Version of Record: https://www.sciencedirect.com/science/article/pii/S0020025522007162
Manuscript_4a143d052e3514c8524735cea8bd92e8

H-FHAUI: Hiding Frequent High Average Utility Itemsets

Bac Le[a,b], Tin Truong[c], Hai Duong[c], Philippe Fournier-Viger[d,*], Hamido Fujita[e]
[a]Department of Computer Science, Faculty of Information Technology, University of Science, Ho Chi Minh city, Vietnam
[b]Vietnam National University, Ho Chi Minh city, Vietnam
lhbac@fit.hcmus.edu.vn
[c]Department of Mathematics and Computer Science, Dalat University, Dalat city, Vietnam
tintc@dlu.edu.vn,haidv@dlu.edu.vn
[d]School of Humanities and Social Sciences, Harbin Institute of Technology, Shenzhen, China
philfv@hit.edu.cn
[e]Faculty of Software and Information Science, Iwate Prefectural University, Iwate, Japan
hfujita-799@acm.org

Abstract. High average-utility itemset mining consists of analyzing a quantitative customer transactional database to identify high average-utility itemsets (HAUIs), that is sets of items that have a high average utility (e.g. profit). Although important information about customers' habits can be revealed by HAUIs, they can expose sensitive information. To address this concern, the problem of hiding frequent HAUIs (FHAUIs) is studied, which is to modify a transaction database to ensure that sensitive FHAUIs cannot be discovered. An algorithm is designed, named H-FHAUI, which relies on an extended border approach based on the support (occurrence frequency) and the average-utility of itemsets. Moreover, to hide all FHAUIs, H-FHAUI utilizes a novel extended lower border named Bd_E^- based on weak upper bounds on the average-utility to only hide a small number of FHAUIs. Then, all remaining FHAUIs are also hidden. Besides, H-FHAUI utilizes a novel weight-based strategy named ICS to choose items and transactions to be modified, a novel TIU-VIU structure to quickly update weak upper bounds, and a strategy named $\mathcal{DUS}_{WUB}$ to quickly hide FHAUIs while ensuring that the total utility of $\mathcal{D}$ is preserved as much as possible. Experimental results show that H-FHAUI outperforms a baseline in terms of runtime, memory usage, and quality of the sanitized database.

Keywords. Privacy-preserving utility mining; Pattern hiding; High average-utility itemset; Upper bound; Weak upper bound; Border approach.

1. Introduction

For more than two decades, data mining techniques have played an important role for analyzing databases. Numerous algorithms have been designed to identify novel, useful, and unexpected patterns in data, which can help to understand the data, support decision-making and provide insightful information about user preferences. However, a major issue is that knowledge discovered by these techniques can also reveal private, sensitive or strategic information such as credit card information, purchase patterns from individuals, and personal identification numbers. As a result, individuals may face privacy threats and their data may be misused. It is also important to protect private and sensitive information of businesses that give them a strategic edge over their competitors, and to protect the privacy of their employees and customers. For instance, if a company releases data publicly or shares data with collaborators, there is a risk that some sensitive information may be extracted from it using data mining algorithms.

To overcome this problem, Privacy-Preserving Data Mining (PPDM) has been proposed to ensure that sensitive information or patterns cannot be extracted from databases [1–3]. The goal of PPDM is to transform an original database so that private and sensitive knowledge cannot be discovered from it by data mining algorithms [5,12,30,32,41]. Many different methods have been proposed to hide sensitive association rules and frequent itemsets in binary databases [8,35,36,41]. But this simple database format is unsuitable for many applications. For example, binary databases are commonly used to represent customer transactions where each record is a transaction indicating that a set of items was purchased by a customer. Though useful, this model is limited as it does not allow to consider product quantities and the profit yield by each item.

To provide a more realistic model for representing customer transactions, the concept of Quantitative Transaction Database (QTDB) was introduced for the problem of high utility itemset mining (HUIM). HUIM is a generalisation of

frequent itemset mining [43], where the objective is to discover all itemsets (sets of items purchased by customers) that have a high utility. The utility is a measure of the importance of an itemset (e.g. in terms of profit) and is defined as the sum of the utilities of its items in transactions where it appears. An itemset is called a high utility itemset (HUI) and is said to be of high utility (HU), if its utility is no less than a user-defined minimum utility threshold, denoted as mu. To protect strategic information that may be revealed by HUIM algorithms, the PPDM task of hiding High Utility Itemsets (HUIs) was proposed [19,43,46]. It consists of hiding a set of sensitive HUIs among those found in a QTDB for a given mu value by modifying the original QTDB so that adversaries or competitors cannot discover them in the modified database for the same mu value. Few algorithms have been proposed for HUIM [4,14,27,39] and for hiding HUIs [15,19,29,33,43,46]. Most sensitive pattern hiding algorithms conceal HUIs by reducing the total utility (quality) of a QTDB or by deleting transactions from a QTDB.

Although HUIM has many real-life applications, it is a difficult task. A challenge is that the number of patterns that must be considered can be very large since some very long HUIs may appear in a QTDB, and many of their subsets may be HUIs. Another important issue is that the utility measure does not take the length of itemsets into account. But in practice, long patterns may be uninteresting to users because it is hard to promote several items together. Moreover, because long patterns often have a higher utility than shorter ones, there can be a bias towards finding longer patterns. To provide an alternative to HUIM that addresses these issues, researchers have proposed the problem of high average-utility itemset mining (HAUIM) [10], which relies on a different utility measure called the average-utility (au). Besides considering the unit profits and quantities of items in a given itemset, HAUIM also considers the number of items that each itemset contains. The au of an itemset is its utility divided by its length. An itemset is said to be a high average-utility itemset (HAUI) if its average-utility is no less than the user-predefined mu threshold. Numerous algorithms have been designed for HAUIM [9,13,18,20,22,23,37,38] such as VMHAUI [38].

HAUIM has applications in many domains. For instance, HAUIM can be used in a business context for cross-marketing and developing new promotional strategies to increase sales of highly profitable products [39], to analyse streaming data (e.g. analysing web-click streams based on the time spent on each webpage [31,45]), and to discover important gene patterns in medical data [47]. But HAUIs found in a QTDB may reveal private or strategic information, which can be problematic. For example, an e-commerce company may want to share data about its customer transactions with another company in the form of a QTDB for collaboration but may not want to reveal the most profitable patterns (HAUIs) that appear in the data so that the other company cannot use this information to its advantage. This is an important concern as data collected by companies about customers are especially hard to collect and valuable for various tasks such as product recommendation. Thus, it is desirable to have control over what can be found in data using HAUIM algorithms. A second example is the data collected from large search engines about search queries. A search query can be represented as a QTDB where each transaction is a set of keywords in a query and where the utility of keywords can be a measure of the importance of words (e.g. term frequency). As search query data is very valuable for businesses, it is also reasonable to hide important associations between keywords before releasing search query data publicly to keep a competitive advantage. Thus, as motivated by these examples, sharing a QTDB can result in privacy, security threats or profit losses. There is hence an obvious need for hiding sensitive HAUIs to prevent their discovery by unauthorized users.

To address this concern, this paper studies the problem of hiding frequent HAUIs (FHAUIs) in a QTDB. This problem consists of modifying the database to hide all FHAUIs for some given minimum utility and support thresholds, denoted as mu and ms. In this problem, an FHAUI is a high average-utility itemset whose frequency is no less than ms. The reason for considering not only the average-utility but also the frequency is that constraints on the frequency are also traditionally used in pattern mining to filter out noisy patterns (that may appear by chance or not be significant due to their low occurrence frequencies). For instance, if some customer behaviour is only observed a few times in a customer transaction database or if some keywords only appear a few times in search queries, those patterns may be ignored. Thus, hiding patterns by considering both factors (utility and frequency) is more useful than just considering one. To our best knowledge, no algorithm has been designed for the problem of frequent high average-utility itemset hiding (FHAUIH).

To address this research gap, a straightforward approach is to design an algorithm that hide all FHAUIs one after the other by successive modifications of the database, until all those FHAUIs have been hidden. Although this baseline approach can hide all FHAUIs, it is inefficient because the number of FHAUIs can be very large, especially for low mu and ms threshold values. Moreover, hidden patterns one by one may perform too many modifications of the input database. This paper addresses these issues by designing an efficient algorithm for FHAUIH named H-FHAUI. The algorithm uses a border approach inspired by previous work on frequent itemset hiding [34] where the anti-monotonicity ($\mathcal{AM}$) property of the support measure is used to reduce the search space. However, extending this border approach for the problem of FHAUIH is not trivial because the average-utility function does not satisfy the $\mathcal{AM}$ property. As a solution, this study proposes an extended lower border, which is applied on weak upper bounds on au and is integrated into the proposed H-FHAUI algorithm to efficiently hide FHAUIs by only hiding a small subset of FHAUIs. The major contributions of this paper are as follows.

1) We introduce an extended lower border based on weak upper bounds on the average-utility to find a small subset of FHAUIs that needs to be hidden to hide all FHAUIs.
2) Two novel strategies, named ICS and $\mathcal{DUS}_{WUB}$, are proposed to quickly hide all FHAUIs and to ensure that the quality of the original QTDB is minimally affected.
3) A novel structure named TIU-VIU (Transaction Itemset Utility-Vertical Item Utility) is proposed to quickly update values of weak upper bounds on au, which are regularly changed during the hiding process.
4) We design two novel algorithms, named H-FHAUI (Hiding Frequent High Average Utility Itemsets) and H-FHAUI-B (Hiding Frequent High Average Utility Itemsets - Baseline) to hide all FHAUIs. H-FHAUI uses the extended lower border to find a small subset of FHAUIs to be hidden. Moreover, it utilizes two strategies called ICS and $\mathcal{DUS}_{WUB}$ to reduce the amount of changes made to the original database. The H-FHAUI-B algorithm is a baseline based on the traditional approach of concealing all FHAUIs one by one and does not utilize the border approach and strategies.
5) Experimental results provide empirical evidences that the proposed H-FHAUI algorithm is much more efficient than the baseline H-FHAUI-B algorithm in terms of runtime, memory usage, and quality of the sanitized QTDB.

The rest of this paper is organized as follows. Section 2 gives an overview of related work on HAUIM and PPDM. Section 3 introduces preliminary concepts and notation for hiding FHAUIs. Section 4 presents novel theoretical results, including a baseline method for hiding FHAUIs and a border approach based on the support and average-utility. Moreover, that section also presents the extended lower border, two novel strategies ICS and $\mathcal{DUS}_{WUB}$, and the designed H-FHAUI algorithm. Section 5 describes the experimental evaluation. Finally, a conclusion is drawn and opportunities for future work are discussed in Section 6.

2. Related work

This section first reviews important related work on high average-utility itemset mining and then about hiding association rules, frequent itemsets, and high utility itemsets.

Mining high average-utility itemsets. The problem of HAUIM was proposed as an alternative to HUIM where the utility measure is replaced by the au measure that is considered fairer as it takes the length of each itemset into account. But a major challenge for designing efficient HAUIM algorithms is that the au of itemsets does not satisfy the downward closure property (DCP) commonly used in itemset mining to reduce the search space. In other words, if an itemset is not of high utility, its super itemsets can be HU itemsets, and thus they should not be pruned from the search space. Therefore, to allow search space pruning, numerous researchers have designed upper bounds (UBs) on the au that satisfy the DCP. Many UBs and weak upper bounds (WUB) have been proposed and used in HAUIM algorithms. Most of them are based on two types of database representations [38], namely the horizontal database representation (HDR) and the vertical database representation (VDR).

UBs based on a HDR are computed using the utilities of items in rows of the Q matrix representation of a database (see Definition 2 in the next section) and are thus called horizontal UBs. Many UBs based on this representation were proposed, such as the $auub$ UB used in the TPAU algorithm [9], the $lubau$ and $tubau$ UBs employed by the D-FHAUM algorithm [22], and the lub and $rtub$ which are integrated into the EHAUPM algorithm [21]. In addition, some *weak* upper bounds (WUBs) relying on HDR were also designed, such as mau in the MHAI algorithm [44] or $mfuub$ and $krtmuub$ in the TUB-HAUPM algorithm [42]. In this context, a measure wub is called a weak UB on au if $au(C) \leq wub(P)$ for any itemset P and any proper extension C of P such that $C \supset P$. Since the UBs or WUBs based on HDR can provide large overestimates of the au of itemsets, algorithms relying on them can explore many candidates, which leads to long runtimes.

In contrast, UBs and WUBs based on a VDR are computed using the utilities of items in columns of the Q matrix, and are called vertical UBs and WUBs. Truong et al. [37] designed four vertical UBs, named $\overline{aub}_1$, $\overline{aub}$, $\overline{iaub}$ and $\overline{laub}$, and integrated them into the dHAUIM algorithm [37] to mine HAUIs. Recently, another four WUBs, named $\overline{vmsub}_1$, $\overline{vmsub}$, $\overline{vmaub}$ and $\overline{vtopmaub}$, were proposed and used in the VMHAUI algorithm [38]. As discussed in [38], vertical UBs and WUBs are usually better than those relying on an HDR.

Hiding association rules, and frequent or high utility itemsets. Many data mining techniques were designed to discover hidden information and reveal interesting relationships in data. But put in the wrong hands, these algorithms can extract private or sensitive information that may lead to serious privacy issues. To address this problem, privacy-preserving data mining (PPDM) has emerged as an important research area, where the aim is to integrate privacy protection in the knowledge discovery process. Sensitive Pattern Hiding (SPH) is a PPDM task, which consists of transforming a database so that sensitive patterns cannot be discovered by data mining techniques. In the following paragraphs, SPH approaches for (quantitative) transaction databases are reviewed and discussed.

For transaction databases, Verykios et al. [40] designed strategies and algorithms for hiding association rules by changing their support and confidence. Li et al. introduced the MICF algorithm [16]. It first identifies transactions containing frequent itemsets (FIs) and then, in each transaction, it deletes the item that appears in the largest number of FIs. This process is repeated to hide all FIs. Hong et al. [11] presented a hiding algorithm named SIF-IDF to conceal sensitive rules based on the SIF and IDF measures. The algorithm first computes the SIF and IDF values of each transaction and then identifies itemsets to be deleted to hide FIs. In addition, a genetic algorithm-based approach [24] to hide FIs by deleting transactions and a PSO (particle swarm optimization) approach [25] to quickly sanitize a database for hiding FIs were designed. Most algorithms hide FIs or association rules by deleting transactions containing them or itemsets from the original database to decrease their support or confidence.

Some algorithms were proposed to hide high utility itemsets (HUIs) in a quantitative transaction database (QTDB), a task called Privacy-Preserving Utility Mining (PPUM) [7,28]. Yeh and Hsu [43] developed the HHUIF and MSICF algorithms to hide HUIs for PPUM. To hide a HUI, the HHUIF algorithm finds the item that has the maximum utility to delete that item or decrease its utility in transactions containing the HUI. MSICF uses a similar method, but the chosen item has the maximal frequency. MSICF may spend more time to hide HUIs because transactions are chosen for sanitization without considering the information about HUIs that are not hidden yet. In [19], a genetic algorithm was designed to conceal HUIs. It adds fake transactions to a QTDB so that its size increases, and thus the support of HUIs is decreased. Another genetic algorithm named PPUMGAT was also introduced by Lin et al. [17]. Instead of inserting dummy transactions into a QTDB, the PPUMGAT algorithm uses transaction deletion to hide HUIs. It was shown in [17] that the method of deleting transactions is more efficient for hiding HUIs than inserting transactions in terms of execution time and database integrity (or quality). Lin et al. proposed two algorithms named MSU-MAU and MSU-MIU to hide HUIs [26]. These algorithms hide HUIs by deleting items or decreasing their quantities and relying on the concepts of minimum and maximum utility of items. Besides, the authors also introduced three similarity measures to evaluate the effectiveness and efficiency of the hiding process for PPUM.

Most of the above pattern hiding algorithms use a traditional approach of hiding patterns one by one. An advantage of this approach is that it is simple and easy to implement. However, its main drawback is that it consumes

much time and memory, especially when the number of patterns to be hidden is large for low minimum thresholds. Sun and Yu [34] proposed a border-based approach to hide frequent itemsets. This approach aims at only hiding a small subset of all patterns from the border such that all remaining patterns are also hidden. Hence, the performance of hiding frequent itemsets can be significantly improved. However, that approach cannot be directly adapted for HAUIM as the average-utility does not respect the DCP. To overcome this challenge, this paper presents novel theoretical results and propose a corresponding algorithm to efficiently hide sensitive FHAUIs using an extended border-based approach.

3. Fundamental concepts and problem definition

This section introduces fundamental concepts and notation regarding the problem of hiding frequent high average-utility itemsets. They are necessary for discussing the novel theoretical results presented in the following sections. Also, a running example is introduced, which will be used to explain the various concepts throughout the paper, to facilitate understanding.

