A fast algorithm for privacy-preserving utility mining

Duc Nguyen[1,2], Bac Le[1,2]

[1]Faculty of Information Technology, University of Science, Ho Chi Minh City, Viet Nam
[2]Vietnam National University, Ho Chi Minh City, Vietnam

Correspondence: Duc Nguyen, nnduc@fit.hcmus.edu.vn
Communication: received 15 December 2021, revised 22 January 2022, accepted 01 March 222
DOI: 10.32913/mic-ict-research.v2022.n1.1026

***Abstract*: Abstract: Utility mining (UM) is an efficient technique for data mining which aim to discover critical patterns from various types of database. However, mining data can reveal sensitive information of individuals. Privacy preserving utility mining (PPUM) emerges as an important research topic in recent years. In the past, integer programming approach was developed to hide sensitive knowledge in a database. This approach required a significant amount of time for preprocessing and formulating a constraint satisfaction problem (CSP). To address this problem, we proposed a new algorithm based on a hash data structure which performs more quickly in itemsets filtering and problem modeling. Experiment evaluations are conducted on real world and synthetic datasets.**

Keywords: *Hash, privacy, sanitization, utility mining*

I. Introduction

Data mining is the process of discovering interesting patterns in a massive and complicated data collection. Those patterns can be used to make the world a more prosperous and secure place to live. In truth, data mining is a double-edged sword. It can be used to improve our lives through personal advertising, improved medicine and health care, reduced crime, and many more ways. At the same time, however, it can also be used to discover confidential information, to target us with unwanted advertising, and furthermore, to control our behavior through social networks. The debate between the benefits and privacy considerations of data mining are apparent in the discussions [1].

The above issues make Privacy-Preserving Data Mining (PPDM) a critical research problem. The first work was published by Agrawal et al. [2]. Since then, several methods has been proposed to hide patterns in binary datasets [3, 4]. The most common approach of PPDM is hiding sensitive information by perturbing the original database. However, it will cause several side effects such as missing non-sensitive patterns and introducing redundant information. Sanitizing data and hiding sensitive information is a NP-hard optimization problem [2, 5].

Utility pattern mining is a high practical technique in data mining that can analyze relationships between itemsets in transactional database [6]. It allows revealing itemsets yielding high profits by considering both unit profits of items and their quantities in transaction. This method can also cause privacy concerns since hidden pattern found by mining process may implicitly reveal private information [7]. For this reason, various approaches for privacy preserving utility mining (PPUM) were proposed [8]. In the majority of published studies, PPUM conceals confidential information, perturb the original database by removing or reducing quantities of certain items in transactions. The first strategies was introduced by Yeh and Hsu with two algorithms named HHUIF and MSICF [7]. Each algorithm find target transaction and item in each iteration by its own way then modify item's quantities to reduce the utility of sensitive high-utility itemset (SHUI). Another two algorithms was presented by Lin et al. [9, 10]. They hide the itemsets by transaction insertion and transaction deletion. Lin also proposed three evaluation measure for PPUM [11]. Although all sensitive itemsets can be hidden by those algorithms, obtained results have less generality and high rates of side effects. To solve this problem, an algorithm named PPUM-ILP was proposed by Li et al. [12]. They model hiding process as a CSP and solve it with integer linear programming (ILP) technique to change itemset states in a precise and general way. However, their publication does not specify the CSP formulation mechanism which can cause serious performance problem.

In this paper, we introduced a new algorithm which based on integer programming and hash data structures with main contributions as follows.

1) Introducing a hash structures called IT which can be used to quickly filter itemsets and formulating

constraints.
2) Proposing an efficient strategy for modeling sanitization process.
3) Providing extensive experimental results with more complicated utility generation to illustrate the disadvantages of ILP based approach.

II. Related works

1. High utility itemset mining

High-utility itemset mining (HUIM) was first represented by Chan et al. [13]. HUIM methods consider items with different profits and their quantities (or frequencies) in transactions not only present with binary values. They aim at discovering all itemsets that yield utilities larger than a user-defined threshold.

