are first generated by applying the EFIM (Zida et al., 2015) algorithm. After that, the sensitive itemsets are selected randomly and while considering the standards of literature (Yun and Kim, 2015). Roughly, a sensitive itemset should mainly satisfy two conditions: (1) it is a high utility itemset; and (2) it consists of two or more items.

All codes were implemented in java, and the experiments were executed on a computer equipped with a third-generation Core-i7 2.1 GHz intel processor, 6 GB memory, and running Windows 7 OS. During the experiments, as can be seen in Table 10, we set the minUtil (MU) and the number of sensitive itemsets (SHIs) such that each hiding task has an appropriate and measurable runtime.

### 6.1. Efficiency evaluation metrics

Seven validation metrics that are widely used in PPUM literature (Jangra and Toshniwal, 2022; Lin et al., 2016; Yeh and Hsu, 2010) are proposed for the experiments. They are as follows:

1) **Hiding Failure (HF).** This metric aims to evaluate the proportion of sensitive itemsets that still exist in the dataset even after the sanitization process has ended. The HF percentage is computed as follows:

$$HF = \frac{|\ SHIs'\ |}{|\ SHIs\ |} \tag{3}$$

where $|\ SHIs\ |$ and $|\ SHIs'\ |$ refer to the number of sensitive itemsets before and after the sanitization process, respectively. The ideal value of HF is zero, which indicates that all sensitive itemsets have been hidden after the sanitization process.

2) **Artificial Cost (AC).** This metric aims to evaluate the proportion of fake high utility itemsets that have been produced as a result of the distortion process of PPUM. The AC percentage is computed as follows:

$$AC = \frac{|\ HUIs - HUIs'\ |}{|\ HUIs'\ |} \tag{4}$$

where $|\ HUIs\ |$ and $|\ HUIs'\ |$ refer to the number of high utility itemsets before and after the sanitization process, respectively. The ideal value of AC is zero, which indicates that no fake patterns have been generated.

3) **Missing Cost (MC).** This metric aims to evaluate the proportion of non-sensitive itemsets that have been lost after the sanitization process. The MC percentage is computed as follows:

$$MC = \frac{|\ NHIs'\ |}{|\ NHIs\ |} \tag{5}$$

where $|\ NHIs\ |$ and $|\ NHIs'\ |$ refer to the number of non-sensitive itemsets before and after the sanitization process, respectively. The ideal value of MC is zero, which indicates that all non-sensitive itemsets still exist in the sanitized dataset.

4) **Itemset Utility Similarity (IUS).** This metric reflects the lack in the total utility of the original high utility itemsets as a consequence of the distortion process of PPUM. The IUS percentage is computed as follows:

$$IUS = \frac{\sum_{Y \in HUIs'} u(Y)}{\sum_{Y \in HUIs} u(Y)}. \tag{6}$$

where *HUIs* and *HUIs'* refer to the produced high utility itemsets before and after the sanitization process, respectively. In general, as the missing cost increases, the IUS decreases.

5) **Dataset Utility Similarity (DUS).** This metric indicates the lack in the total utility of the dataset as a consequence of the distortion process of PPUM. The DUS percentage is computed as follows:

$$DUS = \frac{\sum_{T_d \in D'} TU(T_d)}{\sum_{T_d \in D} TU(T_d)}. \tag{7}$$

where $D$ and $D'$ refer to the dataset before and after the sanitization process respectively, and $TU(T_d)$ refers to the total utility of transaction $T_d$. Clearly, the more the value of the DUS, the better the quality of the sanitized dataset.

6) **Transactions Modification Ratio (TMR).** This metric indicates the percentage of transactions that have been distorted during the sanitization procedure. The TMR can be computed as next:

$$TMR = \frac{\#\ \text{modified transactions}}{\text{total}\ \#\ \text{transactions}}. \tag{8}$$

7) **Runtime (RT).** The processing time is an important metric for evaluating the efficiency of any PPDM algorithm. In PPUM, the hiding time can fluctuate depending on multiple factors such as (1) the dataset density (dense, sparse), (2) the dataset size (large, small), (3) the number of itemsets that are required to be hidden, and (4) the number of dataset scans required to complete the sanitizing process.

Since all the compared algorithms follow the item-based sanitization model and can successfully hide all the sensitive high utility itemsets without generating any false patterns, we find it not reasonable to consider the Hiding Failure and Artificial Cost metrics in the evaluation, and thus the results of these two metrics are not shown in this paper.

### 6.2. Runtime

In this section, we assess the required time to complete the hiding process in all the compared algorithms. Figs. 3 and 4 describe the runtime comparison between the three proposed algorithms and the other algorithms while varying the number of