The type of data considered in this paper is called a quantitative transaction database, which is commonly used to store transactions made by customers with information about purchase quantities and unit profits of items.

Definition 1 (*Quantitative transaction database*). Given a set of items $\mathcal{A} = \{a_1, a_2, \ldots, a_m\}$, a subset A of $\mathcal{A}$ is called an *itemset*. Without loss of generality, elements of A are sorted according to the lexicographical order $\prec$. Moreover, each item a of $\mathcal{A}$ is associated with a positive number $p(a)$ (called *external utility*), which indicates its importance or unit profit. Each item a of A is also associated with a positive number q (named *internal utility*), which may indicate information such as a purchase quantity. A pair (a, q) is said to be a *quantitative item*, or briefly q-item. A *quantitative itemset* (q-itemset) is a set of q-items. A *quantitative transaction database* (QTDB) is a finite set of input quantitative *itemsets* (or transactions), $\mathcal{D}$ = {transaction $T_i \mid i = 1, 2, ..., n\}$. Moreover, a *unit profit* vector is defined as all external utility values of items, that is $\mathcal{P} = (p(a) \mid a \in \mathcal{A})$. Each transaction T_i of $\mathcal{D}$ is associated with a unique transaction identifier, denoted as $TID(T_i) = i$.

For the convenience of the reader, Table 1 summarizes the notations used in the rest of this paper.

Table 1. Notations.

Type	Representation	Example
Item	Roman letter	$a, b, c, x_{index}, y_{index}$
Itemset	Capitalized roman letter	A, B, C
q-item	(Roman letter, number)	$(a, 2), (x_1, 4)$
q-itemset	{q-item, …, q-item}	$\{(a, 2), (b, 4)\}$
Input transaction	{q-item, …, q-item}	$t_i = T = \{(y_1, 5), \ldots, (y_k, 7)\}$

An itemset $A = \{x_1, x_2, \ldots, x_k\}$ of $\mathcal{A}$ is said to be included in a quantitative itemset $T = \{(y_1, q_1), (y_2, q_2), \ldots, (y_l, q_l)\}$ and denoted as $A \subseteq T$, if there exist (*unique*) integers $1 \leq i_1 < i_2 < \ldots < i_k \leq l$ such that $x_j = y_{i_j}, \forall j = 1, \ldots, k$. Let $\rho(A) = \{T \in \mathcal{D} \mid T \supseteq A\}$ be the set of all transactions containing A. The *support* of an itemset A, denoted as $supp(A)$, is the number of transactions containing A, that is $supp(A) \stackrel{\text{def}}{=} |\rho(A)|$.

Running example. An example QTDB $\mathcal{D}$ is presented in Table 2. It has four transactions and six distinct items (Table 2.a) and a profit vector $\mathcal{P}$ representing the unit profits of items (Table 2.b). In the context of market basket analysis, these transactions may represent different sets of items purchased by customers. The transaction t_1 indicates that a customer has purchased 1 unit of item b, 5 units of items d, 13 units of item e, and 4 units of item f. The unit profits of these items are 2, 1, 3 and 4, respectively. The itemset $\{b, d\}$ appears in transaction $\rho(\{b, d\}) = \{t_1, t_2\}$ and hence has a support of 2. For brevity, we hereafter denote an itemset $\{b, d\}$ as bd and a transaction set $\{t_1, t_2\}$ as 12.

Table 2.a. A QTDB $\mathcal{D}$.

TID \ Item *it*	***a***	***b***	***c***	***d***	***e***	***f***
t_1	0	1	0	5	13	4
t_2	8	2	0	10	0	0
t_3	2	1	1	0	0	0
t_4	1	0	10	3	0	0

Item *it*	***a***	***b***	***c***	***d***	***e***	***f***
$p(it)$	1	2	2	1	3	4

Table 2.b. A profit vector $\mathcal{P}$.

Note that the above ρ operator maps each non-empty itemset A to transactions where all items of A have purchase quantities greater than zero. Thus, in the rest of this paper, only itemsets appearing in at least one transaction are considered, that is those of the set $\mathcal{IS} \stackrel{\text{def}}{=} \{A \subseteq \mathcal{A} \mid \rho(A) \neq \emptyset\}$. For instance, consider the itemset $A = bd$. Since $A \subseteq t_1, t_2$, we have $\rho(A) = 12$, and thus $supp(A) = 2$.

Definition 2 (*Integrated quantitative matrix*). To avoid repeatedly calculating *the utility* $u((a_j, q_{ij})) = q_{ij} * p(a_j)$ of a q-item (a_j, q_{ij}) in a transaction t_i when discovering high utility patterns from a quantitative database, the utility of each q-item can be computed once and stored in the corresponding transaction. The database transformed from a quantitative database QTDB $\mathcal{D}' \stackrel{\text{def}}{=} \{t_i \stackrel{\text{def}}{=} \{(a_j, q'_{ij}), j \in J_i\}, J_i \subseteq J, i \in I\}$ by applying this process is called an *integrated quantitative matrix* (or, briefly, *integrated matrix*) $\mathcal{Q}$, where $I \stackrel{\text{def}}{=} \{1,2,\dots,n\}$ and $J \stackrel{\text{def}}{=} \{1,2,\dots,m\}$ are, respectively, two sets of line and column indexes of $\mathcal{Q}$, defined as $\mathcal{Q}_{nxm} \stackrel{\text{def}}{=} \{q'_{ij}, i \in I, j \in J\}$ and $q'_{ij} = q_{ij} * p(a_j)$ if $(a_j, q_{ij}) \in t_i$ (otherwise, $q'_{ij} = 0$), $\forall i \in I, j \in J$.

Table 3 provides an example of the integrated matrix $\mathcal{Q}$, corresponding to the QTDB of Table 2. The value of the item e in the first row is 39, which is obtained by multiplying this item's internal utility of 13 in t_1 by its unit profit of 3. Hereafter, we only consider the *integrated* QTDB (or the integrated matrix Q) and for brevity, it will also be denoted as $\mathcal{D}$.

Table 3. The integrated matrix $\mathcal{Q}$ corresponding to Tables 2.a-2.b.

TID \ Item *it*	***a***	***b***	***c***	***d***	***e***	***f***
t_1	0	2	0	5	39	16
t_2	8	4	0	10	0	0
t_3	2	2	2	0	0	0
t_4	1	0	20	3	0	0

In HAUIM, itemsets that are presented to the user are selected based on the average utility measure, which is defined as follows.

Definition 3 (*Utility and average utility of itemsets* [9,37]). The utility of a q-item (a, q) in a QTDB $\mathcal{D}$ is denoted and defined as $u(a, q) \stackrel{\text{def}}{=} q \times p(a)$. The utility of a q-itemset $t = \{(y_1, q_1), (y_2, q_2), \dots, (y_k, q_k)\}$ in a QTDB $\mathcal{D}$ is denoted as $u(t)$ and defined as $u(t) \stackrel{\text{def}}{=} \sum_{1 \leq j \leq k} u(y_j, q_j)$, $u(\mathcal{D}) \stackrel{\text{def}}{=} \sum_{1 \leq i \leq n} u(t_i)$. The utility of an itemset $A = \{x_1, x_2, \dots, x_k\}$ in an input q-itemset $t_i = \{(y_1, q_1), (y_2, q_2), \dots, (y_l, q_l)\}$ containing A (i.e. $\exists$ integers $1 \leq i_1 < i_2 < \dots < i_k \leq l$ such that $x_j = y_{i_j}, \forall j = 1, \dots, k$) is denoted and defined as $u(A, t_i) \stackrel{\text{def}}{=} \sum_{1 \leq j \leq k} q_{i_j} \times p(y_{i_j}) = \sum_{j:\, a_j \in A} u(a_j, t_i)$, where $u(a_j, t_i) \stackrel{\text{def}}{=} q'_{ij}$ according to the i^{th} row and the j^{th} column of the integrated matrix $\mathcal{Q}$. The utility of A in $\mathcal{D}$ is denoted as $u(A)$) and defined as $u(A) \stackrel{\text{def}}{=} \sum_{t_i \in \rho(A)} u(A, t_i)$, and as a convention $u(\emptyset) = u(\emptyset, t_i) \stackrel{\text{def}}{=} 0, \forall t_i \in \mathcal{D}$. The average-utility of A is denoted and defined as $au(A) \stackrel{\text{def}}{=} u(A)/length(A)$.

More generally, the average-utility of an itemset A in a transaction subset T of $\rho(A)$ is denoted and defined as $au(A, T) \stackrel{\text{def}}{=} u(A, T) / |A|$, where $u(A, T) \stackrel{\text{def}}{=} \sum_{t_i \in T} u(A, t_i)$ is the utility of A in T, and $u(A) \stackrel{\text{def}}{=} \sum_{T \in \rho(A)} u(A, T)$.

Moreover, it was shown that the average-utility can also be defined using a vertical form [37,38] as $u(A) \stackrel{\text{def}}{=} \sum_{a_k \in A} v_k(A, \rho(A))$, where $v_k(A,T) \stackrel{\text{def}}{=} \sum_{t_i \in T} u(a_k, t_i)$ is the vertical utility according to the k^{th} column of the utility matrix Q (see Table 3) in a transaction subset T of $\rho(A)$, $a_k \in A$ and $T \subseteq \rho(A)$.

For example, consider the itemset $A = bd$. Then, $\rho(A) = 12$, $u(A, t_1) = 2 + 5 = 7$, and $u(A, t_2) = 4 + 10 = 14$. The utility of A in horizontal form is $u(A) = u(A, t_1) + u(A, t_2) = 7 + 14 = 21$. Moreover, since the indexes of b and d are 2 and 4, respectively, we have $v_2(A, \rho(A)) = 2 + 4 = 6$ and $v_4(A) = 5 + 10 = 15$. Thus, the utility of A in vertical form is $u(A) = v_2(A, \rho(A)) + v_4\big(A, \rho(A)\big) = 6 + 15 = 21$. The average-utility of A is $u(A) = u(A) / 2 = 10.5$.

Definition 4 (*Transaction utility, the total utility of a QTDB* [9]). The transaction utility of a transaction t_i is denoted and defined as $tu(t_i) \stackrel{\text{def}}{=} \sum_{j:q'_{ij}>0} q'_{ij}$. Then, the total utility of the *integrated* QTDB $\mathcal{D}$ is denoted as $TU(\mathcal{D})$ and defined as the sum of the utilities of all its transactions, i.e. $TU(\mathcal{D}) \stackrel{\text{def}}{=} \sum_{t_i \in \mathcal{D}} tu(t_i)$.

For example, $tu(t_1) = 2 + 5 + 39 + 16 = 62$ and $TU(\mathcal{D}) = tu(t_1) + tu(t_2) + tu(t_3) + tu(t_4) = 62 + 22 + 6 + 24 = 114$.

Given user-specified minimum utility and support thresholds denoted as mu and ms (positive numbers), and a QTDB $\mathcal{D}$, an itemset A is said to have a *high average utility* (HAU) if $au(A) \geq mu$. An itemset A is said to be a *frequent high average utility* itemset (FHAUI) if it is *frequent* (i.e. $supp(A) \geq ms$) and *high average utility*. Let $FHAUI_{\mathcal{D}}$ (or, briefly, $FHAUI$) be the set of all FHAUIs in $\mathcal{D}$.

For instance, assume that mu and ms are set to 9.46 (or 8.3% of $TU(\mathcal{D}) = 114$) and 1 (or 25% of $size(\mathcal{D}) = 4$), respectively. In the integrated matrix Q of the running example, the discovered FHAUIs, their average utilities and supports are $FHAUI = \{a^3_{11}, c^2_{22}, d^3_{18}, e^1_{39}, f^1_{16}, ac^2_{12.5}, ad^2_{11}, bd^2_{10.5}, be^1_{20.5}, bde^1_{15.3}, bef^1_{19}, bdef^1_{15.5}, cd^1_{11.5}, de^1_{22}, df^1_{10.5}, def^1_{20}, ef^1_{27.5}\}$, where the notation A^s_{au} means that for an itemset A, $supp(A) = s$ and $au(A) = au$. The number of FHAUIs is $|FHAUI| = 17$.

Some of the reasons for considering the concept of FHAUI instead of that of HAUI in this study are the following. *First*, if $ms = 1$, the concept of FHAUI becomes that of HAUI, i.e. the former is more general than the latter. *Second*, we can always transform the problem of mining HAUIs into that of mining FHAUIs. *Indeed*, if $au(A) \geq mu$, then $mu \leq supp(A) \times maxQ$, and hence $supp(A) \geq ms$, where $ms \stackrel{\text{def}}{=} \lceil mu/maxQ \rceil$, $maxQ \stackrel{\text{def}}{=} max\{q | (a,q) \in \mathcal{D}\}$ and $\lceil x \rceil$ is the ceiling of a positive real number x. For instance, for the running example, if $mu = 40, maxQ = 39$, then $ms = \lceil 40/39 \rceil = 2$. By applying the $\mathcal{AM}$ property of $supp$ and the support-based border approach, we can enhance the performance of the process of mining FHAUIs as well as hiding the set of all FHAUIs, especially for high mu threshold values.

Let $FHAUI$ be the set of all FHAUIs to be hidden. To hide all FHAUIs, the QTDB $\mathcal{D}$ needs to be transformed into a QTDB $\mathcal{D}'$ such that the set of FHAUIs obtained from $\mathcal{D}'$, denoted as $FHAUI_{\mathcal{D}'}$ (or, briefly, $FHAUI'$), becomes empty. As a result, the quality or total utility of the original database can be significantly reduced. Thus, an algorithm designed for hiding all FHAUIs needs to ensure that the total utility of $\mathcal{D}$ (i.e. $TU(\mathcal{D})$) is decreased as minimal as possible. Then, the sets of side effects (hiding failure, missing cost and artificial cost) [19,25] become empty. The reason is that the $FHAUI$ set of sensitive itemsets to be hidden includes all FHAUIs, and that the sanitized dataset $\mathcal{D}'$ is obtained by deleting transactions and/or decreasing the internal utility of items in transactions of $\mathcal{D}$. Thus, to evaluate the performance of algorithms in this aspect, the following definition using the *relative change of utility ratio* (or *utility loss* from the database) is considered.

Definition 5 (*Relative ratio of utility difference*). The relative ratio of utility difference r_{tu} of the QTDB $\mathcal{D}$ is defined as follows:

$$r_{tu} \stackrel{\text{def}}{=} (TU(\mathcal{D}) - TU(\mathcal{D}'))/TU(\mathcal{D}),$$

where $TU(\mathcal{D})$ is the total utility of $\mathcal{D}$ before applying the hiding process, and $TU(\mathcal{D}')$ is the total utility of the QTDB $\mathcal{D}'$ after performing the hiding process. Hereafter, we will rely on the ratio r_{tu} to evaluate the quality loss resulting from sanitizing a QTDB.

Problem Definition (FHAUI Hiding). The problem considered in this paper is to hide all FHAUIs (sensitive itemsets) by transforming $\mathcal{D}$ into a QTDB $\mathcal{D}'$ by *decreasing the quantities* of items in transactions of $\mathcal{D}$ such that the ratio r_{tu} is as small as possible.

4. Hiding FHAUIs

Note that although mining HAUIs has many advantages compared to discovering all HUIs, there is to our best knowledge no algorithm for hiding all FHAUIs. To solve that problem, this section proposes two novel algorithms named H-FHAUI-B and H-FHAUI. The former is a baseline algorithm that hides all FHAUIs one by one, while the latter is a novel efficient algorithm that improves the performance of FHAUIH using an extended border-based approach.

4.1. Baseline approach for hiding FHAUIs

The baseline approach for hiding all FHAUIs is designed based on the idea that an itemset $A \in FHAUI$ can be hidden by performing the following two steps repeatedly until A becomes either infrequent or not high average utility: (1) choose an item $a_l \in A$, a transaction $t_k \in \rho(A)$, and find a corresponding q-item $(a_l, q) \in t_k$; (2) decrease q by a suitable quantity or delete the q-item from t_k. However, an important issue is how to choose a_l and t_k such that the hiding process terminates quickly and the total utility of the QTDB $\mathcal{D}$ is minimally reduced. There are some simple heuristic methods to solve this problem, such as using maximum, minimum or random functions for selecting an item a_l and a transaction t_k. To design a baseline algorithm for hiding all FHAUIs, the current study uses the following strategy, which is based on the maximum function with the goal of decreasing the utility as well as the average utility of patterns to be hidden as quickly as possible.