In recent years, HUIM has become a popular research topic. It was first represented by Chan et al. [13]. Later on, Yao and et al. published two algorithms with a formal model for HUIM [6]. For an effective pruning process, Liu et al. introduced the Transaction Weighted Utilization (TWU) [14] model. This model speed up the discovering process by finding High-TWU itemset (HTWUI). All above methods follow the level-wise approach, they perform unnecessary candidate generations and database scans. To overcome this issue, Lin et al. [15] designed a tree structured named HUP-tree. First, the algorithm find all 1-HTWUIs (HTWUIs contain only one item). After that, the HUP-tree is constructed for finding all HUIs. Another approach with a novel algorithm called HUI-miner was presented by Liu and Qu [16]. They used the utility-list structure for storing database information and directly mining HUIs without database scan either candidate generation. For real life problems, Lin et al. [17] introduced a new approach named potential high-utility itemsets mining (PHUIM) to discover itemsets with high-utility as well as itemsets with high existence probabilities. Besides, Lin proposed algorithms for mining HUIs with negative unit profits [18] and multiple minimum utility thresholds [19]. Stream data [20], closed HUIs [21, 22], etc. were also received attention.

2. Privacy-preserving utility mining

With the development of high-utility pattern mining, concerns about privacy especially in commercial environments have become increasingly important. The requirement of protecting sensitive information is becoming more and more urgent. In order to mitigate the privacy threats in utility pattern mining, privacy-preserving utility mining was proposed and got attention of researchers all over the world.

The first work in PPUM was introduced by Yeh et al. [7] with two algorithms, namely, HHUIF and MSICF. The main approach of these algorithms is using heuristic for finding and reducing the quantities of items in specific transactions, thereby achieving the goal of the problem. However, over scanning database in HHUIF and MSICF leads to slow executions. To alleviate this problem, Yun and Kim [23] constructed the FPUTT tree structure and used two structures to speed up the hiding process. FPUTT only need three database scans for the hiding process. Although faster than previous algorithms, FPUTT tree is not compact. During sanitization process, many conditional trees are generated, which make highly computational and memory costs.

Two other heuristic algorithms were proposed by Lin et al. [11] named MSU-MAU and MSU-MIU [11]. They designed the Maximum Sensitive Utility concept to delete items or reduce their quantities. Because the differences with traditional frequent itemset mining, Lin also proposed three evaluations for utility mining. In the other hand, meta-heuristic method also got adapted to solve PPUM problems. Two algorithms based on genetic algorithm, namely, PPUM+insert and PPUMGAT were designed by Li et al. [9, 10]. PPUMGA+insert insert artificial transaction to original database while PPUMGAT remove available transaction. However, insertion/deletion change the number of transactions and may introduce ghost itemsets.

Minimizing the negative effects of the data perturbation is still a problem that has to be solved. To address this problem, an improved algorithm of MSICF named IMSICF was proposed by Liu et al. [24]. The conflict counts of each sensitive itemset are computed dynamically during the hiding process. Although the computational cost is high, in worst cases, we still lose nearly eighty-percent non-sensitive information. For more accurate and general results, instead of using heuristic Li et al. [12] formulate the sanitization process into a CSP problem and use the CSP solution to hide sensitive data.

III. Preliminaries

In this section, some preliminaries and PPUM problem is described briefly. A quantitative transactional database D contains N transactions: $D = \{T_1, T_2, ..., T_N\}$. A transaction $T_n \in D (1 \leq n \leq N)$ includes one or more different items and their quantities in transaction. Let $I = \{i_1, i_2, \ldots, i_M\}$ denoted the M unique items of D, a transaction will be a subset of I ($T_n \subset I$). Each transaction is identified with a id: tid. Table I represents a quantitative database, values in table I are external utilities of items in I.