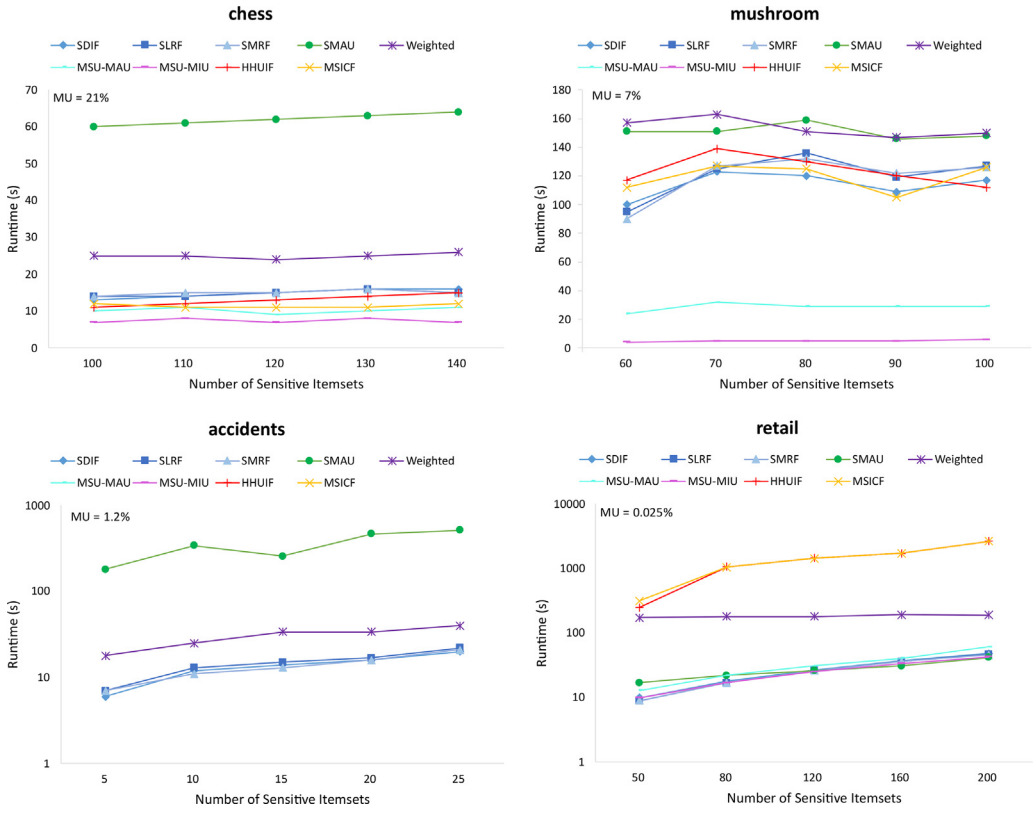


**Fig. 3.** Comparison of the runtimes using various number of sensitive itemsets.

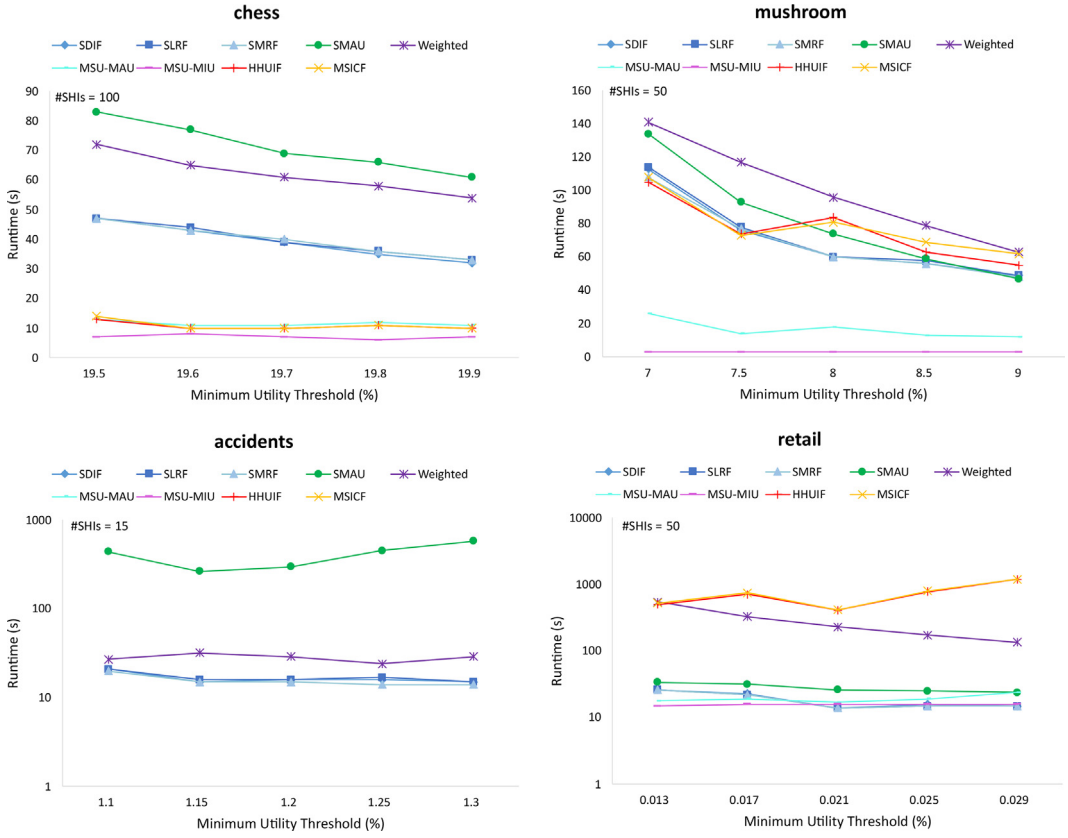


**Fig. 4.** Comparison of the runtimes using various minimum utility thresholds.

sensitive itemsets and the minUtil values, respectively. As can be observed in Fig. 3, the runtime predominantly tends to increase as the number of sensitive itemsets increases. This is because the more itemsets required to be hidden the more time is needed to complete the sanitization process. This seems to be the opposite in Fig. 4 because as the minUtil threshold increases the number of generated high utility itemsets decreases and thus less number of non-sensitive itemsets needs to be considered. From the results in Figs. 3 and 4, it can be noticed that the runtime of the suggested algorithms can fluctuate depending on the dataset type, size, and the itemsets that were chosen for sanitization. We can also confirm that the runtime efficiency of the proposed algorithms is the best compared to the most recent PPUM algorithms (SMAU and Weighted) in all the studied datasets. The reasons are as follows. SMAU requires more time to build and update its expensive data structure, while the Weighted algorithm requires more time for its double-sorting technique. In the large dataset, accidents, the proposed algorithms displayed the fastest runtime compared to the latest PPUM algorithms. As we could not complete the runs of the baseline algorithms; HHUIF, MSICF, MSU-MAU, and MSU-MIU in accidents dataset, their results were not recorded in this dataset. This in turn confirms that our proposed PPUM algorithms can indeed be run for large-scale datasets efficiently. In chess dataset, the proposed algorithms were found to be almost two times faster than the Weighted algorithm and six times faster than SMAU. In the highly sparse dataset, retail, the proposed algorithms showed almost similar performance to SMAU, MSU-MIU and MSU-MAU algorithms, and far better performance than HHUIF, MSICF and Weighted algorithms. This is because HHUIF and MSICF algorithms have to scan the dataset multiple times to complete the hiding process, whereas the Weighted algorithm spends more time in double sorting the sensitive transactions. The performance gap between our algorithms and the most recent PPUM algorithms is moderately obvious in mushroom dataset. However, the baseline algorithms, MSU-MAU and MSU-MIU, performed better than the proposed algorithms in mushroom dataset since they choose the victim items and transactions based on the concept of utility only. In brief, the proposed algorithms are simply fast due to the following reasons. First, they require only one dataset scan to identify the sensitive transactions and estimate the RISU of the sensitive items. Second, by taking advantage of the knowledge represented by the RISU values, they are able to specify an ideal victim item for each sensitive itemset, and therefore no need to re-evaluate and select a victim item multiple times in each iteration of the sanitizing process. Third, the proposed algorithms adopt the Weighted Sorting technique to obviate the need to double-sort the transactions or re-evaluate the sensitive transactions in each iteration to find a proper victim transaction.