Simple item-transaction choosing strategy (SCS). To quickly decrease the value of $au(A)$, and thus the execution time of the algorithm, the item a_l is chosen as follows:

$$a_l \stackrel{\text{def}}{=} argmax\{v_j(A, \rho(A)), a_j \in A\}, \text{ i.e. } v_l(A, \rho(A)) \stackrel{\text{def}}{=} max\{v_j(A, \rho(A)), a_j \in A\}.$$

After a_l is selected, the transaction t_k is chosen such that

$$t_k \stackrel{\text{def}}{=} argmax\{u(a_l, t_i), t_i \in \rho(A)\}, \text{ i.e. } u(a_l, t_k) \stackrel{\text{def}}{=} max\{u(a_l, t_i), t_i \in \rho(A)\}.$$

This means that we choose the l^{th} column with the maximum vertical utility $v_l(A, \rho(A))$ among columns containing items of A, and then the transaction t_k in that column with the maximum utility value $u(a_l, t_k)$.

$q'_{12} + q'_{22} = 2 + 4 = 6$ and $v_4(A, TS(A)) = u(a_4, t_1) + u(a_4, t_2) = q'_{14} + q'_{24} = 5 + 10 = 15$. Since $v_4(A, TS(A)) > v_2(A, TS(A))$ and $u(a_4, t_2) > u(a_4, t_1)$, the selected item and transaction are $a_l = a_4 = d$ and $t_k = t_2$, respectively.

Consider a FHAUI $A \in FHAUI$, $supp(A) \geq ms$ and $au(A) \geq mu$, to be hidden. Let $difs(A) = supp(A) - ms + 1$ (≥ 1) and $difu(A) = \lfloor u(A) - |A| \times mu \rfloor + 1$ (≥ 1) be the *minimum decreased* quantities for the support and utility of A such that $supp(A) < ms$ and $au(A) < mu$, respectively. *Indeed*, after decreasing the support and utility of A by $difs(A)$ and $difu(A)$, we have the corresponding updated values $supp(A) := supp(A) - difs(A) = ms - 1 < ms$ and $au(A) := (u(A) - difu(A))/|A| < (|A| \times mu)/|A| = mu$ since $\lfloor x \rfloor + 1 > x$ (for any real number x, where $\lfloor x \rfloor$ is the floor integer of x).

For each itemset A of $FHAUI$ ($supp(A) \geq ms$ and $au(A) \geq mu$), A can be hidden by applying the following two strategies that gradually decrease the support and utility of A based on the minimum decreased quantities $difs(A)$ and $difu(A)$.

***Decreasing support strategy* ($\mathcal{DSS}$)**. This strategy aims at decreasing the support of A by repeatedly performing the following steps until the condition $difs(A) \leq 0$ (or $supp(A) < ms$) holds:

(1) Choose an item $a_l \in A$ and a transaction $t_k \in \rho(A)$ (containing a_l) based on SCS.
(2) Delete a_l from t_k and update other information about t_k accordingly, if needed.

Note that when $\mathcal{DSS}$ is used, we need to delete $difs(A)$ items in $difs(A)$ different transactions such that $supp(A) < ms$.

***Decreasing utility strategy* ($\mathcal{DUS}$)**. This strategy aims at lowering the average utility of A by repeatedly performing the following steps until the conditions $difs(A) \leq 0$ or $difu(A) \leq 0$ are satisfied:

(1) Select an item $a_l \in A$ and a transaction $t_k \in \rho(A)$ based on SCS.
(2) Update the quantity (internal utility) $q(a_l, t_k)$ of a_l in t_k as $q(a_l, t_k) := max\{q(a_l, t_k) - \lceil difu(A)/p(a_l) \rceil, 0\}$, and the values of $difs(A)$ and $difu(A)$, where $\lceil x \rceil$ denotes the ceiling integer of a real number x.

If the updated value $q(a_l, t_k)$ is decreased to 0, the item a_l is removed from t_k, so $supp(A)$ is decreased by 1. Otherwise, $supp(A)$ is not changed. In other words, decreasing the utility of items is more general than deleting items.

The main advantage of $\mathcal{DSS}$ is that the hiding process is performed simply. However, the total utility $tu(\mathcal{D})$ of the database $\mathcal{D}$ can be greatly reduced (i.e. the r_{tu} value is quite large) because deleting the item a_l from t_k means that the utility $q(a_l, t_k)$ is immediately decreased to 0. Applying $\mathcal{DUS}$ requires more time than $\mathcal{DSS}$ to hide FHAUIs and $tu(\mathcal{D}')$ may be larger (or r_{tu} may be smaller). Thus, an important question arises: in which cases should $\mathcal{DSS}$ or $\mathcal{DUS}$ be used to hide a FHAUI such that the ratio r_{tu} is minimal? In the following, we present theoretical results to provide an answer to this question.

Definition 6 (*Maximum and minimum decreased utility quantity*). Assume that the utilities $(u(A,t), t \in \rho(A))$ of a frequent itemset A are sorted in *descending* order to obtain the series $(u(A, t_{k_i}), 1 \leq i \leq m)$ with $m \stackrel{\text{def}}{=} supp(A) \geq ms$. Then, we define the *maximum* and *minimum boundary* values that indicate two maximum and minimum decreased utility quantities as:

$$Bd_{max}(A) \stackrel{\text{def}}{=} \sum_{1 \leq i \leq difs(A)} u(A, t_{k_i}) \text{ and}$$
$$Bd_{min}(A) \stackrel{\text{def}}{=} \sum_{m - difs(A) + 1 \leq i \leq m} u(A, t_{k_i}).$$

Remark 1. If $Bd_{max}(A) \leq difu(A)$, i.e. the maximum decreased utility quantity is insufficient for $au(A) < mu$, $\mathcal{DSS}$ is *better* than $\mathcal{DUS}$ since when using $\mathcal{DUS}$, we must further decrease the utilities of items in more than $difs(A)$ transactions. Conversely, if $Bd_{max}(A) > difu(A)$, then there are the following two possible cases. If $Bd_{min}(A) > difu(A)$, i.e. the minimum decreased utility quantity is sufficient for $au(A) < mu$, then $\mathcal{DSS}$ is *less efficient* than $\mathcal{DUS}$. The reason is that when lowering the utility such that $au(A) < mu$ by adopting $\mathcal{DUS}$, $supp(A)$ can still be larger than ms. Otherwise, if $Bd_{max}(A) > difu(A) \geq Bd_{min}(A)$, $\mathcal{DUS}$ should be used because decreasing the utility of items (in transactions) is *more general and flexible* than *deleting* items. Thus, in the case of $Bd_{max}(A) > difu(A)$, $\mathcal{DUS}$ is *more efficient* than $\mathcal{DSS}$.

For example, for $ms = 1$ and $mu = 9.46$, consider the itemset $A = bd$. We have $supp(A) = 2$ and $u(A) = 21$. Then, $difs(A) = 2$, $difu(A) = 3$ and $Bd_{max}(A) = Bd_{min}(A) = u(A, t_1) + u(A, t_2) = 7 + 14 = 21$. To hide A we should use the $\mathcal{DUS}$ strategy instead of $\mathcal{DSS}$, since $Bd_{max}(A) > difu(A)$. *Indeed*, if using $\mathcal{DSS}$, we will delete item d from transactions t_1 and t_2, i.e. $u(d, t_1) = u(d, t_2) = 0$, so $supp(A) = 0 < ms$, and obtain $TU(\mathcal{D}') = 99$. Then, $r_{tu} = (114 - 99)/114 = 0.13$. In contrast, if using $\mathcal{DUS}$, the utility q'_{24} of d in t_2 will be reduced by $difu(A) = 3$, i.e. $q'_{24} = 10 - 3 = 7$ (in $\mathcal{D}'$), thus $au(A) = u(A)/|A| = (21 - 3)/2 = 9 < mu$ and $TU(\mathcal{D}') = 111$ (> 99). Hence, $r_{tu} = (114 - 111)/114 = 0.026$ (< 0.13). Similarly, for $ms = 2$ and $mu = 1$, consider the itemset $A = d$. We have $supp(A) = 3$ and $u(A) = 18$. Then, $difs(A) = 2$, $difu(A) = 18$, and $Bd_{max}(A) = u(A, t_1) + u(A, t_2) = 5 + 10 = 15$. Thus, $Bd_{max}(A) \leq difu(A)$, $\mathcal{DSS}$ should be used to hide A. *Indeed*, if $\mathcal{DSS}$ is used, the item d will be removed from the transactions t_2 and t_1 such that $supp(A) = 1 < ms$. Then, $TU(\mathcal{D}') = 99$ and $r_{tu} = 0.13$. Otherwise, if $\mathcal{DUS}$ is utilized, the utility of A needs to be decreased by the quantity $difu(A)$ such that the condition $au(A) < mu$ holds. Then, $TU(\mathcal{D}') = 96$ and $r_{tu} = 0.16$ (> 0.13).

The H-FHAUI-B algorithm. Based on the above theoretical results, we propose a baseline algorithm named H-FHAUI-B (Hiding Frequent High Average-Utility Itemsets – Baseline) for hiding all FHAUIs. The pseudo-code of H-FHAUI-B is shown in Fig. 1 (see Table 6 in the appendix for the meaning of notation and abbreviation used in this algorithm). For each itemset $A \in FHAUI$, H-FHAUI-B calls the HidingOneItemset procedure to hide A. In that HidingOneItemset procedure, after calculating the values of $difs(A)$, $difu(A)$, and $Bd_{max}(A)$ (lines 2-3), the algorithm checks the condition $Bd_{max}(A) \leq difu(A)$ to choose the most efficient strategy ($\mathcal{DSS}$ or $\mathcal{DUS}$) for the hiding process (lines 4-6).

H-FHAUI-B ($\mathcal{D}$, mu, ms, $FHAUI$, SCS**)**

Input: The original QTDB $\mathcal{D}$ represented in *vertical* form, the thresholds mu and ms, the set $FHAUI$ of all FHAUIs to be hidden, and the SCS strategy for choosing an item and a transaction.

Output: The sanitized QTDB $\mathcal{D}'$.

1. **for each** itemset $A \in FHAUI$ **do**
2. **HidingOneItemset** (A, $\mathcal{D}$, mu, ms, $FHAUI$, SCS);

Procedure HidingOneItemset (A, $\mathcal{D}$, mu, ms, $FHAUI$, SCS)

1. **while** ($supp(A) \geq ms$ and $au(A) \geq mu$) **do** {
2. $difu(A) = \lfloor u(A) - |A| \times mu \rfloor + 1$;
3. Calculate the maximum boundary value $Bd_{max}(A)$; *//Definition 7*
4. **if** ($Bd_{max}(A) \leq difu(A)$) **then** *//Remark 1*
5. $\mathcal{DSS}$ = true; *//Use $\mathcal{DSS}$ strategy*
6. **else** $\mathcal{DSS}$ = false; *//Use $\mathcal{DUS}$ strategy*
7. Choose the item $a_l \in A$ and the transaction $t_k \in TS(A)$ based on SCS
8. **UpdateValues**(A, $difu(A)$, a_l, t_k, $\mathcal{DSS}$).
9. Update the au and $supp$ of all remaining itemsets of $FHAUI$.
10. } *//end while*

Fig. 1. The H-FHAUI-B algorithm.

UpdateValues(A, $difu(A)$, a_l, t_k, $\mathcal{DSS}$**)**

Input: The FHAUI A, the minimal decreased utility $difu(A)$, the chosen item and transaction, a_l *and* t_k, respectively.

Output: The updated values, $supp(A)$, $u(A)$, $au(A)$ and $q(a_l, t_k)$.

1. **if** ($\mathcal{DSS}$) **then** { *// Use $\mathcal{DSS}$ or $I\mathcal{DSS}$ strategies*
2. $supp(A) = supp(A) - 1$;
3. $u(A) = u(A) - u(A, t_k)$;
4. $q(a_l, t_k) = 0$; *//remove a_l from t_k*
5. }
6. **else** { *// Use $\mathcal{DUS}$ or $\mathcal{DUS}_{WUB}$ strategies*
7. **if** ($\lceil difu(A)/p(a_l) \rceil > q(a_l, t_k)$) **then** {
8. $supp(A) = supp(A) - 1$;
9. $u(A) = u(A) - u(A, t_k)$;
10. $q(a_l, t_k) = 0$; *//remove a_l from t_k*
11. }
12. **else** {
13. $u(A) = u(A) - \lceil difu(A)/p(a_l) \rceil \times p(a_l)$;
14. $q(a_l, t_k) = q(a_l, t_k) - \lceil difu(A)/p(a_l) \rceil$; *//decrease the quantity $q(a_l, t_k)$ of a_l in t_k*
15. }
16. } *//end if*
17. $au(A) = u(A)/|A|$;

Fig. 2. The UpdateValues procedure.

Choosing an item a_l and a transaction t_k is done based on the SCS strategy (line 7). Then, the H-FHAUI-B algorithm calls the UpdateValues procedure in Fig. 2 to update values as needed and stops when all itemsets of $FHAUI$ have been hidden.

Note that the baseline approach can have good performance when the cardinality of the $FHAUI$ set is small. However, because the baseline needs to hide all FHAUIs of $FHAUI$ one by one, it may require more time and memory to complete the hiding process, especially when the cardinality of $FHAUI$ is very large. Moreover, in this case the total utility of the QTDB $\mathcal{D}$ can be greatly decreased. To overcome this issue, in the next subsection, we propose a novel approach to improve the performance of the hiding process based on border notions of a set of itemsets and weak upper bounds on the average-utility.

4.2. Border-based approach for hiding FHAUIs

This section presents novel theoretical results, which are the basis of the proposed efficient H-FHAUI algorithm for concealing all FHAUIs. The section firstly presents a novel efficient strategy to preserve the total utility of the QTDB $\mathcal{D}$ as much as possible during the sanitization process. Secondly, border concepts, which rely on the support of itemsets and weak upper bounds on the average-utility, are introduced. As it will be shown, using borders can remarkably reduce the number of itemsets that need to be hidden. Finally, this section describes the designed H-FHAUI algorithm, which integrates these ideas to efficiently conceal FHAUIs.

Note that the simple item-transaction selection strategy SCS in Sub-section 4.1 is applied by hiding patterns in $FHAUI$ one by one. Thus, hiding FHAUIs using the SCS strategy to choose an item $a_l \in A$ and a transaction $t_k \in \rho(A)$ can result in considerably decreasing the quality (total utility) of the QTDB $\mathcal{D}$ and consuming much time for the hiding process. To address this issue, the paper proposes choosing items a_l (and transactions t_k) that appear in (contain, respectively) as many FHAUIs of the $FHAUI$ set as possible. In other words, the chosen items a_l and transactions t_k should be related to many FHAUIs that have been not hidden yet. Then, decreasing the utility $q(a_l, t_k)$ may result in decreasing the utility and/or support of other FHAUIs simultaneously. This helps to significantly reduce the number of items and transactions that are chosen to decrease the utility. To utilize this idea, the following novel strategy named ICS is proposed for selecting items and transactions during the hiding process.

Improved item-transaction choosing strategy (ICS). To choose an item a_l and a transaction t_k such that the number of itemsets that are hidden is reduced as much as possible, we propose the following strategy.

- Select an item a_l with the *maximum item weight* $iw(a_l)$, $a_l \triangleq^{\text{def}} argmax\{iw(a_j), a_j \in A\}$, where
$$iw(a_j) \triangleq^{\text{def}} SC_{it}(FHAUI, a_j) \text{ and}$$
$SC_{it}(FHAUI, a_j)$ is the number of itemsets of $FHAUI$ containing a_j.

- Then, select a transaction t_k having the *maximum transaction weight* $tw(t_i)$, $t_k \triangleq^{\text{def}} argmax\{tw(t_i), t_i \in \rho(A)\}$, where
$$tw(t_i) \triangleq^{\text{def}} SC_t(FHAUI, t_i),$$
and $SC_t(FHAUI, t_i)$ is the number of itemsets of $FHAUI$ contained in t_i.

For example, consider the itemset $A = bd \in FHAUI$. We have $iw(b) = SC_{it}(FHAUI, b) = 5$ and $iw(d) = SC_{it}(FHAUI, d) = 9$. Thus, item d is chosen instead of b, since $iw(d) > iw(b)$. Similarly, because $\rho(A) = 12$, we have $tw(t_1) = SC_t(FHAUI, t_1) = 12$ and $tw(t_2) = SC_t(FHAUI, t_2) = 4$. Therefore, the transaction t_1 is selected instead of t_2 because $tw(t_1) > tw(t_2)$.

***Improved decreasing support strategy* ($IDSS$)**. To minimally decrease the total utility of $\mathcal{D}$, the SCS is replaced by the ICS in the $\mathcal{DSS}$ strategy.

To reduce the number of FHAUIs to be hidden by decreasing their support values, the proposed algorithm employs the concept of border. The following paragraphs recall key concepts related to border.

Definition 7 (*Border elements* [34]). Given a set of itemsets U, the upper border (or lower border) of U, denoted as $Bd^+(U)$ (or $Bd^-(U)$), is the set of *maximal* (or *minimal*) itemsets of U, i.e. $Bd^+(U)$ (or $Bd^-(U)$) is a subset of U, and $\forall Y \in Bd^+(U), \nexists Z \in Bd^+(U): Z \supset Y$, and $\forall X \in U, \exists Y \in Bd^+(U): Y \supseteq X$ (or $\forall Y \in Bd^-(U), \nexists X \in Bd^-(U): X \subset Y$, and $\forall Z \in U, \exists Y \in Bd^-(U): Y \subseteq Z$, respectively).