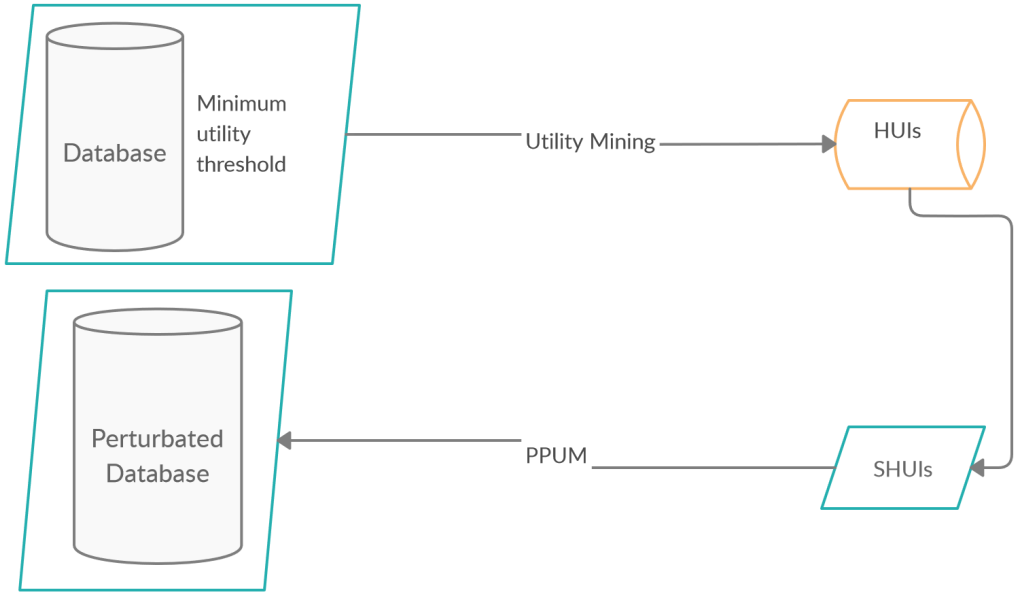

Figure 1. The general PPUM process.

TABLE I
A QUANTITATIVE TRANSACTIONAL DATABASE

tid	A	B	C	D	E
0	3	7	0	4	5
1	3	6	0	0	0
2	0	0	4	6	0
3	0	0	6	4	3
4	7	1	0	6	0
5	0	0	0	3	9
6	7	3	0	0	0
7	5	0	2	3	0
8	8	0	6	0	0
9	6	0	8	0	6

TABLE II
EXTERNAL UTILITY VALUES

Item	External Utility
A	9
B	8
C	1
D	6
E	4

Definition 1 (Internal Utility): The internal utility of item i_m in transaction T_n denoted as $\text{In}(i_m, T_n)$ is the quantity of i_m in transaction T_n.

For example, in Table I the internal utility of item A in transaction T_0 is 3: $\text{In}(A, T_0) = 3$.

Definition 2 (External Utility): The external utility of item i_m denoted as $\text{Ex}(i_m)$ represents the significant or profit of item i_m.

For example, in Table II the external utility of item B is 8: $\text{Ex}(B) = 8$.

Definition 3 (Utility of item i_m in transaction T_n): The utility of item i_m in transaction T_n denoted as $u(i_m, T_n)$ is computed by: $u(i_m, T_n) = \text{In}(i_m, T_n) \times \text{Ex}(i_m)$.

For example, the utility of item B in transaction T_0 is : $u(B, T_0) = \text{In}(B, T_0) \times \text{Ex}(D) = 7 \times 8 = 56$.

Definition 4 (The utility of itemset X in transaction T_n): The utility of itemset X in transaction T_n ($X \subseteq T_n$) denoted as $u(X, T_n)$ is the sum utility of all items of X in transaction T_n:

$$u(X, T_n) = \sum_{i_m \in X} u(i_m, T_n).$$

Note that, $u(X, T_n) = 0$ if $X \nsubseteq T_n$. Particularly, the utility of itemset $\{AD\}$ in transaction T_0: $u(\{AD\}, T_0) = u(A, T_0) + u(D, T_0) = 3 \times 9 + 4 \times 6 = 51$. Because $\{AD\} \nsubseteq T_1$ its utility in transaction T_1 $u(\{AD\}, T_1) = 0$.

Definition 5 (The utility of transaction): The utility of transaction T_n: $tu(T_n) = \sum_{i \in T_n} u(i, T_n)$.

Definition 6 (The utility of itemset X in database D): The utility of itemset X in D is the sum of utilities of X in all transactions contain X: $u(X) = \sum_{T \in D, X \subseteq T_n} u(X, T_n)$.