### 6.3. Missing cost

Losing non-sensitive itemsets is common and almost unavoidable in PPUM due to items overlapping between the sensitive and non-sensitive itemsets. Hence, one of the crucial challenges in PPUM is to conserve the Legitimate knowledge of the original dataset during the sanitization process. Motivated by this, the proposed algorithms adopt different strategies when deciding which victim item to choose in favor of lessening the number of wasted innocent patterns. According to the results in Figs. 5 and 6, it can be claimed that the proposed SDIF and SLRF algorithms guarantee the best performance in terms of decreasing the missing cost compared to the other competitors in all the studied datasets. The results in Figs. 5 and 6 also indicate that the proposed SDIF has even more ability to maintain the non-sensitive itemset than the proposed SLRF in high-density datasets, where the overlapping degree between the sensitive and non-sensitive itemsets is very high. This can be observed in chess dataset; when the number of sensitive itemsets was 120, the missing cost of SDIF was about 4% less than that of SLRF, and when the minUtil was 19.6%, the missing cost of SDIF was about 5% less than SLRF. The main reason behind this is that the SDIF algorithm pays great attention to the non-sensitive HUIs while choosing the victim items, thus it loses less non-sensitive knowledge than the SLRF algorithm. It is interesting to observe that the proposed SMRF algorithm has relatively close performance to SMAU algorithm in almost all datasets. This is because SMAU algorithm chooses the item with the maximum utility as the victim item, whereas SMRF chooses the item with the maximum RISU value as the ideal victim item. Also, a noticeable result is that the missing cost of the baseline algorithms was found to be very high in chess dataset. This shows the weakness of these algorithms in datasets having a high overlapping degree among the sensitive and non-sensitive itemsets. In mushroom dataset, the SLRF algorithm showed superior performance compared to the other contenders. This is because in SLRF algorithm the item with the minimum RISU value is selected for sanitization, thus the resultant damage on the non-sensitive itemsets is also minimized. In conclusion, the missing cost results corroborate the intuition that utilizing the RISU values, which reflect the influence of each sensitive item on all sensitive transactions, can lead to a proper sanitization process. Moreover, the proposed Weighted Sorting technique, which considers the sensitive and non-sensitive covers of transactions, can significantly reduce the missing cost and make the proposed algorithms even more efficient.

### 6.4. Itemset utility similarity

In this section, the IUS values of the compared nine algorithms are analyzed under a varying number of sensitive itemsets and minUtil values. Unlike the missing cost, the IUS measure pertains to the loss in the total utility of HUIs rather than the loss in the number of the non-sensitive HUIs. There is thus an inverse relationship between the MC and the IUS, in which increasing one of them decreases the other. As shown in Figs. 7 and 8, the proposed SDIF and SLRF algorithms have outstanding performance compared to the baseline and the latest PPUM algorithms. This is something expected to see since both SDIF and SLRF algorithms have lower missing cost than the other competitors. We can also observe that MSU-MIU and Weighted algorithms outperform the proposed SMRF in mushroom dataset. The reason is that in the victim item selection process of SMRF, the item with the greatest RISU value is selected as the ideal victim item. However, in retail dataset, the SMRF algorithm showed optimistic performance as the overlapping degree between items is less in sparse datasets. In short, the IUS results validate that our adopted ideas can effectively mitigate the negative impacts incurred by the sanitization process.

### 6.5. Dataset utility similarity

This section deliberates on the negative influence of the proposed sanitization algorithms on the total utility of the dataset. Figs. 9 and 10 illustrate the results of the DUS metric. As can be seen, the proposed SLRF and SDIF algorithms yield superior performance with respect to the DUS among the compared algorithms. However, there is an exception when it comes to the MSU-MIU algorithm since it offers slightly better results than SLRF and SDIF. The reason why MSU-MIU consumes less dataset utility is that it chooses the item with the minimal utility as the victim item, and thus less utility is reduced from the total utility of the dataset. Apart from this, the SLRF algorithm also outperforms the SDIF algorithm as the item with the least the RISU value is chosen directly for sanitization. In contrast, the performance of HHUIF, SMAU, MSU-MAU, and SMRF algorithms is always the worst. These