For example, consider the running example with $mu = 9.46$ and $ms = 1$, and for $U = FHAUI = \{a, c, d, e, f, ac, ad, bd, be, bde, bef, bdef, cd, de, df, def, ef\}$, then we obtain the sets $Bd^+(U) = \{ac, ad, cd, bdef\}$ and $Bd^-(U) = \{a, c, d, e, f\}$ as illustrated in Fig. 3.

It is found from this example that the cardinality of $Bd^-(U)$ is much less than that of U. Thus, a question that arises is if we can consider only the $Bd^-(U)$ set for hiding (instead of the U set) and still conceal all elements of U. The below proposition will answer this question positively.

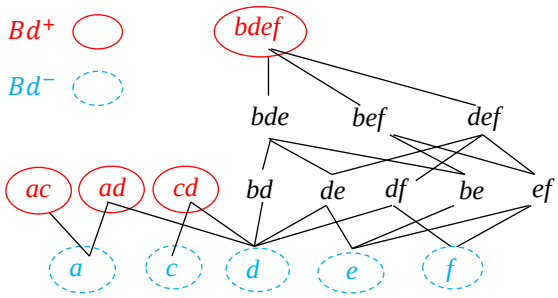


Fig. 3. The illustration of border elements.

a) Border approach based on support

Note that for any two itemsets X, Y such that $X \subseteq Y$, we have $supp(Y) \leq supp(X)$ (the anti-monotonic ($\mathcal{AM}$) property of the support). This means that if an itemset is infrequent, then all its super-itemsets are also infrequent. Based on this property and the lower border concept of Definition 7, the following proposition is derived.

Proposition 1 (*Hiding the FHAUI set based on the Lower Border according to the support - HLB_{supp}*). Consider the task of hiding the $FHAUI$ set (all FHAUIs). If the subset $Bd^-(FHAUI)$ is hidden by using $IDSS$ based on the support measure, then the whole set $FHAUI$ is also hidden.

Proof. It is obviously true because of the anti-monotonicity property of the support and the concept of lower border presented in Definition 7. □

Proposition 1 allows reducing the size (or cardinality) of the set of FHAUIs to be hidden. In other words, we only need to hide itemsets of the subset $Bd^-(FHAUI)$, whose cardinality is often quite small, by using $\mathcal{DSS}$ instead of the whole $FHAUI$ set of all FHAUIs.

For example, assume that we need to conceal the $FHAUI$ set of 17 FHAUIs for the running example database. Instead of hiding all itemsets of $FHAUI$, we only need to hide the 5 itemsets of the set $Bd^-(FHAUI) = \{a, c, d, e, f\}$ (Proposition 1).

b) Extended border approach based on the average-utility

The advantage of HLB_{supp} is that it can be used to reduce the number of FHAUIs to be hidden. However, because it is performed by removing q-items (x, q) from transactions (i.e. q is suddenly decreased to 0), the total utility of the QTDB $\mathcal{D}$ can greatly decrease. Thus, when applying the traditional border approach based on the support for the sequential pattern hiding problem to conceal FHAUIs, the average-utility (au) measure should additionally be

considered. Unfortunately, au does not respect the anti-monotonicity ($\mathcal{AM}$) property. A solution to address this problem is to use an extended border approach, which relies on upper bounds (UB) [37] or weak upper bounds (WUB) [38] on au that satisfy the $\mathcal{AM}$ or anti-monotonicity-like ($\mathcal{AML}$) properties.

For any two *non-empty* itemsets A and C such that $c \succ a, \forall c \in C$ and $a \in A$, the *proper forward* extension (or, briefly, *forward* extension) of A with C is the union $A \cup C$, denoted as $A \oplus C$. If $C = \{y\}$ including only an item y such that $y \succ a, \forall a \in A$, then $A \oplus \{y\}$ is denoted briefly as Ay. A function ub of itemsets is said to be an *upper bound* (UB) on au if $ub(A) \geq au(A)$, for any itemset A. A function ub of itemsets is said to be a weak upper bound (WUB) on au if $ub(A) \geq au(B)$, for any *forward* extension B of A.

For example, consider the UB $aub \triangleq \overline{aub_1}$ [37] on au satisfying $\mathcal{AM}$, i.e. $au(B) \leq aub(B) \leq aub(A), \forall B \supseteq A$. To conceal the itemset $bd \in Bd^-(FHAUI)$ based on aub, we transform $\mathcal{D}$ to $\mathcal{D}'$ such that $aub(bd) < mu$. Then, all super-itemsets of bd in $FHAUI$ such as bde, are also hidden (according to au), since $au(bde) \leq aub(bde) \leq aub(bd) < mu$.

Since the values of UBs proposed in [37] and the two WUBs, $\overline{vmsub_1}$ and $\overline{vmsub}$, presented in [30] are often much larger than the average-utilities of itemsets, transforming $\mathcal{D}$ into $\mathcal{D}'$ can greatly decrease the quality of $\mathcal{D}$. In [38], the authors proposed two other WUBs, named $\overline{vtopmaub}$ and $\overline{vmaub}$, that are tighter than these UBs or WUBs. In this context, given any two UBs or WUBs ub_1 and ub_2 on au, ub_1 is said to be *tighter* than ub_2 if $ub_1(A) \leq ub_2(A), \forall A$. Thus, in the rest of this sub-section, an extended border approach, which is based on the latter two WUBs on au to enhance the performance of the hiding process, will be proposed.

Two subsets of transactions containing a given itemset A are denoted and defined as: $TS(A) \triangleq \{t_i \in \rho(A) \mid MinRU(a_{maxInd(A)}, t_i) \neq 0\}$ and $TS_j(A) \triangleq \{t_i \in \rho(A) \mid MinU_j(A, t_i) \neq 0\}$, where $Max = maxInd(A) \triangleq max\{j \in J \mid a_j \in A\}$, $MinRU(a_j, t_i) \triangleq min\{u(a_k, t_i) \mid k > j\}$, and $MinU_j(A, t_i) \triangleq min\{u(a_k, t_i) \mid k \geq j \text{ and } a_k \notin A\}$. We have the following definition.

Definition 8 (*Two weak upper bounds on* au [38]).
a) A first WUB on au, named $\overline{vmaub}$, is defined as follows:

$$\overline{vmaub}(A) \triangleq \begin{cases} f(mn(A)), if\ vmru(A,\ TS(A)) > au(A, TS(A)) \\ f(1), \quad\quad if\ vmru(A,\ TS(A)) \leq au(A, TS(A)) \end{cases},$$

where $vmru(A,\ TS(A)) \triangleq max\{v_k(A,\ TS(A)) \mid k > Max\}$ is the vertical maximum remaining utility according to A, $f(e) \triangleq \frac{L.au(A,\ TS(A)) + e.vmru(A,TS(A))}{L+e}$ is an intermediate value between $au(A, TS(A))$ and $vmru(A,\ TS(A))$, $\forall e = 1 \ldots mn(A)$, $Max = maxInd(A)$, $L = |A|$, $n(a_{Max}, t_i) \triangleq |\{a_k \in t_i \mid k > Max\}|$ is the number of remaining items in t_i after a_{Max}, and $mn(A) \triangleq max\{n(a_{Max}, t_i) \mid t_i \in TS(A)\}$ is the maximum number of remaining items in $\mathcal{D}$ according to A.

b) Let $\mathcal{V}(A, TS(A)) \triangleq (v_k(A, TS(A)),\ a_k \in A \text{ or } k > maxInd(A))$. Without loss of generality, the values of $\mathcal{V}(A, TS(A))$ can be sorted in descending order, i.e. $\mathcal{V}(A, TS(A)) = (v_1^*(A, TS(A)),\ v_2^*(A, TS(A)), \ldots, v_n^*(A, TS(A)))$, where $n > |A|$ and $v_j^*(A, TS(A)) \geq v_{j+1}^*(A, TS(A)), \forall j = 1..n-1$. A second WUB on au, named $\overline{vtopmaub}$, is defined as:

$$\overline{vtopmaub}(A) \triangleq \frac{\sum_{j=1}^{|A|+1} v_j^*(A,\ TS(A))}{|A|+1}.$$

Then, $\overline{vmaub}$ and $\overline{vtopmaub}$ are two WUBs on au [38].

For example, consider the itemset $A = bd$, $\rho(A) = 12$, $TS(A) = 1$ and $Max = maxInd(A) = 4$. Because $a_2 = b$ and $a_4 = d$ belong to A, the vertical utilities of A in $TS(A)$ at the columns 2 and 4 are $v_2(A,\ TS(A)) = 2$ and $v_4(A,\ TS(A)) = 5$, respectively. Besides, we have $v_5(A,\ TS(A)) = 39$ and $v_6(A,\ TS(A)) = 16$. Thus, $\mathcal{V}(A, TS(A)) = (v_2(A,\ TS(A)),\ v_4(A,\ TS(A)),\ v_5(A,\ TS(A)),\ v_6(A,\ TS(A))) = (2, 5, 39, 16)$. After sorting $\mathcal{V}(A, TS(A))$ in descending order, we obtain $\mathcal{V}(A, TS(A)) = (v_1^*(A, TS(A)), v_2^*(A, TS(A)),\ v_3^*(A, TS(A)), v_4^*(A, TS(A)) =$

$(39, 16, 5, 2)$. Therefore, $\overline{vtopmaub}(A) = (39 + 16 + 5)/3 = 20$. Meanwhile, since $mn(A) = 2, vmru(A, TS(A)) = max\{v_5(A,\ TS(A)), v_6(A,\ TS(A))\} = 39 > au(A, TS(A)) = 3.5$, we have $\overline{vmaub}(A) = f(mn(A)) = \frac{|A|.au(A,\ TS(A)) + mn(A).vmru(A,TS(A))}{|A|+mn(A)} = \frac{2 \text{ x } 3.5 + 2 \text{ x } 39}{2+2} = 21.3$.

To reduce the number of FHAUIs in the $FHAUI$ set to be hidden, this paper proposes the concept of an extended lower border of U based on WUBs $\overline{vmaub}$ and $\overline{vtopmaub}$ presented in Definition 8, which is introduced in the following definition.

Definition 9 (*Extended Lower Border*). Given a set U of itemsets, an *extended* lower border of U based on $\overline{vmaub}$ or $\overline{vtopmaub}$, denoted as $Bd_E^-(U)$, is the set of *minimal* itemsets of U w.r.t. the partial order relation based on *forward extension*, i.e. $Bd_E^-(U)$ is a subset of U such that $\forall Y \in Bd_E^-(U), \nexists X \in Bd_E^-(U)$: Y is a *forward extension* of X, and $\forall Z \in U, \exists Y \in Bd_E^-(U)$ such that Z is a *forward extension* of Y or $Z = Y$.

For example, consider the set $U = \{bd, bde, bdef, def, df\}$. Then, we have $Bd_E^-(U) = \{bd, def, df\}$. Note that although the itemset def is a super-itemset of df, it is not a forward extension of df and thus belongs to $Bd_E^-(U)$. In this case, $Bd^-(U) = \{bd, df\} \subset Bd_E^-(U)$. Fig. 4 shows an illustration of $Bd^-(U)$ and $Bd_E^-(U)$.

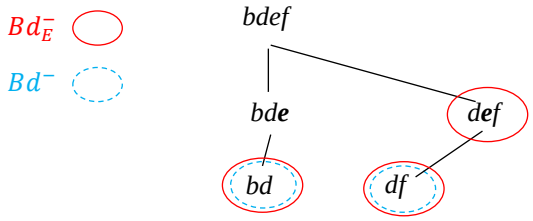


Fig. 4. An illustration of Bd^- and Bd_E^-.

Note that for large QTDBs or low mu and ms values, the cardinality of $Bd_E^-(FHAUI)$ is often much less than that of $FHAUI$. For example, consider the running example with $mu = 1$ and $ms = 1$. Then, the number of frequent high average utility itemsets in the $FHAUI$ set is 25, but the number of elements in the $Bd_E^-(FHAUI)$ set is only 6. Thus, hiding only FHAUIs of the $Bd_E^-(FHAUI)$ set that can conceal all remaining FHAUIs of $FHAUI$ is necessary to speed up the hiding process and to limit the utility loss (quality) of the original dataset. To do this, the below proposition is derived.

Proposition 2 (*Hiding the FHAUI set based on an Extended Lower Border according to the average utility* - $HELB_{au}$).

(*i*) For each itemset $A \in Bd_E^-(FHAUI)$, if we transform $\mathcal{D}$ into $\mathcal{D}'$ such that $\overline{vtopmaub}(A) < mu$ (or $\overline{vmaub}(A) < mu$), then all forward extension itemsets of A that belong to $FHAUI$ are hidden (according to au).

(*ii*) Thus, if all itemsets of the $Bd_E^-(FHAUI)$ set are concealed according to au, then the whole $FHAUI$ set is also hidden.

Proof.

(*i*) Let $B \in FHAUI$ be any forward extension itemset of A. Then, we have $au(B) \leq \overline{vtopmaub}(A) < mu$. Hence, B will be hidden according to au in $\mathcal{D}'$.

(*ii*) The assertion (*ii*) is clearly true according to Definition 9. □

Proposition 2 helps reducing the execution time of the hiding process because we only need to hide FHAUIs of $Bd_E^-(FHAUI)$ instead of the whole $FHAUI$ set.

Remark 2.

(*i*). Since the two WUBs $\overline{vmaub}$ and $\overline{vtopmaub}$ cannot be compared [38], they should be used concurrently during the hiding process.

(*ii*). Furthermore, because both of the WUBs $\overline{vmaub}$ and $\overline{vtopmaub}$ use a vertical database representation, for each itemset $A \in FHAUI$ to be hidden, choosing an item $a_l \in A$ should be performed before selecting a transaction t_k containing A in the SCS and ICS strategies.

Proposition 2 shows that for each itemset $A \in Bd_E^-(FHAUI)$, to hide all forward extension itemsets of A in the $FHAUI$ set, we only need to reduce the values of WUBs $\overline{vtopmaub}$ or $\overline{vmaub}$ such that the conditions $\overline{vtopmaub}(A) < mu$ or $\overline{vmaub}(A) < mu$ are satisfied. However, it is not simple to determine the minimum quantity to remove from $\overline{vtopmaub}$ or $\overline{vmaub}$ such that the conditions hold, but the total utility (quality) of the dataset is minimally decreased and the hiding process is performed quickly. To address this issue, Proposition 3 and Proposition 4 are derived.

Hiding a FHAU itemset based on $\overline{vtopmaub}$. For an itemset $A \in FHAUI$, from Definition 9 we have $\overline{vtopmaub}(A) \stackrel{\text{def}}{=} \frac{1}{|A|+1} Sum$, where $Sum \stackrel{\text{def}}{=} \sum_{k=1..|A|+1} v_{i_k}^*(A, TS(A))$ and $v_j^*(A, TS(A)) \stackrel{\text{def}}{=} \sum_{t_i \in TS(A)} u(a_j, t_i)$ is the utility of the j^{th} item of A in $TS(A)$. Then, the following proposition is derived.

Proposition 3. Minimum integer deviation needed to reduce from Sum such that $\overline{vtopmaub}(A) < mu$ is $difu = \lfloor Sum - (|A| + 1) \times mu \rfloor + 1 > 0$.

Proof. After reducing Sum by $difu$, we have $\overline{vtopmaub}(A) = (Sum - difu)/(|A| + 1) < ((|A| + 1) \times mu)/(|A| + 1) = mu$, since $\lfloor x \rfloor + 1 > x$. □

Hiding a FHAU itemset based on $\overline{vmaub}$. Consider an itemset $A \in FHAUI$. To hide A based on the WUB $\overline{vmaub}$, if $au' \stackrel{\text{def}}{=} au(A, TS(A)) \geq mu$, we *first* transform $\mathcal{D}$ to $\mathcal{D}'$ such that $au' < mu \leq au(A)$ by using the SCS or ICS strategies, where $\rho(A)$ is replaced by $TS(A)$. *Next*, we need to transform $\mathcal{D}$ into a database $\mathcal{D}'$ such that $\overline{vmaub}(A) = f(mn(A)) < mu$ according to the case $vmru(A,\ TS(A)) > au'$. Then, we have the following proposition.

Proposition 4. The minimum quantity that should be removed from $vmru(A,\ TS(A))$ so that $\overline{vmaub(A)} < mu$ is $diffu = vmru(A,\ TS(A)) - \frac{|A|.(mu - au')}{mn} - mu + 1$.