For example, the utility of itemset $\{AC\}$ in the database I is: $u(\{AC\}) = u(\{AC\}, T_7) + u(\{AC\}, T_8) + u(\{AC\}, T_9) = (5 \times 9 + 2 \times 1) + (8 \times 9 + 6 \times 1) + (6 \times 9 + 8 \times 1) = 47 + 78 + 62 = 187$.

Definition 7 (The minimum utility threshold): The minimum utility threshold denoted as δ, is a constant defined by the user to determine whether an itemset is valuable or not.

Definition 8 (The high-utility itemset (HUI)): An itemset X is called high-utility if its utility is greater than or equal the minimum utility δ.

Let H be the set of all high-utility itemset in D: $H = \{X \subseteq I : u(X) \geq \delta\}$. **Problem.** Given a database D, a set of HUIs H and its subset S contains sensitive high-utility itemset (SHUIs) defined by the user. PPUM problem is finding an appropriate way to modify D into the sanitized one D' that conceals all itemsets in S and minimizes the negative effects to the non-sensitive patterns in H.

A general process of PPUM is described in Figure 1

1. Side effects

All information that does not affect privacy should be retained to acquire benefits of pattern discovering process. However, hiding sensitive information cause some undesirable impact to the non-sensitive information. Three measures for the negative effects proposed by Lin [11] will be used for results evaluations. Let the original data denoted as D, the perturbed D'. H' is the set of high-utility itemsets (HUIs) in D'. Let N is the set of non-sensitive high-utility itemsets (NHUIs) in D and S is the set of SHUIs in D.

Definition 9 (Hiding Failure): Hiding Failure represents that some sensitive itemsets in D are still be found as high-utility itemsets in D'. Let HF denote the hiding failure, HF can be calculated as follows:

$$HF = \frac{|S \cap H'|}{|S|}$$

Definition 10 (Missing Cost): Missing cost indicates the lost of NSHUIs after the sanitization process. Let MC denoted the missing cost. MC can be calculated as follows:

$$MC = \frac{|N - N \cap H'|}{|N|}$$

Definition 11 (Artificial Cost): Artificial cost represents the redundant itemsets introduced by PPUM algorithms. Let AC denote the artificial cost. AC can be calculated as follows:

$$AC = \frac{|H' - H \cap H'|}{|N|}$$

In addition, in [12] Li et al. proposed a new measure named hiding cost by combining the missing cost and the artificial cost and used hiding cost to evaluate the side effects of PPUM algorithms.

Definition 12 (Hiding cost): Hiding cost denote the ratio of the number of lost NSHUIs and false HUIs in D' to the number of NSHUIs in the original database. HC can be calculated as follows:

$$HC = \frac{|N - H'|}{|N|}$$

The formula show that the hiding cost represents no information about the redundant HUIs (or the false HUIs) in D'. Therefore, we did not use the hiding cost for our evaluation.

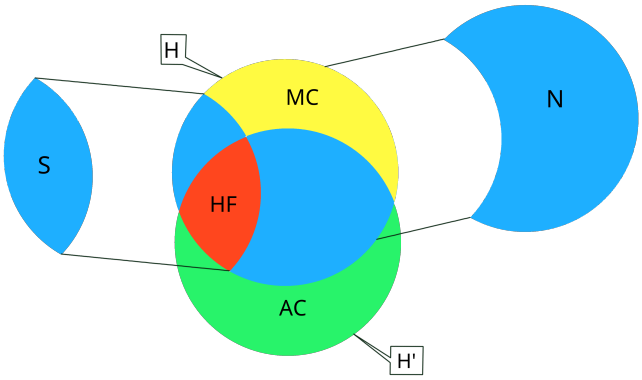

Figure 2. Three side effects of PPUM.

IV. The proposed algorithm FILP

The main idea of PPUM based on integer programming is formulation the hiding sensitive information process into a constraint satisfaction problem. The CSP has two principal objectives, i.e, hiding SHUIs and minimizing negative effects. To do that, we divide the PPUM based ILP approach into four main steps. Firstly, data structures maintain relationships between itemsets and their support transactions will be constructed. Secondly, the unaffected and redundant itemsets get removed. Thirdly, CSP is formulated and solved. Finally, the CSP solution is used to perturb the database. The whole sanitization process is described in Figure 3.