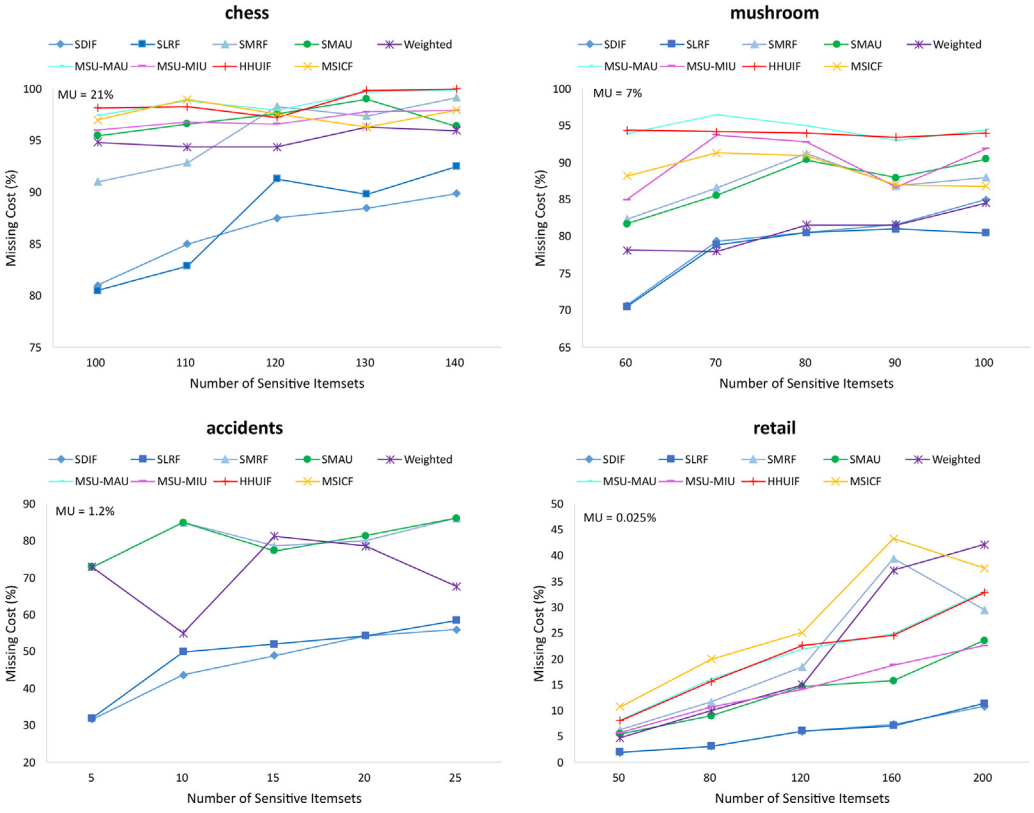


**Fig. 5.** Missing cost using various number of sensitive itemsets.

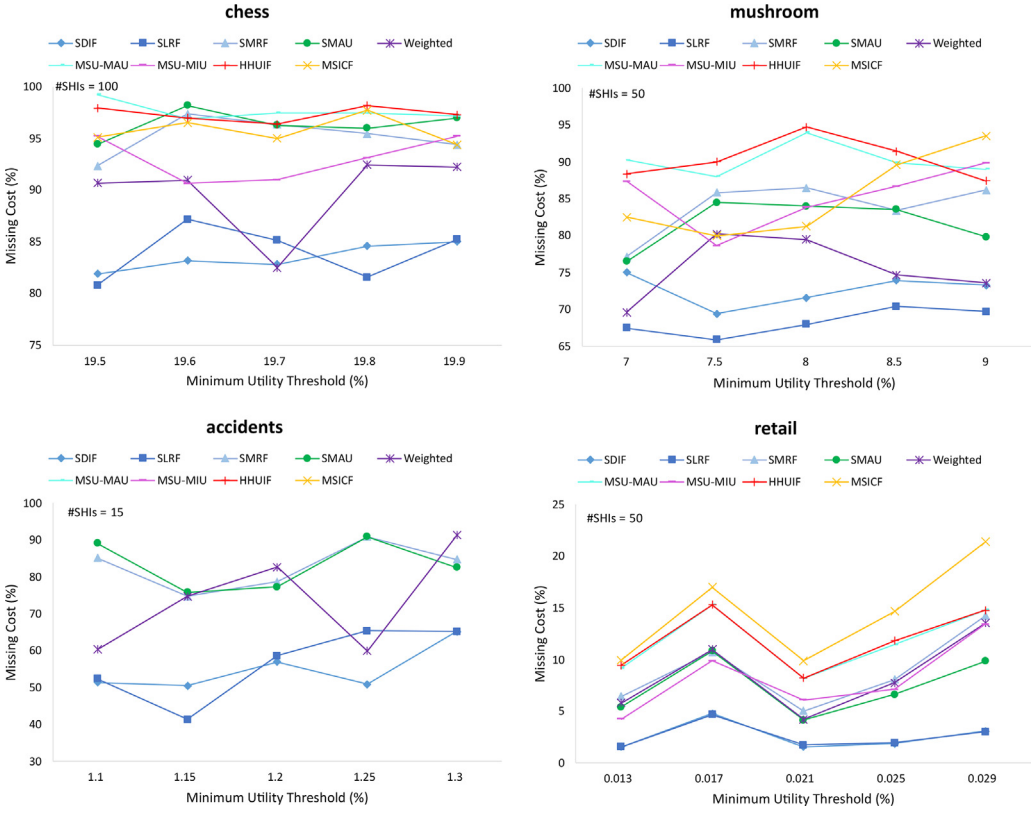


**Fig. 6.** Missing cost using various minimum utility thresholds.

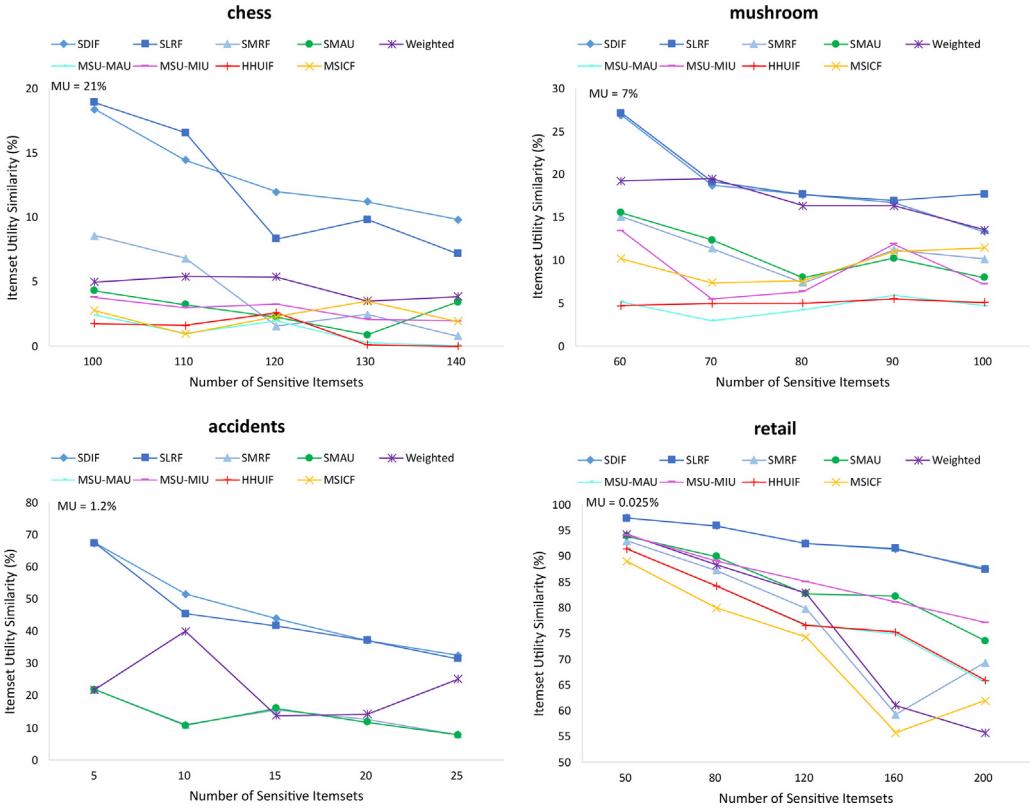


**Fig. 7.** IUS using various number of sensitive itemsets.

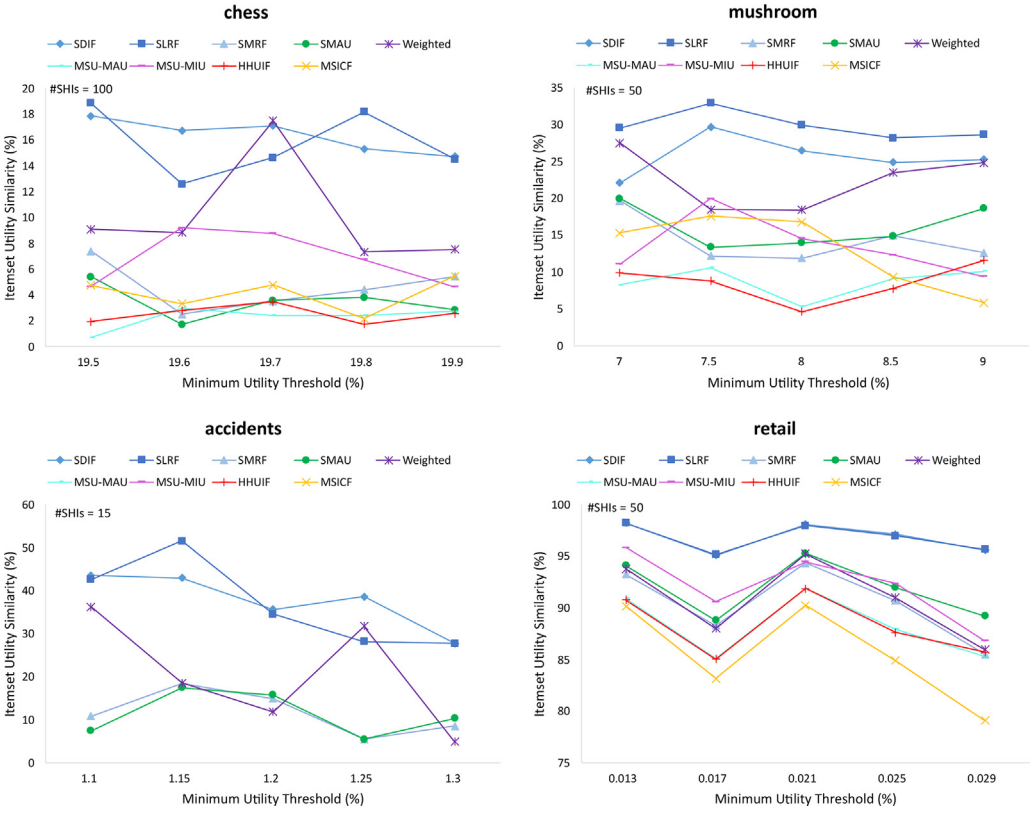


**Fig. 8.** IUS using various minimum utility thresholds.

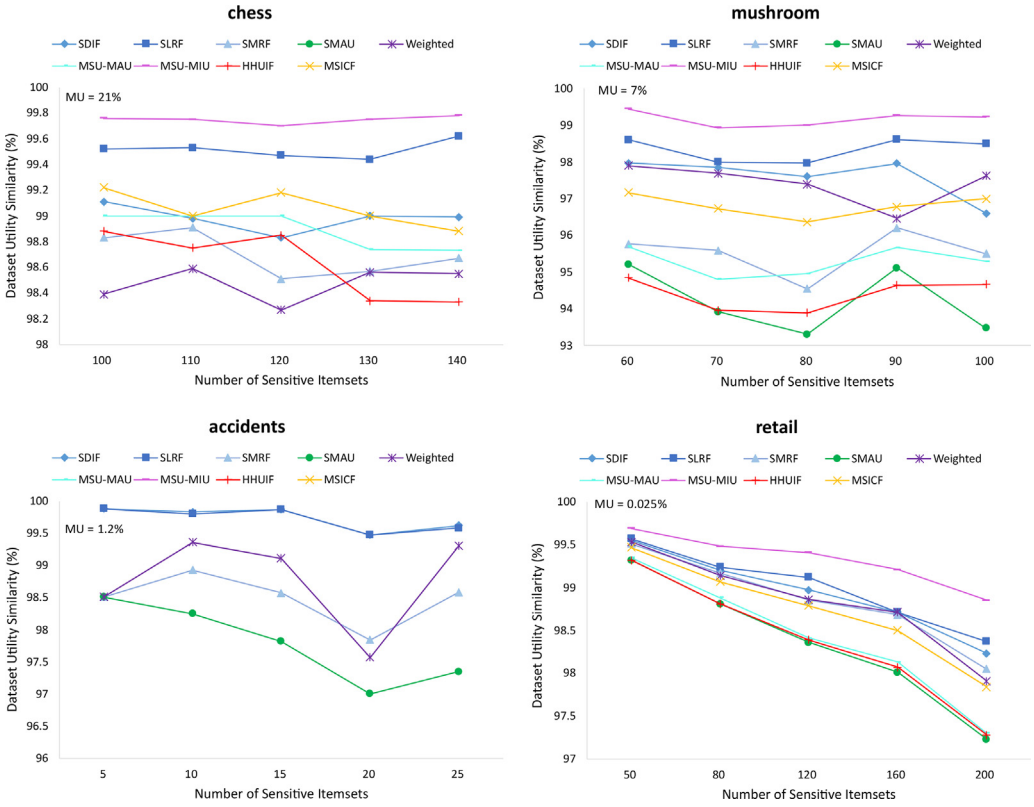


**Fig. 9.** DUS using various number of sensitive itemsets.

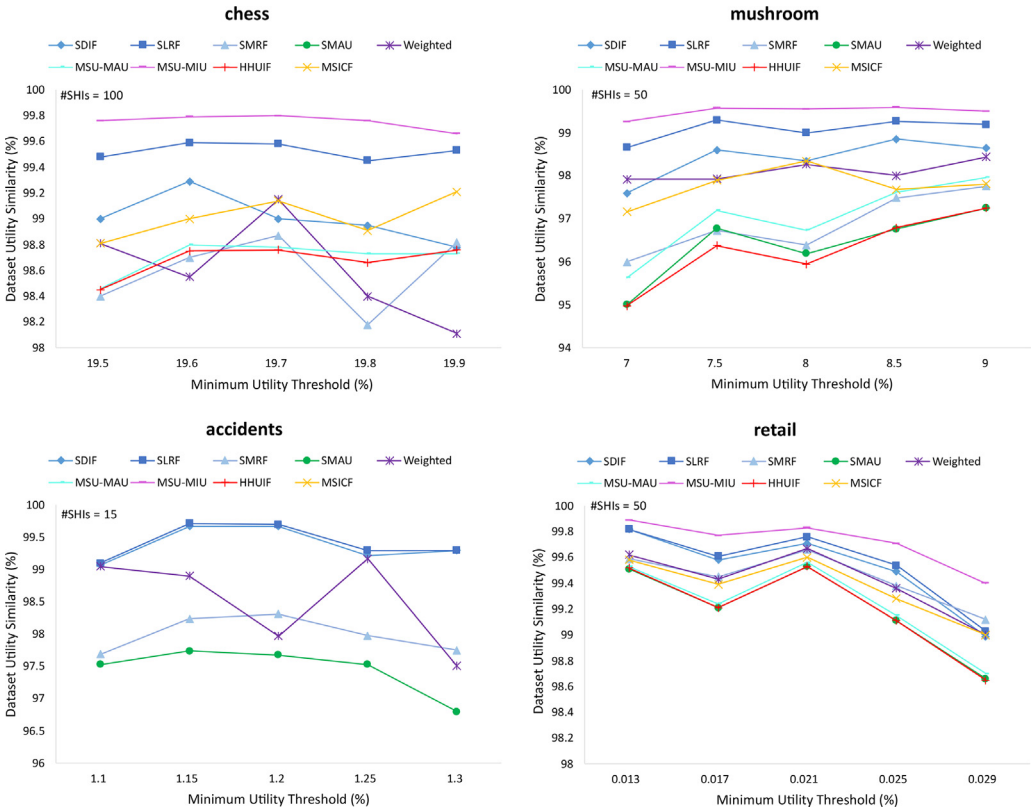


**Fig. 10.** DUS using various minimum utility thresholds.

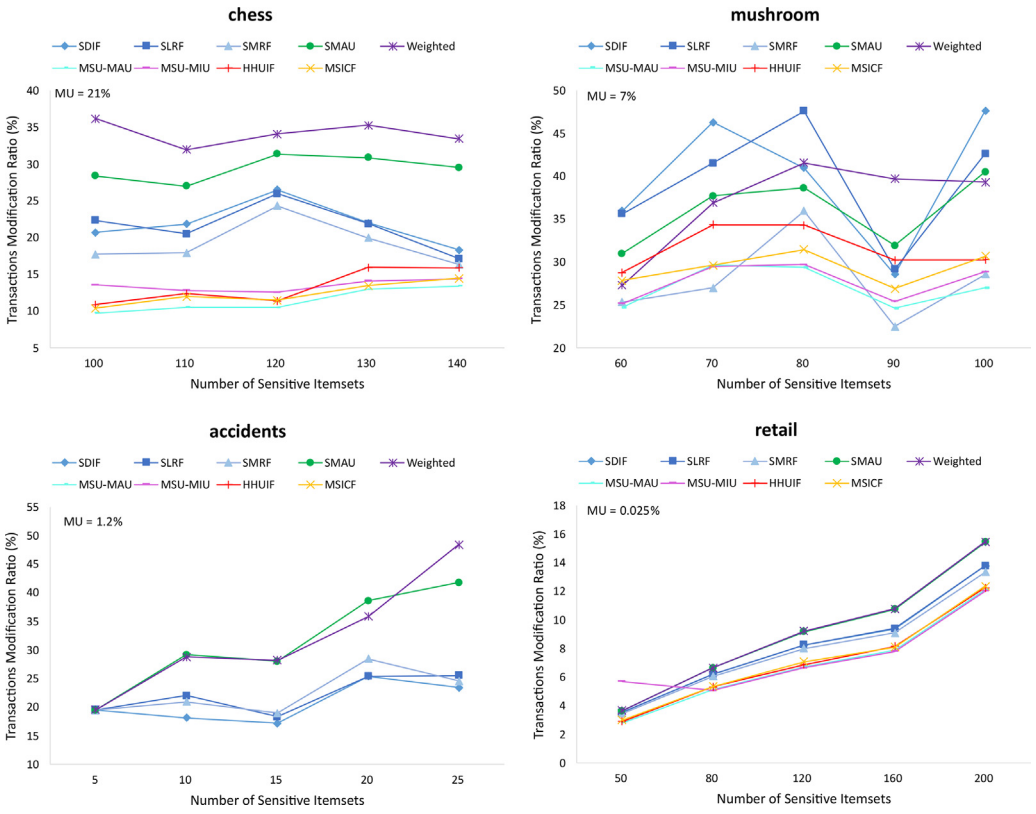


**Fig. 11.** TMR using various number of sensitive itemsets.

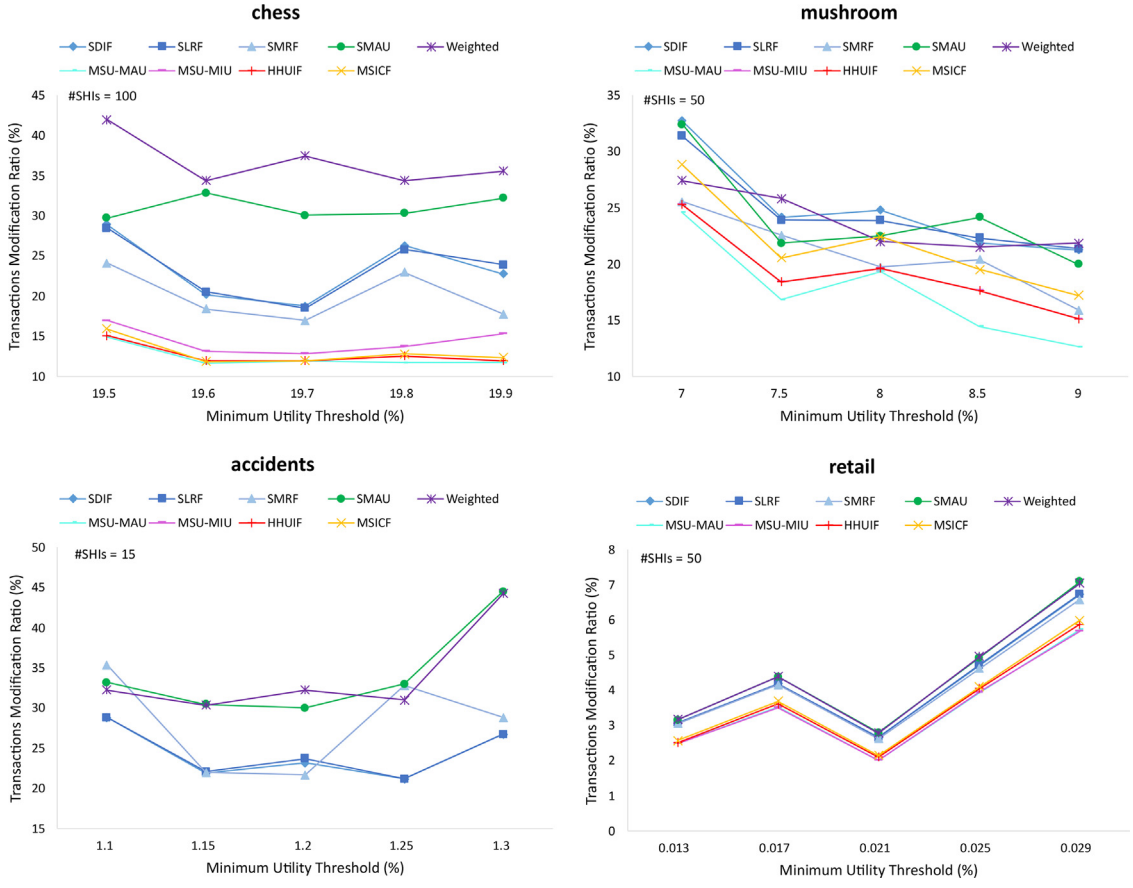


**Fig. 12.** TMR using various minimum utility thresholds.

performance trends are reasonable, given that the item with the maximum utility or maximum RISU value is always chosen for sanitization in these algorithms. Nevertheless, this is not the case for the MSICF and Weighted algorithms as the first considers the overlapping degree of items when choosing the victim item, while the second adopts utility-independent criteria to evaluate the