Proof. If $au' < mu \leq au(A)$, then $\overline{vmaub}(A) \geq au(A) \geq mu$. We will prove that $vmru' \stackrel{\text{def}}{=} vmru(A,\ TS(A)) > au'$, so $\overline{vmaub}(A) = f(mn(A))$. Assume conversely that $vmru' \leq au'$, so $\overline{vmaub}(A) = f(1) = \frac{|A|.au' + vmru'}{|A| + 1} \geq mu$ and $vmru' \geq |A|.(mu - au') + mu > mu > au'$. This is a contradiction. After decreasing $vmru'$ by $diffu$, we have the updated value $vmru' := vmru' - diffu = \frac{|A|.(mu - au')}{mn} + mu - 1 < \frac{|A|.(mu - au')}{mn} + mu$. Hence, $\overline{vmaub}(A) = f(mn(A)) = \frac{|A|.au' + mn.vmru'}{|A| + mn} < mu$. □

From Proposition 3, Proposition 4, and the ICS strategy, the paper proposes a novel, efficient strategy named $\mathcal{DUS}_{WUB}$ to hide all FHAUIs based on the WUBs $\overline{vtopmaub}$ or $\overline{vmaub}$. Note that for each FHAUI $A \in FHAUI$, the $\mathcal{DUS}_{WUB}$ strategy hides not only A but also all other FHAU itemsets of $FHAUI$ that are forward extensions of A by considering only A. This is one of the outstanding advantages of the proposed method for hiding FHAUIs. The $\mathcal{DUS}_{WUB}$ strategy is presented as follows.

Decreasing utility strategy based on WUBs ($\mathcal{DUS}_{WUB}$). To hide a FHAU itemset $A \in FHAUI$ based on the WUBs $\overline{vtopmaub}(A)$ or $\overline{vmaub}(A)$, the following steps are repeatedly performed until $\overline{vtopmaub}(A) < mu$ or $\overline{vmaub}(A) < mu$ or $supp(A) < ms$:

(1) Choose the item $a_l \in A$ and the transaction $t_k \in TS(A)$ based on ICS.

(2) Calculate the value of $difu$: if $\overline{vtopmaub}(A) < \overline{vmaub}(A)$, $difu = \lfloor Sum - (|A| + 1) \times mu \rfloor + 1 > 0$; otherwise, $difu = vmru(A) - \frac{|A|.(mu - au(A,TS(A)))}{mn} - mu + 1$.

(3) Reduce the quantity $q_{k,l}$ of the q-item $(a_l, q_{k,l})$ in t_k by $\lceil difu/p(a_l) \rceil$, i.e. the updated value of $q_{k,l}$ is $q_{k,l} := max\{q_{k,l} - \lceil difu/p(a_l) \rceil;\ 0\}$. If $q_{k,l} = 0$, i.e. the q-item $(a_l, q_{k,l})$ is deleted from the transaction t_k, and then $supp(A)$ is also decreased.

(4) Update the value of $\overline{vtopmaub}(A)$ or $\overline{vmaub}(A)$.

For example, let $ms = 1$, $mu = 9.46$, and $HAUI = \{a, c, d, e, f, ac, ad, bd, be, bde, bef, bdef, cd, de, df, def, ef\}$. Consider $A = bd \in FHAUI$, $\rho(A) = 12$, $TS(A) = 1$ with $au(A) = 10.5 \geq mu$. It is observed that bd belongs to the extended lower border of $FHAUI$ w.r.t. $\overline{vtopmaub}$ and $\overline{vmaub}$, $Bd_E^-(FHAUI) = \{a, c, d, e, f, bd, be\}$, because all other itemsets in $FHAUI$ are *forward extensions* of elements in $Bd_E^-(FHAUI)$. We have $\mathcal{V}(A, TS(A)) = (v_j(A, TS(A)),\ a_j \in A \text{ or } j > MaxInd(A)) = (2, 5, 39, 16)$, $\mathcal{V}^*(A, TS(A)) = (v_{l_j}^*(A, TS(A)), j = 1..|A| + 1 = 3) = (v_{l_1=5}^* = 39; v_{l_2=6}^* = 16; v_{l_3=4}^* = 5)$. Thus, $\overline{vtopmaub}(A) = (39 + 16 + 5)/3 = 20 \geq mu$. Similarly, we obtain $\overline{vmaub}(A) = 21.3$.

Using $Bd_E^-(FHAUI)$ and $ICS(\mathcal{DUS}_{WUB})$, we have $iw(a_{l_1}) = SC_{it}(FHAUI, a_{l_1}) = 8$, $iw(a_{l_2}) = SC_{it}(FHAUI, a_{l_2}) = 6$, and $iw(a_{l_3}) = SC_{it}(FHAUI, a_{l_3}) = 9$. Then, by using the ICS strategy, the item $a_l = a_{l_3} = d$ is chosen to decrease the utility, since $iw(a_{l_3}) > iw(a_{l_1}) > iw(a_{l_2})$. Since $|TS(A)| = 1$, the transaction $t_k = t_1$ is also chosen. Since $\overline{vtopmaub}(A) < \overline{vmaub}(A)$, $\overline{vtopmaub}$ is chosen instead of $\overline{vmaub}$. We have $Sum = 60$ and $difu = \lfloor Sum - (|A| + 1) \times mu \rfloor + 1 = \lfloor 60 - (2 + 1) \times 8.6 \rfloor + 1 = 32$. Then, the updated values are: $q_{1,4} = max\{q_{1,4} - \lceil difu/p(a_{l_3}) \rceil;\ 0\} = 0$, $\rho(A) = 2$, $TS(A) = \emptyset$, $\mathcal{V}(A, TS(A)) = 0$. Thus, $\overline{vtopmaub}(A) = 0 < mu$, and thus for *all forward extensions* $F = A \oplus B$ of A ($F = bde$ and $F = bdef$) $au(F) \leq \overline{vtopmaub}(A) < mu$. Also, $au(A) = \frac{4+10}{2} = 7 < mu$. Hence, bd and its forward extensions, bde and $bdef$, are concealed according to Proposition 2, and $r_{tu} = (114 - 109)/114 = 0.044$.

If we use $Bd_E^-(FHAUI)$ without ICS, i.e. the SCS strategy is applied to choose a_l and t_k. Then, $a_l = a_{l_1=5} = e$ because $v_{l_1=5}^*(A, TS(A)) = max\{v_{l_j}^*(A, \rho(A)) | j = 1..3\} = 39$. Then, in the column $v_l(A, TS(A))$, the transaction $t_k = t_1 \in TS(A)$ is selected such that $u(a_l, t_k) = max\{u(a_l, t_j) | t_j \in TS(A)\} = 39$. Next, the updated values are: $q_{1,5} = max\{q_{1,5} - \lceil difu/p(a_{l_1}) \rceil;\ 0\} = max\{13 - \lceil 32/3 \rceil;\ 0\} = 2$, $TS(A) = 1$, $\mathcal{V}(A, TS(A)) = \{2, 5, 6, 16\}$, $\mathcal{V}^*(A, TS(A)) = (v_{l_1=6}^* = 16;\ v_{l_2=5}^* = 6;\ v_{l_3=4}^* = 5)$. Thus, $\overline{vtopmaub}(A) = \frac{16+6+5}{3} = 9 < mu$, so all forward extensions of A, bde and $bdef$, are hidden according to Proposition 3. However, since the chosen item e does not appear in $A = bd$, the average-utility of A, $au(A) = 10.5$, remains equal to its old value, which is greater than mu. To hide $A = bd$ and $\rho(A) = 12$, we select the pair $(a_l, t_k) = (a_4 = d, t_2)$ and $difu(A) = \lfloor u(A) - |A| \times mu \rfloor + 1 = \lfloor 21 - 2 \times 9.46 \rfloor + 1 = 3$. Then, we obtain the updated values $q(a_l, t_k) = 7$, $\rho(A) = 12$ and $au(A) = 9 < mu$, and $r_{tu} = \frac{114-78}{114} = 0.316$ (> 0.044). It is found that when using $Bd_E^-(FHAUI)$ and $ICS(\mathcal{DUS}_{WUB})$, the chosen item d appears in 9 patterns of $FHAUI$, while using $Bd_E^-(FHAUI)$ and SCS, the chosen item e, which is not contained in $A = bd$, appears only in 8 patterns of $FHAUI$.

In that example, if the baseline method is used to hide all itemsets bd, bde and $bdef$ one by one, we obtain $r_{tu} = (114 - 84)/114 = 0.263$. And if we use the border-based approach and $\mathcal{DUS}_{WUB}$, we obtain a value of r_{tu} that is much smaller ($r_{tu} = 0.044$) than the baseline approach and also than if we utilize the border-based approach without using the ICS strategy ($r_{tu} = 0.316$).

4.3. The novel TIU-VIU structure

To hide an itemset A based on $\mathcal{DUS}_{WUB}$, it is found that the $\overline{vmaub}$ and $\overline{vtopmaub}$ WUBs must be regularly updated during the hiding process. In addition, to decide which strategy, $\mathcal{IDSS}$ or $\mathcal{DUS}_{WUB}$, should be applied for the process to ensure that the quality of the QTDB $\mathcal{D}$ is minimally decreased, the value of Bd_{max} is also frequently re-calculated. Therefore, a serious issue is how to quickly update the values of $\overline{vmaub}$, $\overline{vtopmaub}$ and Bd_{max} of an itemset. To address this issue, this section proposes a novel structure called TIU-VIU (Transaction Itemset Utility – Vertical Item Utility).

Definition 10 (*The vertical utility list of an itemset - VUL*). The vertical utility list (VUL) of an itemset is a list of elements, where each element contains two fields, named it and vu. For an itemset A, the fields of an element are defined as follows:

- The it field stores an item such that $it \in S \stackrel{\text{def}}{=} (a_k,\ a_k \in A\ or\ k > maxInd(A))$.
- The vu field stores the vertical sum of the utility of the item it in transactions that belong to $TS(A)$, i.e. $vu = \sum_{t_i \in TS(A)} u(it, t_i)$.

Definition 11 (*The remaining item number and transaction utility list – RINTUL*). For an itemset A, the remaining item number and transaction utility list (RINTUL) of an itemset A is a list of elements, where each element consists of three fields, defined as follows:

- The tid field indicates a transaction that belongs to the set $\rho(A)$ of all transactions containing A.
- The tu field indicates the transaction utility of A in tid, i.e. $tu = u(A, tid)$.
- The rin field is the remaining item number in tid after $a_{MaxInd(A)}$, i.e. $rin = |\{a_k \in tid \mid k > MaxInd(A)\}|$.

Based on the lists *VUL* and *RINTUL* of Definitions 10 and 11, respectively, we propose the following structure.

Definition 12 (*The transaction itemset utility and vertical item utility structure – TIU-VIU*). The transaction itemset utility and vertical item utility structure (TIU-VIU) of an itemset A includes four fields (u, *VUL*, *RIC-TUL* and mn), defined as follows:

- The u field stores the value of $u(A, TS(A))$.
- The VUL field contains the VUL list of the itemset A.
- The $RINTUL$ field indicates the RINTUL list of A.
- The $mrin$ field is the maximum remaining item number in $\mathcal{D}$ after the last item $a_{maxInd(A)}$ according to the set of transaction identifiers in the $RINTUL$ list, i.e. $mrin = max\{RINTUL.rin\}$, where $RINTUL.rin$ denotes the set of all values of the rin field in the $RINTUL$ list.

The main difference between the NVUV-list structure [38] and the proposed TIU-VIU structure is that the latter stores two lists of *VUL* and *RINTUL*, while the former only stores a vector *vsuv*. These two lists are very useful during the hiding process as they help to quickly update the values of $\overline{vmaub}$, $\overline{vtopmaub}$, and Bd_{max}. Rather than re-calculating the vector $\mathcal{V}(A, TS(A))$, $\overline{vmaub}$ and $\overline{vtopmaub}$, only the necessary information must be updated. In addition, the VUL list in the TIU-VIU structure is an extension of the *vsuv* vector by adding the it field to facilitate updating the $\overline{vmaub}$ and $\overline{vtopmaub}$ WUBs.

For example, consider the *integrated* QTDB $\mathcal{D} = \{t_1: b2, d5, e39, f16; t_2: b4, d10, f2\}$ that has two transactions, where $b2$ in t_1 means that the utility of item b in transaction $t_1$is 2, i.e. $u(b, t_1) = 2$. For an itemset $A = bd$, let T be the *TIU-VIU* structure of A. Then, the $T.VUL$ and $T.RINTUL$ of A are depicted in Fig. 5. The first element of $T.VUL$ is (b, 6), i.e. the vertical sum utility of the item $it = b$ on transactions in $TS(A) = 12$ is 6. Similarly, the first element (t_1, 7, 2) of $T.RINTUL$ means that the transaction utility of A for the transaction t_1 is 7 and the remaining item number for t_1 after $MaxInd(A) = 4$ is 2. In addition, we also have $T.u = 21$ and $T.mrin = max\{T.RINTUL.rin\} = 2$.

Next, we illustrate how to quickly update the values of $\overline{vmaub}$, $\overline{vtopmaub}$, and Bd_{max}. For $mu = 8.6$, $ms = 1$ and $A = bd$, we have $\rho(A) = TS(A) = 12$, $au(A) = 10.5 > mu$ and $supp(A) = 2 > ms$, and thus itemset A is a FHAUI. Assume that we need to hide the itemset A. Then, based on the *VUL* of A in Fig. 5, we obtain $\mathcal{V}(A, TS(A)) = (T.VUL.vu) = (6, 15, 39, 18)$ and $\mathcal{V}^*(A, TS(A)) = (v^*_{l_1=5} = 39; v^*_{l_2=6} = 18; v^*_{l_3=4} = 15)$. Therefore, $\overline{vtopmaub}(A) = (39 + 18 + 15)/3 = 24 \geq mu$. Similarly, based on the *RINTUL* of A, we also have $mn(A) = T.mrin = max\{T.RINTUL.rin\} = 2$ and $Bd_{max}(A) = 14 + 7 = 21$. Since $vmru(A,\ TS(A)) = max\{T.VUL.vu, Index(T.VUL.it) > MaxInd(A)\} = 39 > au(A, TS(A)) = 10.5$, so $\overline{vmaub}(A) = f(mn(A)) = \frac{|A|.au(A,\ TS(A)) + mn(A).vmru(A,TS(A))}{|A|+mn(A)} = \frac{2 \text{ x } 10.5 + 2 \text{ x } 39}{2+2} = 24.75$. By using the ICS strategy, the item $a_l = d$ and the transaction $t_k = t_1$ are chosen for hiding A. Then, to hide A based on the $\mathcal{DUS}_{WUB}$ strategy, the following two cases are considered:

VUL of bd	
it	vu
b	6
d	15
e	39
f	18

RINTUL of bd		
tid	tu	rin
t_1	7	2
t_2	14	1

Fig. 5. The *VUL* and *RINTUL* of the itemset bd.

If $difu(A) < u(a_l, t_k)$, then to quickly update $\overline{vmaub}$, $\overline{vtopmaub}$, and Bd_{max}, we only need to update the following values: $T.u = T.u - difu(A)$, $T.VUL.vu_{a_l} = T.VUL.vu_{a_l} - difu(A)$ and $T.RINTUL.tu_{t_k} = T.RINTUL.tu_{t_k} - difu(A)$, where $T.VUL.vu_{a_l}$ is the value of $T.VUL.vu$ at $it = a_l$ and $T.RINTUL.tu_{t_k}$ is the value of $T.RINTUL.tu$ at $tid = t_k$.

If $difu(A) \geq u(a_l, t_k)$, then the following values are updated: $T.u = T.u - T.RINTUL.tu_{t_k}$, $T.RINTUL.tu_{t_k} = T.RINTUL.rin_{t_k} = 0$ and $T.VUL.vu_a = T.VUL.vu_a - u(a, t_k)$, for $\forall a \in T.VUL.it$. Thus, the values of $\overline{vmaub}$, $\overline{vtopmaub}$, and Bd_{max} are quickly updated. For instance, since $\overline{vtopmaub}(A) = 24 < \overline{vmaub}(A) = 24.75$, we have $difu(A) = \lfloor Sum - (|A| + 1) \times mu \rfloor + 1 = \lfloor 72 - (2 + 1) \times 8.6 \rfloor + 1 = 47 > u(a_l, t_k) = 5$. Then, $T.u = T.u - T.RINTUL.tu_{t_k} = 21 - 7 = 14$, $T.VUL.vu_b = 6 - 2 = 4$, $T.VUL.vu_d = 15 - 5 = 10$, $T.VUL.vu_e = 39 - 39 = 0$ and $T.VUL.vu_f = 18 - 16 = 2$ (see Fig. 6). Thus, $\mathcal{V}(A, TS(A)) = (T.VUL.vu) = (4, 10, 0, 2)$ and $\mathcal{V}^*(A, TS(A)) = (10, 4, 2)$, so $\overline{vtopmaub}(A) = (10 + 4 + 2)/3 = 5.33$. Since $vmru(A,\ TS(A)) = 2 < au(A, TS(A)) = T.u/2 = 7$, it is found that $\overline{vmaub}(A) = \frac{|A|.au(A,\ TS(A)) + vmru(A,TS(A))}{|A|+1} = \frac{2 \text{ x } 7 + 2}{2+1} = 5.33$. Also, $Bd_{max}(A)$ is quickly updated to 17 based on the *RINTUL* of A.