1. It tables construction

The PPUM problem input consists of: the original database D, the set of sensitive itemsets S, the set of HUIs H and the minimum utility threshold δ. For preserving the relations between itemsets and transactions, and speeding up the preprocessing and establishing constraints processes, we designed a table structure called itemset-tidset table (IT).

Definition 13: Given a set of itemsets, the itemset-tidset table (IT) consists of 3 columns. The first column stores itemset by hash set structure. The second also uses hash set to store tids of transactions that respectively support the itemsets in first column. The third column stores the itemset utilities.

The sensitive itemset-tidset table (S-IT) is constructed by scan the original database once. Each $s_i \in I_S$ get hashed and stores in S-IT table with it utility. Transactions support

TABLE III
THE IT TABLE STRUCTURE

(Hashed) High-utility itemset	(Hashed) Tidset	Utils
a_1	t_1	u_1
a_2	t_2	u_2
a_3	t_3	u_3

s_i are found by the scan and the set of their tids gets hashed. The S-IT table is used to establish CSP and find special HUIs. With the utilization of S-IT and a prober CSP construction methods, we can reduce database scanning times in preprocessing and CSP formulation of PPUM based ILP approach and avoid handling separately each SHUI like heuristic, based methods. Non-sensitive itemset-tidset (N-IT) table also is constructed in same database scan. The only difference of N-IT and S-IT is that N-IT store information for NSHUIs instead of SHUIs. The table construction is described in Algorithm 1.

Algorithm 1: Table Construction

```
Input: D, the original database; H, the set of HUIs
  discovered from D; S, the set of SHUIs to be hidden
Output: N-IT, S-IT tables.
begin
    foreach transaction T_n ∈ D do
        foreach itemset X ∈ H do
            if X ⊆ T_n then
                if X ∈ S then
                    Insert X, U(X) into S-IT, stores tid
                      of T_n;
                end
                Insert X, U(X) into N-IT, stores tid of
                  T_n;
            end
        end
    end
    Hash stored tidsets and insert them to IT tables;
end
```

2. Preprocessing

Concealing SHUIs affect the NSHUIs that share same items and transactions. The consequence is losing information or introducing redundant information. As [12] suggested, filtering itemsets that get affected by perturbation make the hiding process more efficient. The proposed algorithm also used the same filtering strategy. The first phase of preprocessing is applying a filter mechanism.

Definition 14 (Filter mechanism [12]): Given the sets of sensitive itemsets S and the set of NHUIs N. Itemset $n_i \in N$ will be affected by hiding process if it shares at least one item and one transaction with any sensitive itemset $s_i \in S$

Let N' be the set of remaining NSHUIs of N after the filtering. Next step we find bound-to-lose itemsets in N'. Because such itemsets have utility lower than its sensitive superset, removing them can help avoid some conflict constraints in the CSP.

Definition 15 (Bound-to-lose itemset [22]): Given itemset X, sensitive itemset Y and the set of transactions T_X, T_Y that respectively support X, Y. X is called bound-to-lose itemset if X is a high-utility itemset and $X \subset Y \wedge T_X = T_Y$.

Let the set of remaining NHUIs is N''. Last step is removing itemsets corresponding to unnecessary constraints. In particular, if two itemset $n_{i_1}, n_{i_2} \in N''$ share same transaction set $T_{n_{i_1}}$ and $n_{i_1} \subset n_{i_2}$, n_{i_2} will be removed of N''. The whole preprocessing is described in Algorithm 2.

Algorithm 2: Preprocessing

```
Input: N-IT, S-IT tables
Output: Modified N-IT, S-IT tables. begin
    foreach itemset n_i in N-IT do
        L ← 0; ;
        foreach itemset s_i in S-IT do
            if n_i ∩ s_i ≠ ∅ and T_{n_i} ∩ T_{s_i} ≠ ∅ then
                if n_i ⊆ s_i and T_{n_i} = T_{s_i} then
                    Delete n_i from N-IT;
                end
            end
            L ← L + 1;
        end
        if L == |S| then
            Delete n_i from N-IT;
        end
    end
    foreach itemset n_{i_2} in N-IT do
        foreach itemset n_{i_1} in N-IT do
            if n_{i_1} ≠ n_{i_2} and n_{i_1} ⊆ n_{i_2} and T_{n_{i_1}} = T_{n_{i_2}}
              then
                Delete n_{i_2} from N-IT;
                break;
            end
        end
    end
end
```