candidate victim items. In a word, even though the SDIF algorithm does not obtain the best values of DUS, it tends to perform better than the other proposed algorithms, SMRF and SLRF, in the MC and IUS. This is because the SDIF algorithm, unlike SLRF algorithm, prefers the items with the least non-sensitive cover to the ones with the least RISU value. Also, we can learn that the SMRF algorithm tends to produce the worst results when it comes to the MC, IUS and DUS metrics. The cause is that it removes the item with the maximum RISU value, which leads to less DUS and IUS, and more MC.

### 6.6. Transactions modification ratio

Last but not least, it is interesting to see how the adopted ideas affected the number of transactions required to be modified during the data distortion process. Figs. 11 and 12 present the percentages of transactions modified as a result of removing the sensitive itemsets. These results reveal a few interesting insights. First, the proposed SMRF algorithm tends to have a less modification ratio than the other two proposed algorithms. The reason is that in SDIF and SLRF algorithms, the item with the least total sensitive utility or RISU value is chosen for sanitization, thus more transactions need to be modified to hide the sensitive itemset. Second, the TMR of all the compared algorithms is relatively higher in dense and large datasets than in the sparse dataset (retail). This is because in sparse datasets the frequency of the itemsets is usually low, thus removing the victim item from a few transactions can significantly reduce the utility of the sensitive itemsets. Third, the baseline algorithms always require a fewer number of modifications compared to the proposed algorithms. This is mainly due to the fact that the proposed algorithms treat the importance of transactions differently as the sensitive and non-sensitive covers of transactions are leveraged to select the best modifications. Conversely, the baseline algorithms follow utility-based criteria in their transaction selection method. Apparently, the proposed algorithms often require less TMR than the state-of-the-art algorithms (SMAU and Weighted). This result is reached thanks to the efficient use of the RISU values and the Weighted Sorting technique. In general, the previous results render that the selection method of transactions along with the density of the dataset play a key role in determining the transactions modification ratio.