VUL of bd	
it	vu
b	6
d	15
e	39
f	18

VUL of bd	
it	vu
b	4
d	10
e	0
f	2

RINTUL of bd		
tid	tu	rin
t_1	7	2
t_2	14	1

RINTUL of □□		
tid	tu	rin
t_2	14	1

(*a*) The *VUL* of bd (*b*) The *RINTUL* of bd

Fig. 6. The updated *VUL* and *RINTUL* of itemset bd.

4.4. Proposed H-FHAUI algorithm

Based on the above theoretical results and data structure, we present a novel algorithm named H-FHAUI (Hiding Frequent High Average-Utility Itemsets) for hiding all FHAUIs of $FHAUI$. The pseudo-code of H-FHAUI is shown in Fig. 7 (see Table 6 in the appendix for the meaning of notation and abbreviation used in this algorithm). H-FHAUI takes as input a QTDB $\mathcal{D}$, the mu and ms thresholds, and a set $FHAUI$ of FHAUIs. First, the algorithm finds the extended lower border of $FHAUI$ (line 1). For each itemset A in the $Bd_E^-(FHAUI)$ set, the values of $difs(A)$, $difu(A)$ and $Bd_{max}(A)$ are calculated (lines 4-9). Then, the algorithm checks the condition $Bd_{max}(A) \leq difu(A)$ in Remark 1 to decide which strategy, between $I\mathcal{DSS}$ or $\mathcal{DUS}_{WUB}$, should be used (lines 10-12). Next, the algorithm applies the ICS strategy to choose the item a_l and transaction t_k for decreasing the support or utility of A (line 13). Then, it calls the UpdateValues procedure (see Fig. 2) to update the values of $supp(A)$, $u(A)$, $au(A)$ and $q(a_l, t_k)$ (line 14). Finally, the average-utility and support of all remaining itemsets of $Bd_E^-(FHAUI)$, and the values of weak upper bounds are also updated (lines 15-16). The algorithm stops if all FHAUIs of $Bd_E^-(FHAUI)$ are hidden, and thus also those of $FHAUI$. The result is that the QTDB $\mathcal{D}$ is transformed into a sanitized QTDB $\mathcal{D}'$. Note that if the condition $au(A) \geq mu$ still holds, the algorithm calls the procedure HidingOneItemset in Fig. 1 to hide an itemset A based on the baseline method where the ICS strategy is used.

H-FHAUI($\mathcal{D}$, mu, ms, $FHAUI$)
Input: The original QTDB $\mathcal{D}$ represented in *vertical* form, the thresholds mu and ms, the set $FHAUI$ of all FHAU itemsets.
Output: The *sanitized* QTDB $\mathcal{D}'$.

1. Find the subset $Bd_E^-(FHAUI)$ of $FHAUI$;
2. **for each** itemset $A \in Bd_E^-(FHAUI)$ **do** {
3. **while** ($supp(A) \geq ms$ and $\overline{vmaub}(A) \geq mu$ and $\overline{vtopmaub}(A) \geq mu$) **do** {
4. $difs(A) = supp(A) - ms + 1$;
5. **if** ($\overline{vmaub}(A) > \overline{vtopmaub}(A)$) **then** //*using WUB* $\overline{vtopmaub}$
6. $difu(A) = \lfloor Sum - (|A| + 1) \times mu \rfloor + 1$;
7. **else** //*using WUB* $\overline{vmaub}$
8. $difu(A) = vmru(A) - \frac{|A|.(mu - au(A,TS(A)))}{mn} - mu + 1$;
9. Calculate the maximum boundary value $Bd_{max}(A)$; //*Definition 7*
10. **if** ($Bd_{max}(A) \leq difu(A)$) **then** //*Remark* 1
11. $I\mathcal{DSS}$ = true; //*using* $I\mathcal{DSS}$ *strategy*
12. **else** $I\mathcal{DSS}$ = false; //*using* $\mathcal{DUS}_{WUB}$ *strategy*
13. Choose the item $a_l \in A$ and the transaction $t_k \in TS(A)$ based on ICS;
14. **UpdateValues**(A, $difu(A)$, a_l, t_k, $I\mathcal{DSS}$).
15. Update the au and $supp$ of all remaining itemsets of $Bd_E^-(Sen)$;
16. Update the values of WUBs $\overline{vtopmaub}(A)$ and $\overline{vmaub}(A)$;
17. } //*end while*
18. **if** ($au(A) \geq mu$) **then**
19. **HidingOneItemset** (A, $\mathcal{D}$, mu, ms, Sen, $FHAUI_r$, ICS);
20. } //*end for*

Fig. 7. The H-FHAUI algorithm.

Discussion on the complexity of the two H-FHAUI-B and H-FHAUI algorithms (in terms of time). Since calculating the time complexity of the two algorithms is quite complex for the general case, we only discuss the complexity in the worst case. For two given thresholds mu and ms, let $N = |FHAUI|$ be the number of all FHAUIs to be hidden, $n = |\mathcal{D}|$ and $m = |\mathcal{A}|$.

The time complexity of **H-FHAUI-B** depends on the number of executions of the strategies $\mathcal{DSS}$ or $\mathcal{DUS}$ to be selected in the *while* loop (line 1 of Fig. 1) of the HidingOneItemset procedure. For each itemset $A \in FHAUI$, assume that $\mathcal{DSS}$ is always selected. Then, the number of executions of this loop will be $difs \stackrel{\text{def}}{=} n - ms + 1$ in the worst case

(i.e. all items of $\mathcal{A}$ always appear in every transaction of $\mathcal{D}$). Then, the complexity in the first situation is $O(N \times (n - ms))$. Similarly, assume that $\mathcal{DUS}$ is always chosen. Then, the number of executions of this loop will also be $difs$ (in the worst case, $q(a_l, t_k)$ is always decreased to 0 and $supp(A) = supp(A) - 1$ for each step of the loop). Therefore, the complexity in the second situation is also $O(N \times (n - ms))$. Thus, we can conclude that in the worst case, the complexity of **H-FHAUI-B** is $O(N \times (n - ms))$.

In the same way, let $N' = |Bd_E^-(FHAUI)|$ be the number of patterns in the border $Bd_E^-(FHAUI)$, which is a subset of $FHAUI$, to be hidden. In the worst case, the complexity of **H-FHAUI** is $O(N' \times (n - ms))$. Since N' is generally much less than N, and by using the improved selection strategy ICS instead of SCS, the chosen items (or transactions) usually appear (or contain, respectively) many other patterns FHAUIs, and the **H-FHAUI** algorithm has a performance that is much better than the baseline **H-FHAUI-B** algorithm.

5. Experimental results

To evaluate the performance of the proposed H-FHAUI algorithm, extensive experiments were carried out on a computer equipped with an Intel(R) Core (TM) i5-6200U 2.3 GHz CPU with 16 GB of memory, running Windows 10. The performance of H-FHAUI was compared with the baseline H-FHAUI-B algorithm for hiding all frequent high average utility itemsets. Both algorithms were implemented in Java 7 SE.

In the experiments, three real-life datasets, namely Mushroom, Chess, and Foodmart, and four synthetic datasets, namely T11I4D40K, T25I10D10K, T20I10D18K and T25I12D20K, were used. The real-life datasets can be obtained from [6] and the synthetic ones have been generated using the IBM Quest data generator (obtained from [6]) with the parameters described in Table 4. Characteristics of the datasets are shown in Table 5.

Table 4. Parameters of the IBM Quest Synthetic Data Generator.

Parameters	Meaning
T	Average transaction length
I	Average size of the maximal potentially frequent itemsets
N	Number of distinct items
D	Number of transactions (in thousands) in the dataset

Table 5. Characteristics of the datasets.

Dataset	No. of trans.	No. of items	Average trans. length	Max trans. Length	Type of data
Mushroom	8,124	120	23	23	Real-life
Chess	3,196	75	35	35	Real-life
Foodmart	4,141	1,559	4.42	11	Real-life
T11I4D40K	40,000	200	5.9	11	Synthetic
T25I10D10K	10,000	600	13	25	Synthetic
T25I12D20K	20,000	500	10.5	25	Synthetic
T20I10D18K	18,000	750	12.6	20	Synthetic

Note that the mu and ms thresholds used in the experiments are defined as percentages of the total utility and transaction number of a dataset, respectively. Moreover, in this section, to evaluate the performance of the two proposed algorithms (the baseline H-FHAUI-B and improved H-FHAUI) for hiding FHAUIs using different strategies to decrease the utility and support of hidden patterns, besides high mu and ms values, low mu and ms values are also considered. The reason is that the number of FHAUIs to be hidden in that case is often very large. Thus, algorithms need to decrease very large quantities of utility (internal utility) of items in many transactions (or even these quantities can be decreased to zero), and then, as a result, the quality or total utility of the QTDB can be reduced significantly.

5.1. Influence of the novel ICS strategy

To evaluate the efficiency of the novel ICS strategy in the proposed H-FHAUI algorithm, two cases were considered: applying H-FHAUI (1) without the ICS strategy (**H-FHAUI-Non**) and (2) with it (**H-FHAUI**). Experiments were performed on the real-life Chess dataset and the synthetic T20I10D18K dataset for different mu values. For each mu threshold value, algorithm and dataset, the runtime and ratio r_{tu} (%) (in Definition 5) were noted. Results are shown in Fig. 8.

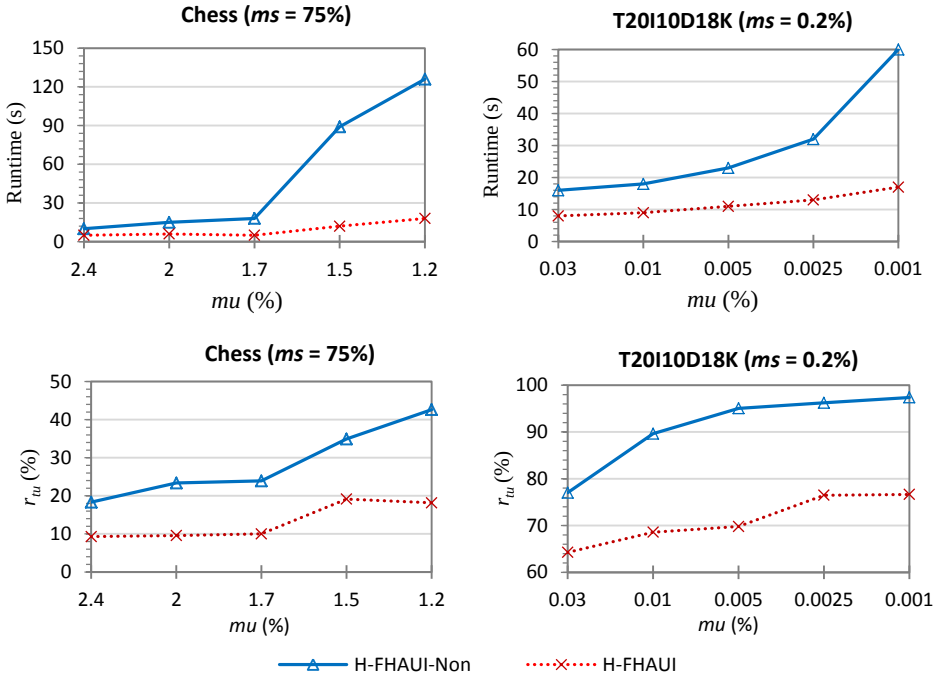


Fig. 8. Influence of the novel ICS strategy on the runtime and r_{tu} ratio.

It is observed in Fig. 8 that the runtime and r_{tu} ratio of H-FHAUI are much less than those of H-FHAUI-Non. This is reasonable because the ICS strategy, which is integrated into the H-FHAUI algorithm, is based on information about FHAU itemsets in the $FHAUI$ set. When hiding each itemset using ICS, many of the remaining itemsets that have not yet been considered can be concealed without additional calculations, thereby reducing the runtime for hiding FHAUIs. Moreover, choosing an item and a transaction based on weights in the ICS strategy to decrease utilities can help to preserve the total utility of the original QTDB as much as possible. Thus, the r_{tu} ratio of H-FHAUI is smaller than that of H-FHAUI-Non. This demonstrates the effectiveness of ICS.

5.2. Influence of using the minimum support threshold ms

Note that the proposed method for hiding FHAUIs using both parameters mu and ms is more general than the traditional method that conceals HAUIs using only mu (i.e. the ms value is then always equal to 1). Thus, to compare the efficiency of these two methods, this sub-section evaluates the performance of the H-FHAUI algorithm for two cases of $ms = 1$ and $ms > 1$ on two QTDBs Chess and T20I10D18K. Results are shown in Fig. 9.

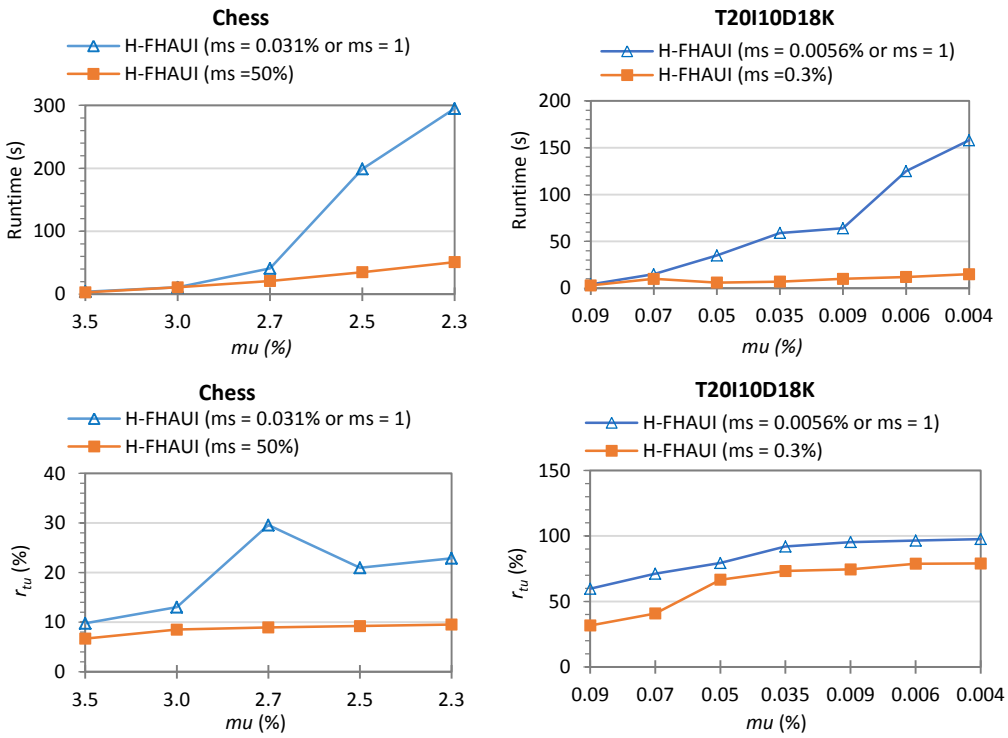


Fig. 9. The performance of the H-FHAUI algorithm with $ms = 1$ and $ms > 1$.

It is observed from Fig. 9 that the runtime and r_{tu} ratio of the H-FHAUI algorithm for the case of $ms > 1$ (the ms values in percentage are 50% on Chess and 0.3% on T20I10D18K) are often less than that for the case of $ms = 1$ on both the QTDBs for all the tested mu values. The reason is that when considering the values $ms > 1$, decreasing the support of a FHAUI to a value that is less than ms (i.e. the FHAUI is hidden) is more quickly performed. As a result, the total utility of the QTDB is reduced less, compared to the case of $ms = 1$.

5.3. Performance evaluation of the proposed H-FHAUI algorithm

This subsection compares the performance of the proposed H-FHAUI algorithm with that of the baseline H-FHAUI-B algorithm on three real-life datasets and three synthetic datasets for various values of the mu and ms parameters. On each dataset, various ranges of mu and ms values have been tested, revealing changes in terms of runtime and peak memory usage for the algorithms.