As shown in Algorithm 2, the preprocessing perform various operators to find the relation, i.e, the intersections between itemsets. Let m is the mean length of high-utility itemsets, the time complexity to find the intersection between two itemsets is $O(m^m)$. In worst cases, we need to find the intersections of $\binom{|H|}{2}$ pair of itemsets, which lead to $O\left(\binom{|H|}{2}^{m^m}\right)$. With the utilization of IT table, tidsets and itemsets are represented by hash structures. It reduced the time complexity for intersection to $O(m)$ and the whole preprocessing in the worst case to $O\left(\binom{|H|}{2}^{m}\right)$.

3. Problem formulation

The effective way to hide sensitive itemsets is changing internal utilities of item. Finding the CSP solution

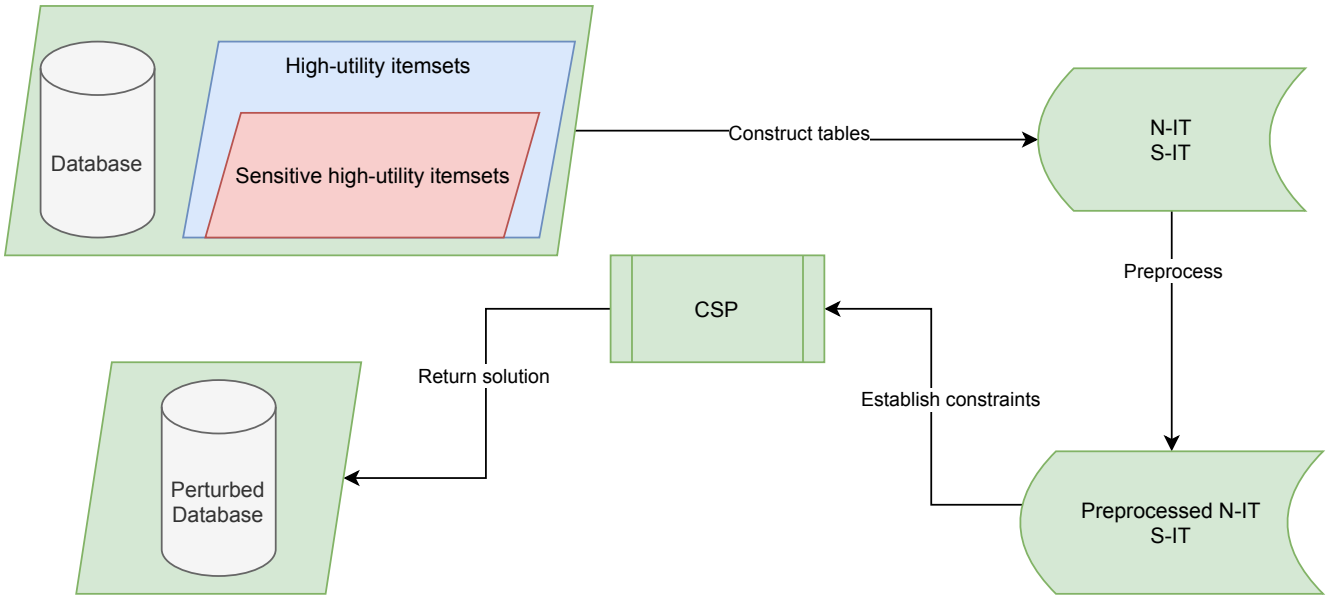

Figure 3. The PPUM based ILP approach.

is finding the optimal values for internal utilities. Those values make the sensitive high-utility itemsets become low-utility itemsets and minimize the negative impact to non-sensitive high-utility itemsets. In [12], authors did not specify the CSP formulation. The problem is a brute-force CSP formulation can lead to serious performance degradation. The proposed algorithm uses a well-designed process and utilizes the proposed data structure for faster formulating.