## 7. Conclusion

Although identifying high utility patterns leads to valuable insights, past findings agree that utility-driven pattern mining can be a double-edged weapon. While on the one hand it can be used to discover vast and rich knowledge, it can on the other hand disclose confidential information about data owners by multiple parties. Without adequate privacy protection, many data suppliers can be conservative about sharing their data. Therefore, Privacy Preserving Utility Mining (PPUM) algorithms are used to protect the private information from utility-driven techniques. In this paper, we developed three PPUM algorithms, namely, SMRF, SLRF, and SDIF, to efficiently remove the sensitive high utility itemsets based on the concept of the Real Item Sensitive Utility (RISU). The three algorithms utilize a sorting technique for the sensitive transactions, named Weighted Sorting, to give the fewer side-effects transactions higher priority for sanitization. The concept of RISU is applied in the victim item selection process to find an ideal victim item for each sensitive high utility itemset in the initial stages of the sanitization process. To validate the viability of the proposed algorithms, extensive comparisons were conducted on four benchmark datasets using five performance measures. The obtained results indicate that the proposed algorithms advance the state-of-the-art algorithms in preserving the quality of the dataset after the sanitization process. For future work, we will focus on how to further support the detection of ideal victim items in the sensitive itemsets. Additionally, the practical application of PPUM in real-life situations is also worthy of attention. Lastly, the design of a parallel model for the task of PPUM in transactional datasets is interesting and challenging too.

## Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## CRediT authorship contribution statement

**Mohamed Ashraf:** Conceptualization, Methodology, Software, Visualization, Investigation, Writing – original draft, Writing – review & editing. **Sherine Rady:** Validation, Writing – review & editing. **Tamer Abdelkader:** Validation, Writing – review & editing. **Tarek F. Gharib:** Validation, Writing – review & editing, Supervision.

## Data availability

Data will be made available on request.

## References


Agrawal, R., Srikant, R., et al., 1994. Fast algorithms for mining association rules. In: Proceedings of 20th International Conference on Very Large Data Bases, Vol. 1215. Citeseer, pp. 487–499.

Ali, M.A., Rady, S., Abdelkader, T., Gharib, T.F., 2023. An efficient hiding method for privacy preserving utility mining. Int. J. Intell. Comput.Inf. Sci. 23 (1), 69–83.

Ashraf, M., Abdelkader, T., Rady, S., Gharib, T.F., 2021. TKN: an efficient approach for discovering top-k high utility itemsets with positive or negative profits. Inf. Sci. (Ny).

Ashraf, M., Rady, S., Abdelkader, T., Gharib, T.F., 2022. A robust privacy preserving approach for sanitizing transaction databases from sensitive high utility patterns. In: Proceedings of the 8th International Conference on Advanced Intelligent Systems and Informatics 2022. Springer, pp. 381–394.

Bandil, L., Soni, R., Rathi, S., 2018. A new method to preserve privacy of utility item sets using differential privacy. In: Proceedings of International Conference on Recent Advancement on Computer and Communication. Springer, pp. 481–487.

Chen, J., Guo, X., Gan, W., Chen, C.-M., Ding, W., Chen, G., 2022. On-shelf utility mining from transaction database. Eng. Appl. Artif. Intell. 107, 104516.

Dinh, D.-T., Huynh, V.-N., Le, B., Fournier-Viger, P., Huynh, U., Nguyen, Q.-M., 2019. A survey of privacy preserving utility mining. In: High-Utility Pattern Mining: Theory, Algorithms and Applications, pp. 207–232.

Fournier-Viger, P., Gan, W., Wu, Y., Nouioua, M., Song, W., Truong, T., Duong, H., 2022. Pattern mining: current challenges and opportunities. In: Database Systems for Advanced Applications. DASFAA 2022 International Workshops: BDMS, BDQM, GDMA, IWBT, MAQTDS, and PMBD, Virtual Event, April 11–14, 2022, Proceedings. Springer, pp. 34–49.

Fournier-Viger, P., Gomariz, A., Gueniche, T., Soltani, A., Wu, C.-W., Tseng, V.S., et al., 2014. SPMF: a java open-source pattern mining library. J. Mach. Learn. Res. 15 (1), 3389–3393.

Fournier-Viger, P., Li, J., Lin, J.C.-W., Chi, T.T., Kiran, R.U., 2020. Mining cost-effective patterns in event logs. Knowl. Based Syst. 191, 105241.

Gan, W., Chun-Wei, J., Chao, H.-C., Wang, S.-L., Philip, S.Y., 2018. Privacy preserving utility mining: a survey. In: 2018 IEEE International Conference on Big Data (Big Data). IEEE, pp. 2617–2626.

Gan, W., Lin, J.C.-W., Fournier-Viger, P., Chao, H.-C., Tseng, V.S., Philip, S.Y., 2019. A survey of utility-oriented pattern mining. IEEE Trans. Knowl. Data Eng. 33 (4), 1306–1327.

Grätzer, G., 2011. Lattice Theory: Foundation. Springer Science & Business Media.

Holland, J.H., 1992. Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence. MIT press.

Huynh, U., Le, B., Dinh, D.-T., Fujita, H., 2022. Multi-core parallel algorithms for hiding high-utility sequential patterns. Knowl. Based Syst. 237, 107793.

Jangra, S., Toshniwal, D., 2022. Efficient algorithms for victim item selection in privacy-preserving utility mining. Future Gener. Comput. Syst. 128, 219–234. doi:10.1016/j.future.2021.10.008.