5.3.1. Execution time

Fig. 10 and Fig. 11 show the execution time of the algorithms when the mu and ms parameters are varied, respectively. These figures show that the algorithms have longer runtimes when parameters are decreased. This is reasonable because the number of FHAUIs increases for lower mu and ms values. It is also observed in these figures that the H-FHAUI algorithm is from one to two orders of magnitude faster than the baseline H-FHAUI-B algorithm for all tested datasets, especially for low mu and ms values. For example, by computing the average runtimes of the algorithms for all tested mu values in Fig. 10 on the Mushroom and T25I10D10K datasets, H-FHAUI is about 24 and 26 times faster than H-FHAUI-B, respectively. Similarly, for all tested ms values in Fig. 11, the corresponding numbers are 29 and 28 times. On the Mushroom dataset, for the lowest values of mu (0.3% in Fig. 10) and ms (8% in Fig. 11), H-FHAUI is about 44 and 90 times faster than H-FHAUI-B, respectively. The reason is that H-FHAUI adopts a border-based approach that hides a small subset of all FHAUIs to hide all of them. Thus, the search space of the H-FHAUI algorithm can be much smaller than that of H-FHAUI-B, which leads to shorter runtimes.

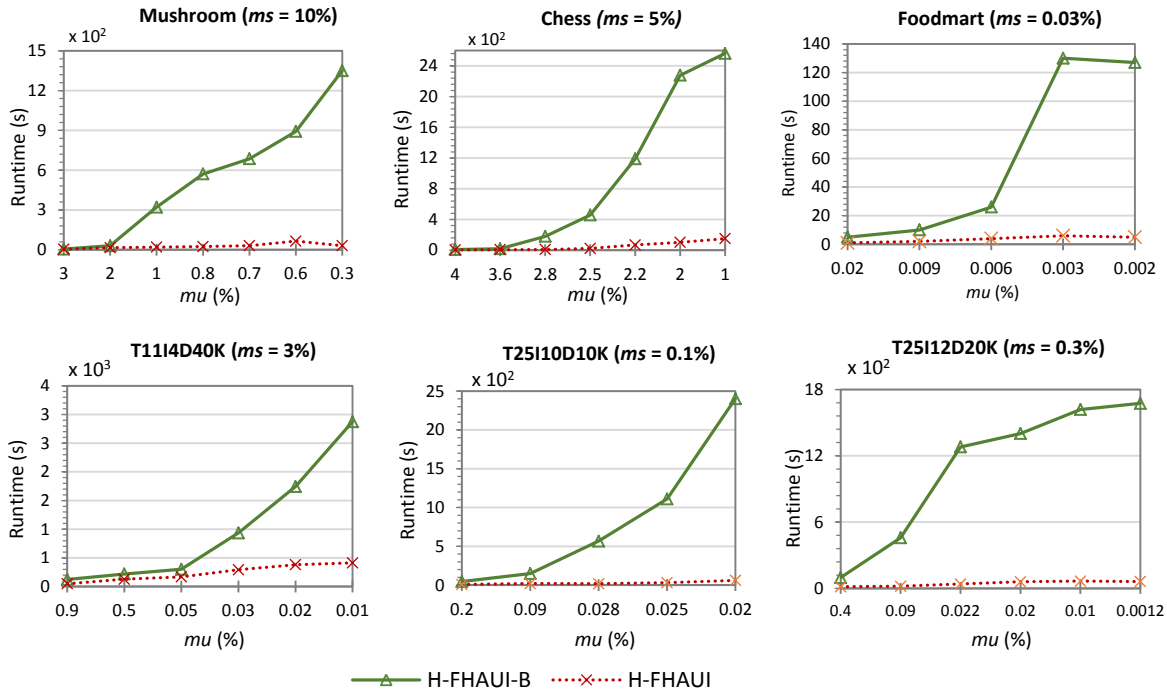


Fig. 10. Runtimes of the algorithms when *mu* is varied.

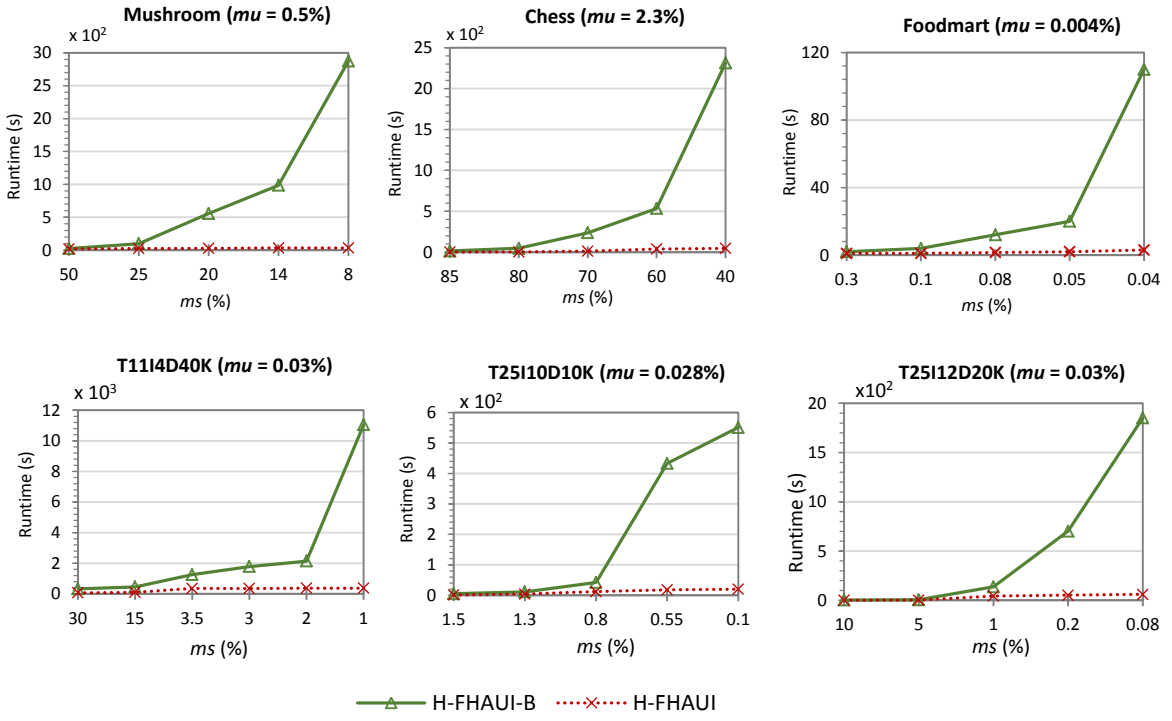


Fig. 11. Runtimes of the algorithms when *ms* is varied.

5.3.2. Memory usage

To assess memory consumption, we recorded the peak memory usage of the algorithms in megabytes when they were run using the *mu* and *ms* values of previous experiments, on all datasets. Results are reported in Fig. 12 and Fig.

13. It is observed that the memory consumption of H-FHAUI is less than that of H-FHAUI-B on all the datasets. The reason is that the number of FHAUIs considered for hiding by H-FHAUI is often less than that of H-FHAUI-B. The reason is that, as explained above, H-FHAUI only considers a small subset of the *FHAUI* set by using the concept of extended lower border w.r.t. forward extension. In addition, for each FHAU itemset to be hidden based on the weak upper-bound, the number of items and transactions considered by the H-FHAUI algorithm is small, but many forward extensions of that itemset can be concealed, leading to consuming a smaller amount of memory for H-FHAUI. For example, in Fig. 12, on the T25I12D20K dataset with $ms = 0.3\%$ and $mu = 0.0012\%$, the peak memory consumption of H-FHAUI-B is about 18 times larger than that of H-FHAUI (723 MB instead of 41 MB).

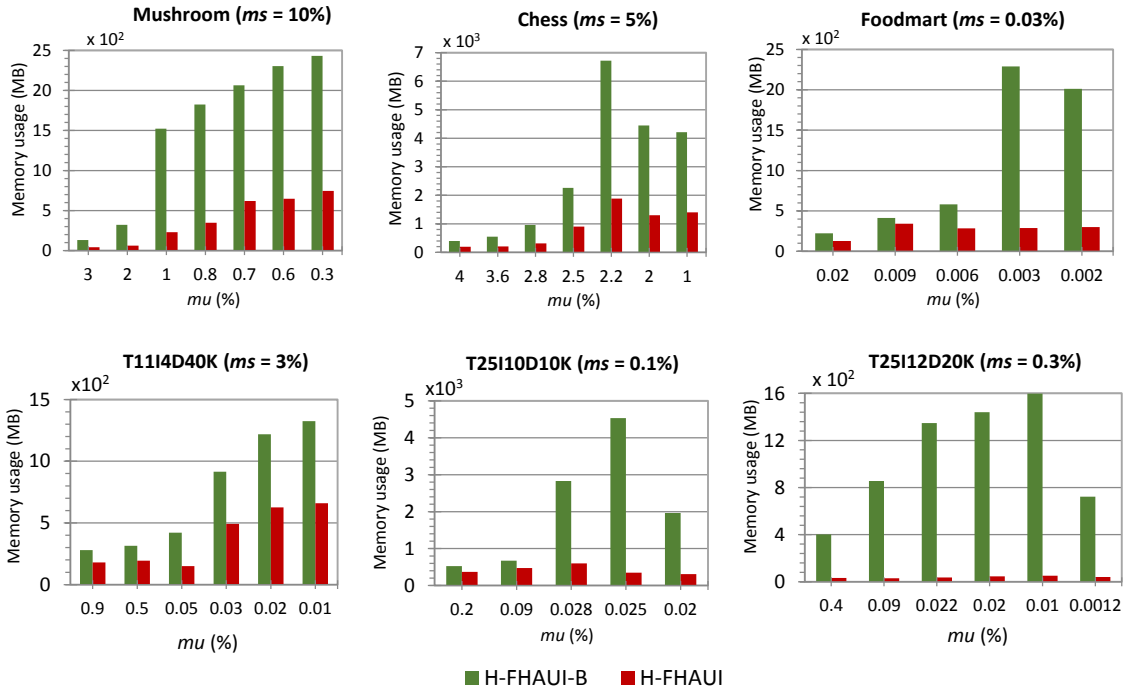
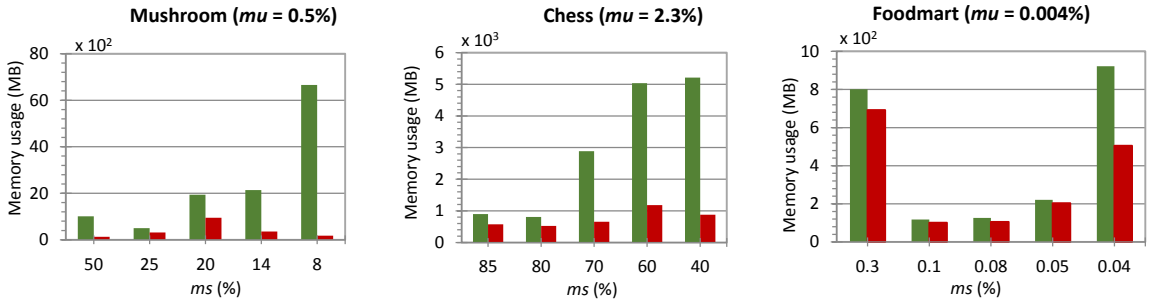


Fig. 12. Memory usage of the algorithms when *mu* is varied.



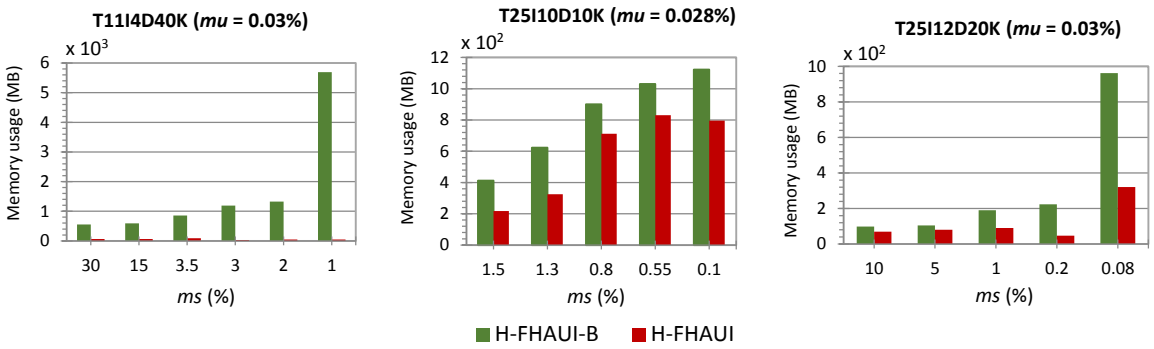


Fig. 13. Memory usage of the algorithms when ms is varied.

5.3.3. The r_{tu} ratio

The performance of the two algorithms has also been compared in terms of the r_{tu} ratio. It can be observed in Fig. 14 and Fig. 15 that the r_{tu} ratios of the H-FHAUI algorithm are much less than those of H-FHAUI-B for the tested mu and ms values on all datasets. The reason is that H-FHAUI utilizes the Bd_E^- extended lower border to reduce the number of itemsets that the algorithm will try to hide, instead of considering all itemsets of the $FHAUI$ set. This can remarkably reduce changes in the utility of items in the datasets. Moreover, H-FHAUI also uses the proposed novel strategies, ICS and $\mathcal{DUS}_{WUB}$, to ensure that the quality of datasets is minimally decreased during the hiding process. For example, in Fig. 15 for the Chess dataset and $mu = 2.3\%$, the r_{tu} ratios of H-FHAUI are about 2 to 5 times less than those of H-FHAUI-B.

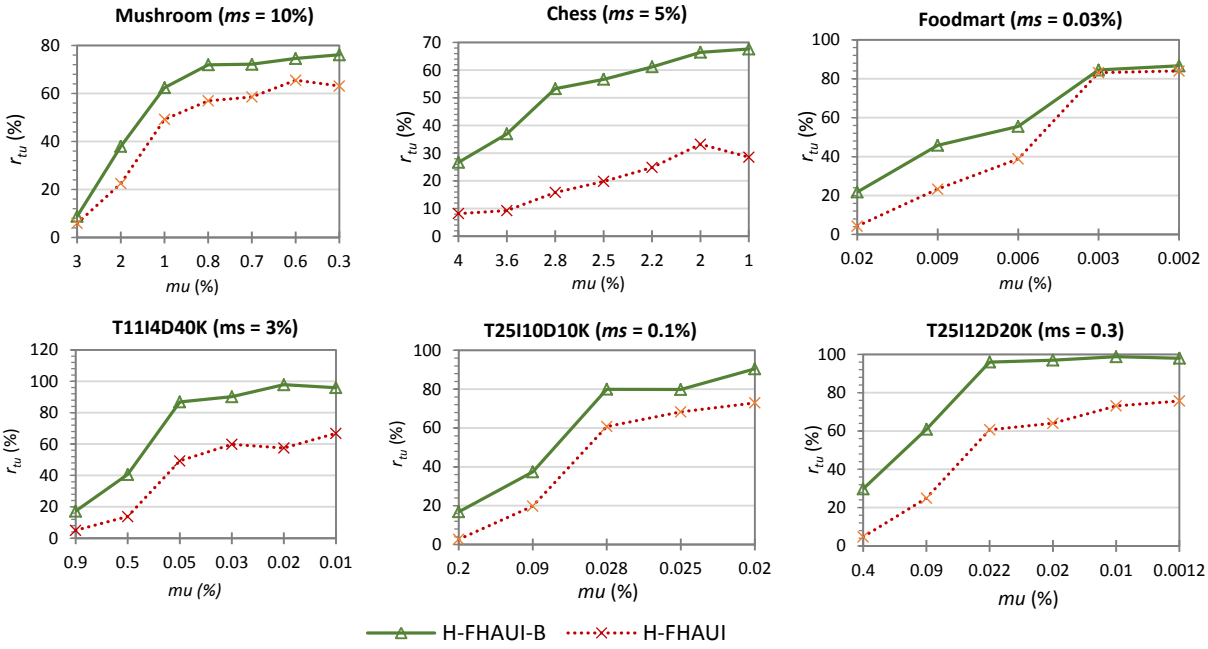


Fig. 14. The r_{tu} ratio of the algorithms when mu is varied.