First, all the internal utilities of SHUIs in S is substituted by integer variables. Specifically, for each SHUI stored in S-IT, we will get the combinations of item and tid. Each combination will be used as index to indicate its utility. After this step, a hash table with variables and their utilities is constructed. Let $V = \{v_{nm} | n \in [1, N] \cap \mathbb{N}, m \in [1, M] \cap \mathbb{N}\}$, v_{nm} represents the internal utility of itemset m in transaction n. The minimum value of a variable is suggested to be 1 [12]. A SHUI X become LUI, if:

$$\sum_{T_n \in D, X \subseteq T_n} \sum_{i_m \in X} v_{nm} Ex(i_m) < \delta \tag{1}$$

In the same above step, the constraints that correspond to formula 1 also are established. Note that, for better communication with mathematical solvers, constraints should be represented by coefficient matrix. For example, let $\boldsymbol{M_S}$ be the coefficient matrix of constraints corresponding to formula 1, $\boldsymbol{v}$ be the vector of variables. To add these constraints to a solver, we simply multiply the coefficient matrix $\boldsymbol{M_S}$ with the vector of variables $\boldsymbol{v}$: $\boldsymbol{M_S v} \leq \boldsymbol{\delta}$.

Let U is the set of internal utilities of item in R:

$$U = \{u_{n'm'} | n' \in [1, N] \cap \mathbb{N}, m' \in [1, M] \cap \mathbb{N}\}$$

$u_{n'm'}$ represents two states of internal utility, i.e:

$$u_{n'm'} = \begin{cases} v_{n'm'}, & \text{if } v_{n'm'} \in V, \\ \text{In}(i_{m'}, T_{n'}), & \text{otherwise} \end{cases}$$

An itemset $Y \in R$ utility remains higher than minimum utility threshold if:

$$\sum_{T_{n'} \in D, Y \subseteq T_{n'}} \sum_{i_{m'} \in Y} u_{n'm'} Ex(i_{m'}) \geq \delta \tag{2}$$

Building constraints according to formula 2 without the hash table of variables and the N-IT table has enormous time cost. To establish the constraint of a NHUI in R, we have to find the intersection between it and each SHUI, which time complexity is $O\left(|S|^{m^m}\right)$. Besides, intersections between tidsets of SHUI and NHUI also have to be found. Let t is the average number of transactions that support a HUI, the time complexity for getting tidset intersections is $O\left(|S|^{t^t}\right)$. Moreover, we need to scan the database once to compute the remaining utility of the NHUI after substitution. Average time complexity for this scan (not included data loading time) is $O(m \times t)$. So the complexity for establishing the constraint of a NHUI is $O\left(|S|^{m^m} + |S|^{t^t} + m \times t\right)$. The complexity of establishing constraints corresponding to formula 2 is $O\left(|R|^{|S|^{m^m} + |S|^{t^t} + m \times t}\right)$. With the variable hash table to check whether a combination is substituted or not only has time complexity $O(1)$. Because the utilities of variables and utilities of NHUIs are stored in data structures. We

can compute the remaining utility of a NHUI only by a subtraction. The time complexity of establishing formula 2 constraints is reduced to $O(|R|^{m\times t})$. The time complexity comparison of proposed algorithm and PPUM-ILP is shown in Table IV. Let $\boldsymbol{M_n}$ be the coefficient matrix of constraints corresponding to formula 2, $\boldsymbol{r}$ be the vector of remaining utilities. These constraints can be formulated by: $\boldsymbol{M_n v} \geq \boldsymbol{\delta} - \boldsymbol{r}$.

For reducing artificial cost, the objective function is minimizing the sum of integer variables. Formally, the CSP is described as follows:

$$min \sum_{v_{nm}\in V} v_{nm} \tag{3}$$

$$s.t. \sum_{i_m\in X}\sum_{T_n\in T_X} v_{nm}Ex(i_m) \leq \delta, \qquad \forall X \in I_S \tag{4}$$

$$\sum_{i_{m'}\in Y}\sum_{T_{n'}\in T_Y} u_{n'm'}Ex(i_{m'}) \geq \delta, \qquad \forall Y \in I_R \tag{5}$$

$$v_{nm} \geq 1, \qquad \forall v_{nm} \in V \tag{6}$$

The CSP does not always arrive at a feasible solution. The proposed algorithm uses the same relaxation method introduced in [12], finding relaxation by removing constraints iteratively.