Jisna, J., Salim, A., 2018. Privacy preserving data utility mining using perturbation. In: International Conference on Distributed Computing and Internet Technology. Springer, pp. 112–120.
Kenthapadi, K., Mironov, I., Thakurta, A.G., 2019. Privacy-preserving data mining in industry. In: Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining, pp. 840–841.
Krishna, G.J., Ravi, V., 2021. High utility itemset mining using binary differential evolution: an application to customer segmentation. Expert Syst. Appl. 181, 115122.
Le, B., Dinh, D.-T., Huynh, V.-N., Nguyen, Q.-M., Fournier-Viger, P., 2018. An efficient algorithm for hiding high utility sequential patterns. Int. J. Approximate Reasoning 95, 77–92. doi:10.1016/j.ijar.2018.01.005.
Li, S., Mu, N., Le, J., Liao, X., 2019. A novel algorithm for privacy preserving utility mining based on integer linear programming. Eng. Appl. Artif. Intell. 81, 300–312. doi:10.1016/j.engappai.2018.12.006.
Lin, C.-W., Hong, T.-P., Wong, J.-W., Lan, G.-C., Lin, W.-Y., 2014. A GA-based approach to hide sensitive high utility itemsets. Sci. World J. 2014.
Lin, J.C.-W., Djenouri, Y., Srivastava, G., Fourier-Viger, P., 2022. Efficient evolutionary computation model of closed high-utility itemset mining. Appl. Intell. 1–13.
Lin, J.C.-W., Hong, T.-P., Fournier-Viger, P., Liu, Q., Wong, J.-W., Zhan, J., 2017. Efficient hiding of confidential high-utility itemsets with minimal side effects. J. Exp. Theor. Artif.Intell. 29 (6), 1225–1245.
Lin, J.C.-W., Wu, T.-Y., Fournier-Viger, P., Lin, G., Zhan, J., Voznak, M., 2016. Fast algorithms for hiding sensitive high-utility itemsets in privacy-preserving utility mining. Eng. Appl. Artif. Intell. 55, 269–284. doi:10.1016/j.engappai.2016.07.003.
Liu, C., Guo, C., 2021. Mining top-n high-utility operation patterns for taxi drivers. Expert Syst Appl 170, 114546.
Liu, X., Chen, G., Wen, S., Song, G., 2020. An improved sanitization algorithm in privacy-preserving utility mining. Math. Probl. Eng. 2020.
Liu, X., Wen, S., Zuo, W., 2020. Effective sanitization approaches to protect sensitive knowledge in high-utility itemset mining. Appl. Intell. 50 (1), 169–191.
Liu, X., Xu, F., Lv, X., 2018. A novel approach for hiding sensitive utility and frequent itemsets. Intell. Data Anal. 22 (6), 1259–1278.
Liu, Y., Liao, W.-k., Choudhary, A., 2005. A fast high utility itemsets mining algorithm. In: Proceedings of the 1st International Workshop on Utility-Based Data Mining, pp. 90–99.
Mendes, R., Vilela, J.P., 2017. Privacy-preserving data mining: methods, metrics, and applications. IEEE Access 5, 10562–10582.
Qu, J.-F., Fournier-Viger, P., Liu, M., Hang, B., Wang, F., 2020. Mining high utility itemsets using extended chain structure and utility machine. Knowl. Based Syst. 208, 106457.
Rajalaxmi, R., Natarajan, A., 2012. Effective sanitization approaches to hide sensitive utility and frequent itemsets. Intell. Data Anal. 16 (6), 933–951.
Segura-Delgado, A., Anguita-Ruiz, A., Alcalá, R., Alcalá-Fdez, J., 2022. Mining high average-utility sequential rules to identify high-utility gene expression sequences in longitudinal human studies. Expert Syst. Appl. 116411.
Selvaraj, R., Kuthadi, V.M., 2013. A modified hiding high utility item first algorithm (HHUIF) with item selector (MHIS) for hiding sensitive itemsets. J. Innov. Comput. Inf.Control 9 (12), 4851–4862.
Tran, H.-Y., Hu, J., 2019. Privacy-preserving big data analytics a comprehensive survey. J. Parallel Distrib. Comput. 134, 207–218.
Trieu, V.H., Ngoc, C.T., Le Quoc, H., Si, N.N., 2018. Algorithm for hiding high utility sensitive association rule based on intersection lattice. In: 2018 1st International Conference on Multimedia Analysis and Pattern Recognition (MAPR). IEEE, pp. 1–6.
Tseng, V.S., Shie, B.-E., Wu, C.-W., Philip, S.Y., 2012. Efficient algorithms for mining high utility itemsets from transactional databases. IEEE Trans. Knowl. Data Eng. 25 (8), 1772–1786.
Tung, N., Nguyen, L.T., Nguyen, T.D., Vo, B., 2021. An efficient method for mining multi-level high utility itemsets. Appl. Intell. 1–22.
Verma, A., Dawar, S., Kumar, R., Navathe, S., Goyal, V., 2021. High-utility and diverse itemset mining. Appl. Intell. 1–15.
Yeh, J.-S., Hsu, P.-C., 2010. HHUIF and MSICF: novel algorithms for privacy preserving utility mining. Expert Syst. Appl. 37 (7), 4779–4786. doi:10.1016/j.eswa.2009.12.038.
Yun, U., Kim, J., 2015. A fast perturbation algorithm using tree structure for privacy preserving utility mining. Expert Syst. Appl. 42 (3), 1149–1165. doi:10.1016/j.eswa.2014.08.037.
Zhang, C., Han, M., Sun, R., Du, S., Shen, M., 2020. A survey of key technologies for high utility patterns mining. IEEE Access 8, 55798–55814.
Zida, S., Fournier-Viger, P., Lin, J.C.-W., Wu, C.-W., Tseng, V.S., 2015. EFIM: a highly efficient algorithm for high-utility itemset mining. In: Mexican International Conference on Artificial Intelligence. Springer, pp. 530–546.

**Mohamed Ashraf** received the BS and MS degrees in computer and information sciences from Faculty of Computer and Information Sciences (FCIS), Ain Shams University (ASU), Cairo, Egypt, in 2016 and 2022, respectively, where he is currently pursuing his PhD in computer and information sciences. From 2018 to 2022, he was a Teaching Assistant in Information Systems Department, FCIS, ASU. Since 2022, he has been an Assistant Lecturer. His research interests include data mining, machine learning, deep learning, software engineering and privacy preserving. His works have been published in prestigious and quality international journals such as Information Sciences. Email: mohamed.a.a.h.is@gmail.com

**Sherine Rady** received the BSc degree in electrical engineering (computer and systems) and the MSc degree in computer and information sciences from Ain Shams University, Cairo, Egypt, and the PhD degree from the University of Mannheim, Germany. She is currently a Professor with the Faculty of Computer and Information Sciences, Ain Shams University. Her research interests include Artificial Intelligence, Data Mining, and Data Science. Email: srady@cis.asu.edu.eg

**Tamer Abdelkader** received the BSc degree inel ectrical and computer engineering and the MSc degree in computer and information sciences from Ain Shams University, Cairo, Egypt, in 2003, and the MSc and PhD degrees in electrical and computer engineering from the University of Waterloo, Ontario, ON, Canada, in 2012. After graduation, he was with the University of Waterloo as a Postdoctoral Researcher and a Visiting Researcher. He worked as the Manager of the Information and Technology Research Consultancy Center, Ain Shams University, Cairo, Egypt. He also worked as an Information and Technological Consultant in several governmental and private companies, including the Information and Communication Technology Project, Egypt, and the Ministry of Electricity. He is currently an Associate Professor and Vice-Dean for Community Services and Environmental Affairs with the Faculty of Computer and Information Sciences, Ain Shams University. He is the author of several publications in IEEE Transactions and other ranked journals and conferences. His current research interests include network and cyber security, privacy preservation, delay-tolerant networks, resource allocation in wireless networks, and energy-efficient protocols. Email: tammabde@cis.asu.edu.eg

**Tarek F. Gharib** is currently a Full Professor of Information Systems at Ain Shams University, Cairo, Egypt. He received his PhD degree in Theoretical Physics from the University of Ain Shams in 1994. His research interests focus on developing novel data mining and machine learning techniques, especially for applications in text mining, social networks, Bioinformatics and Data Analytics. He has over 90 publications. He received the National Science Foundation Award in 2001. Email: tfgharib@cis.asu.due.eg