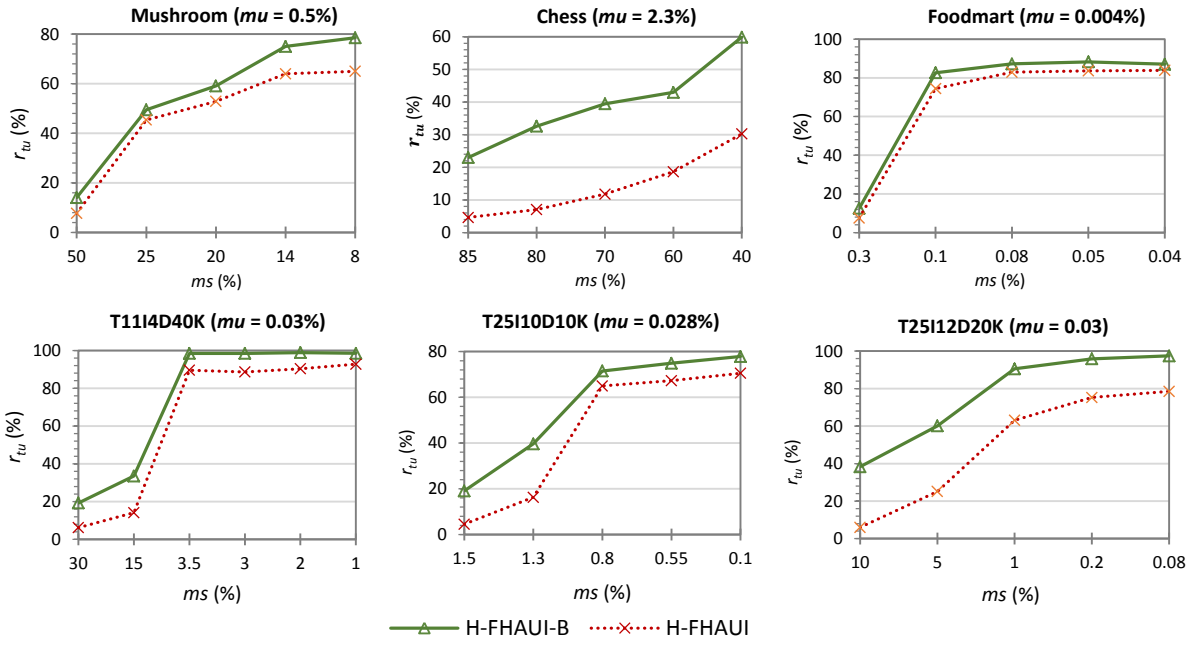


Fig. 15. The r_{tu} ratio of the algorithms when ms is varied.

It is also observed in Fig. 14 and Fig. 15 that for the highest ms and mu values on the tested datasets, the r_{tu} ratios of the H-FHAUI algorithm are very low (always less than 10%), while those of the H-FHAUI-B algorithm are quite high (up to 38%). For example, in Fig. 14 for the highest value $mu = 4\%$ on Chess, the r_{tu} ratios of H-FHAUI and H-FHAUI-B are respectively 8.20 and 26.67. Similarly, in Fig. 15, for the highest value $ms = 10\%$ on T25I12D20K, the r_{tu} ratios of the two algorithms are 5.97 and 38.34.

Overall, the above experiments have shown that the proposed H-FHAUI algorithm outperforms the baseline H-FHAUI-B algorithm in terms of runtime, peak memory consumption, and r_{tu} ratio. These empirical evidences demonstrate that the novel theoretical results used in H-FHAUI are meaningful and considerably improve the performance for hiding frequent high average-utility itemsets.

6. Conclusions and future work

This paper has studied the problem of hiding all FHAUIs, and presented an extended lower border Bd_E^- based on weak upper bounds to reduce the size of the set of FHAUIs that must be hidden. Thus, the number of itemsets that the algorithm considers for hiding is much less than that of the baseline algorithm. The paper has also proposed two novel strategies, named ICS and $\mathcal{DUS}_{WUB}$, to quickly hide all FHAUIs and to ensure that the quality of the original QTDB can be minimally decreased. Besides, to quickly update values of weak upper bounds on au, a novel structure named TIU-VIU was also proposed. Based on these theoretical results and data structure, a novel algorithm, named H-FHAUI, has been developed to efficiently hide all FHAUIs for privacy-preserving data mining. Experiments conducted on several real-life and synthetic databases have shown that H-FHAUI is not only one to two orders of magnitude faster than the baseline H-FHAUI-B algorithm, but also consumes less memory and preserves more the database utility.

In the future, the proposed theoretical results will be extended for the more general problem of hiding high utility or high average utility sequential patterns in quantitative sequence databases for privacy-preserving data mining.

Acknowledgment

This research is funded by Vietnam National Foundation for Science and Technology Development (NAFOSTED) under grant number 102.05-2018.307.

Appendix

For the convenience of readers, Table 6 summarizes some notations for the main strategies, weak upper bounds on the average utility, (extended) lower and upper borders of a set U of itemsets, which are used in this article.

Table 6. Notation and abbreviation.

Notation	Meaning	Algorithm using notation
SCS	Simple item-transaction choosing strategy	H-FHAUI-B
$\mathcal{DSS}$	Decreasing support strategy	H-FHAUI-B
$\mathcal{DUS}$	Decreasing utility strategy	H-FHAUI-B
ICS	Improved item-transaction choosing strategy	H-FHAUI
$IDSS$	Improved decreasing support strategy	H-FHAUI
$\mathcal{DUS}_{WUB}$	Decreasing utility strategy based on weak upper bounds	H-FHAUI
$\overline{vmaub}$	A weak upper bound on the average utility	H-FHAUI
$\overline{vtopmaub}$	Another weak upper bound on the average utility	H-FHAUI
$Bd^{-}(U)$	The lower border of the U set of itemsets	H-FHAUI
$Bd^{+}(U)$	The upper border of the U set of itemsets	H-FHAUI
$Bd_{E}^{-}(U)$	The extended lower border of the U set of itemsets	H-FHAUI

References

[1] Charu C. Aggarwal and Philip S. Yu. 2008. A General Survey of Privacy-Preserving Data Mining Models and Algorithms. In *Proceedings of C.C. Aggarwal, P.S. Yu (Eds.), Privacy-Preserving*, 11–52.

[2] Rakesh Agrawal and Ramakrishnan Srikant. 2000. Privacy preserving data mining. *ACM Sigmod Rec.* 29, (2000), 439–450.

[3] Elisa Bertino, Dan Lin, and Wei Jiang. 2008. A Survey of Quantification of Privacy Preserving Data Mining Algorithms. In *Proceedings of C.C. Aggarwal, P.S. Yu (Eds.), Privacy-Preserving Data Mining: Models and Algorithms*, 183–205.

[4] Raymond Chan, Qiang Yang, and Yi-Dong Shen. 2003. Minging high utility itemsets. In *Proceedings of IEEE International Conference on Data Mining*, 19–26.

[5] Alexandre Evfimievski, Ramakrishnan Srikant, Rakesh Agrawal, and Johannes Gehrke. 2004. Privacy preserving mining of association rules. *Inf. Syst.* 29, 4 (2004), 343–364.

[6] Philippe Fournier-Viger, Jerry Chun-Wei Lin, A Gomaris, T Gueniche, A. Soltani, Z. Deng, and H. T. Lam. 2014. SPMF: a Java Open-Source Pattern Mining Library Version 2. *Mach. Learn. Res.* 15, 1 (2014), 3389–3393.

[7] Wensheng Gan, Jerry Chun-Wei, Han Chieh Chao, Shyue Liang Wang, and Philip S. Yu. 2019. Privacy Preserving Utility Mining: A Survey. In *Proceedings of IEEE International Conference on Big Data, Big Data*, 2617–2626.

[8] Fosca Giannotti, Laks V.S. Lakshmanan, Anna Monreale, Dino Pedreschi, and Hui Wang. 2013. Privacy-preserving mining of association rules from outsourced transaction databases. *IEEE Syst. J.* 7, 3 (2013), 385–395.

[9] Tzung-Pei Hong, Cho Han Lee, and Shyue Liang Wang. 2009. Mining high average-utility itemsets. In *Proceedings of IEEE International Conference on Systems, Man and Cybernetics*, 2526–2530.

[10] Tzung-Pei Hong, Cho Han Lee, and Shyue Liang Wang. 2011. Effective utility mining with the measure of average utility. *Expert Syst. Appl.* 38, 7 (2011), 8259–8265.

[11] Tzung-Pei Hong, Jerry Chun-Wei Lin, Kuo Tung Yang, and Shyue Liang Wang. 2013. Using TF-IDF to hide sensitive itemsets. *Appl. Intell.* 38, 4 (2013), 502–510.

[12] Yogendra Kumar Jain, Vinod Kumar Yadav, and GS Geetika S Panday. 2011. An Efficient Association Rule Hiding Algorithm for Privacy Preserving Data Mining. *Comput. Sci. Eng.* 3, 7 (2011), 2792–2798.

[13] Guo Cheng Lan, Tzung-Pei Hong, and Vincent S. Tseng. 2012. Efficiently Mining High Average-Utility Itemsets With an Improved Upper-Bound Strategy. *Inf. Technol. Decis. Mak.* 11, 05 (2012), 1009–1030.

[14] Guo Cheng Lan, Tzung-Pei Hong, and Vincent S. Tseng. 2014. An efficient projection-based indexing approach for mining high utility itemsets. *Knowl. Inf. Syst.* 38, 1 (2014), 85–107.

[15] Bac Le, Duy Tai Dinh, Van Nam Huynh, Quang Minh Nguyen, and Philippe Fournier-Viger. 2018. An efficient algorithm for Hiding High Utility Sequential Patterns. *Int. J. Approx. Reason.* 95, (2018), 77–92.

[16] Yu Chiang Li, Jieh Shan Yeh, and Chin Chen Chang. 2007. MICF: An effective sanitization algorithm for hiding sensitive patterns on data mining. *Adv. Eng. Informatics* 21, 3 (2007), 269–280.

[17] Jerry Chun-Wei Lin, Tzung-Pei Hong, Philippe Fournier-Viger, Qiankun Liu, Jia Wei Wong, and Justin Zhan. 2017. Efficient hiding of confidential high-utility itemsets with minimal side effects. *Exp. Theor. Artif. Intell.* 29, 6 (2017), 1225–1245.

[18] Jerry Chun-Wei Lin, Tzung-Pei Hong, and Wen Hsiang Lu. 2010. Efficiently mining high average utility itemsets with a tree structure. In *Lecture Notes in Computer Science*, 131–139.

[19] Jerry Chun-Wei Lin, Tzung-Pei Hong, Jia Wei Wong, Guo Cheng Lan, and Wen Yang Lin. 2014. A GA-based approach to hide sensitive high utility Itemsets. *Sci. World J.* 2014, 1 (2014).

[20] Jerry Chun-Wei Lin, Ting Li, Philippe Fournier-Viger, Tzung-Pei Hong, Justin Zhan, and Miroslav Voznak. 2016. An efficient algorithm to mine high average-utility itemsets. *Adv. Eng. Informatics* 30, 2 (2016), 233–243.

[21] Jerry Chun-Wei Lin, Shifeng Ren, Philippe Fournier-Viger, and Tzung-Pei Hong. 2017. EHAUPM: Efficient High Average-Utility Pattern Mining with Tighter Upper Bounds. *IEEE Access* 5, 8 (2017), 12927–12940.

[22] Jerry Chun-Wei Lin, Shifeng Ren, Philippe Fournier-Viger, Tzung-Pei Hong, Ja-Hwung Su, and Bay Vo. 2017. A fast algorithm for mining high average-utility itemsets. *Appl. Intell.* 47, 2 (2017), 331–346.

[23] Jerry Chun-Wei Lin, Shifeng Ren, Philippe Fournier-viger, Ja-hwung Su, and Bay Vo. 2017. More Efficient Algorithm to Mine High Average-Utility Patterns. In *In: Pan JS., Tsai PW., Huang HC. (eds) Advances in Intelligent Information Hiding and Multimedia Signal Processing. Smart Innovation, Systems and Technologies*, 101–110.

[24] Jerry Chun-Wei Lin, Binbin Zhang, Kuo Tung Yang, and Tzung-Pei Hong. 2014. Efficiently hiding sensitive itemsets with transaction deletion based on genetic algorithms. *Sci. World J.* 2014, (2014), 23–26.

[25] Jerry Chun -Wei Lin, Qiankun Liu, Philippe Fournier-Viger, Tzung-Pei Hong, Miroslav Voznak, and Justin Zhan. 2016. A sanitization approach for hiding sensitive itemsets based on particle swarm optimization. *Eng. Appl. Artif. Intell.* 53, (2016), 1–18.

[26] Jerry Chun -Wei Lin, Tsu Yang Wu, Philippe Fournier-Viger, Guo Lin, Justin Zhan, and Miroslav Voznak. 2016. Fast algorithms for hiding sensitive high-utility itemsets in privacy-preserving utility mining. *Eng. Appl. Artif. Intell.* 55, (2016), 269–284.

[27] Mengchi Liu and Junfeng Qu. 2012. Mining high utility itemsets without candidate generation. In *Proceedings of ACM International Conference on Information and Knowledge Management*, 55–64.

[28] Xuan Liu, Genlang Chen, Shiting Wen, and Guanghui Song. 2020. An Improved Sanitization Algorithm in Privacy-Preserving Utility Mining. *Math. Probl. Eng.* 2020, 1 (2020).

[29] Xuan Liu, Shiting Wen, and Wanli Zuo. 2020. Effective sanitization approaches to protect sensitive knowledge in high-utility itemset mining. *Appl. Intell.* 50, 1 (2020), 169–191.

[30] Grigorios Loukides and Aris Gkoulalas-Divanis. 2012. Utility-preserving transaction data anonymization with low information loss. *Expert Syst. Appl.* 39, 10 (2012), 9764–9777.

[31] Heungmo Ryang and Unil Yun. 2016. High utility pattern mining over data streams with sliding window technique. *Expert Syst. Appl.* 57, (2016), 214–231.

[32] Yi Saygm, Vassilios S Verykios, and Chris Clifton. 2001. Using Unknowns to Prevent Discovery of Association Rules Privacy Preserving Association Rules. *SIGMOD Rec.* 30, 4 (2001), 45–54.

[33] Udit Sharma, Durga Toshniwal, and Shivani Sharma. 2020. A sanitization approach for big data with improved data utility. *Appl. Intell.* 50, 7 (2020), 2025–2039.

[34] Xingzhi Sun and Philip S. Yu. 2007. Hiding Sensitive Frequent Itemsets by a Border-Based Approach. *Comput. Sci. Eng.* 1, 1 (2007), 74–94.

[35] Akbar Telikani, Amir H. Gandomi, Asadollah Shahbahrami, and Mohammad Naderi Dehkordi. 2020. Privacy-preserving in association rule mining using an improved discrete binary artificial bee colony. *Expert Syst. Appl.* 144, (2020), 113097.

[36] Akbar Telikani and Asadollah Shahbahrami. 2018. Data sanitization in association rule mining: An analytical review. *Expert Syst. Appl.* 96, (2018), 406–426.

[37] Tin Truong, Hai Duong, Bac Le, and Philippe Fournier-Viger. 2018. Efficient Vertical Mining of High Average-Utility Itemsets Based on Novel Upper-Bounds. *IEEE Trans. Knowl. Data Eng.* 31, 2 (2018), 301–314.

[38] Tin Truong, Hai Duong, Bac Le, Philippe Fournier-Viger, and Unil Yun. 2019. Efficient high average-utility itemset mining using novel vertical weak upper-bounds. *Knowledge-Based Syst.* 183, 1 (2019), 104847.

[39] Vincent S. Tseng, Bai En Shie, Cheng Wei Wu, and Philip S. Yu. 2013. Efficient algorithms for mining high utility itemsets from transactional databases. *IEEE Trans. Knowl. Data Eng.* 25, 8 (2013), 1772–1786.

[40] Vassilios S. Verykios, Ahmed K. Elmagarmid, Elisa Bertino, Yucel Saygin, and Elena Dasseni. 2004. Association Rule Hiding. *IEEE Trans. Knowl. Data Eng.* 16, 4 (2004), 434–447.

[41] Vassilios S Verykios, Elisa Bertino, Igor Nai Fovino, Loredana Parasiliti Provenza, Yucel Saygin, Yannis Theodoridis, and Universita Milano. 2004. State-of-the-art in Privacy Preserving Data Mining - Classification of Privacy Pre. *ACM SIGMOD Rec.* 33, 1 (2004), 50–57.

[42] Jimmy Ming Tai Wu, Jerry Chun-Wei Lin, Matin Pirouz, and Philippe Fournier-Viger. 2018. TUB-HAUPM: Tighter Upper Bound for Mining High Average-Utility Patterns. *IEEE Access* 6, 1 (2018), 18655–18669.

[43] Jieh Shan Yeh and Po Chiang Hsu. 2010. HHUIF and MSICF: Novel algorithms for privacy preserving utility mining. *Expert Syst. Appl.* 37, 7 (2010), 4779–4786.

[44] Unil Yun and Donggyu Kim. 2017. Mining of high average-utility itemsets using novel list structure and pruning strategy. *Futur. Gener. Comput. Syst.* 68, 1 (2017), 346–360.

[45] Unil Yun, Donggyu Kim, Heungmo Ryang, Gangin Lee, and Kyung Min Lee. 2016. Mining recent high average utility patterns based on sliding window from stream data. *J. Intell. Fuzzy Syst.* 30, 6 (2016), 3605–3617.

[46] Unil Yun and Jiwon Kim. 2015. A fast perturbation algorithm using tree structure for privacy preserving utility mining. *Expert Syst. Appl.* 42, 3 (2015), 1149–1165.

[47] Morteza Zihayat, Heidar Davoudi, and Aijun An. 2017. Mining significant high utility gene regulation sequential patterns. *BMC Syst. Biol.* 11, 6 (2017), 1–14.