Algorithm 3: CSP Formulation

```
Input: S-IT, N-IT tables.
Output: CSP model.
begin
    Using the S-IT table create a hash table V to store
      variable positions and utilities; establish inequalities
      according to formula 4.;
    foreach itemset n_i in N-IT do
        Subtract the sum of the variable utilities in n_i
          from n_i utility.;
        Establish inequalities according to formula 5.;
    end
end
```

V. Illustrated example

In this section, the overall algorithm process is described step-by-step via an illustrated example. Returning to the database D in Table I, suppose that the minimum utility threshold δ is set to 124 and $\{AB\}$ is the sensitive itemset. The N-IT (Table V) and S-IT (Table VI) tables are first constructed. To reduce the size of problem, the itemsets in the N-IT table are preprocessed. First, the itemsets that will not be affected by the sanitization process are removed. In this case, itemsets $\{D\}$ and $\{DE\}$ are removed because they have no items in common with itemset $\{AB\}$. Itemsets $\{AC\}$ and $\{AB\}$ are not supported by any of the same transactions; thus, we delete $\{AC\}$ from the N-IT table. Itemset $\{B\}$ is not closed to $\{AB\}$, so it is eliminated to avoid conflict when establishing the constraints. Last, redundant itemsets are removed to obtain a smaller-sized problem. Itemset $\{BD\}$ is not closed to itemset $\{ABD\}$, so $\{ABD\}$ is deleted directly. Table VII shows the preprocessed N-IT table.

After preprocessing, we establish the CSP. Sensitive items of sensitive itemsets in transactions are replaced by integer variables, and constraints corresponding to formula 4 are formulated by scanning the S-IT table (Table V). Next, we scan the N-IT table (Table VII) and establish constraints for formula 5. We can compute the remaining utilities of the itemsets rapidly after substitution by subtracting variable utilities from their utilities. In our example, remaining utility of itemset $\{A\}$ after subtracting its utility from the T_0, T_1, T_4 and T_6 transactions is 171. One-hundred seventy-one is higher than minimum utility threshold (124), so there is no need to establish a constraint for $\{A\}$. In this example, the CSP does not have a feasible solution, so a relaxed solution is used to create sanitized database D' (Table VIII).

In the mining result (Table IX), sensitive itemset $\{AB\}$ is hidden; note that we lose some NSHUIs: $\{B\}$ (bound-to-lose itemset), $\{ABDE\}$ and $\{BD\}$. Furthermore, no LUI becomes a HUI. Overall, $HF = 0$, $MC = 0.3$ and $AC = 0$.

VI. Experimental results

1. Experimental environments

We performed the experiments on an Intel(R) Core(TM) i7-6700 CPU @ 3.40GHz, RAM 32GB. Two real world small datasets and a synthetic large dataset with their characteristics are displayed in Table X. The density of a dataset is measured as the average transaction length divided by the number of distinct items: Density = $\frac{AvgLen}{|I|}$. For each dataset, the external utility of each item is generated in the interval $[1, 1000] \cap \mathbb{N}$ with the log normal distribution, whereas the internal utilities of the items in the transactions are generated to be between 1 and 10 using the uniform distribution. In this paper, we used the state of the art d2HUP [25, 26] algorithm as the mining technique. The performances of the proposed algorithm were evaluated and compared with other algorithms across the datasets by execution time and side effects. The selected algorithms are as follows: PPUM-ILP, HHUIF, MSICF [7], MSU-MAU, MSU-MIU [17]; we did not compare our algorithm with PPUMGAT [9] and PPUMGAT+ [17] because these two algorithms hide information based on insertion/deletion transactions and have the different context. For better comparison all the algorithms were implemented with the high performance language Julia and the fastest solver for ILP, i.e, Gurobi 9.1.2 [27] was used to solve the CSP

TABLE IV
COMPARISON OF TIME COMPLEXITY BETWEEN PROPOSED ALGORITHM AND PPUM-ILP.

Algorithm	FILP		PPUM-ILP	
Phase	Preprocessing	Establishing formula 2	Preprocessing	Establishing formula 2
Time complexity	$O\left(\binom{\lvert H\rvert}{2}^m\right)$	$O\left(\lvert R\rvert^{m\times t}\right)$	$O\left(\binom{\lvert H\rvert}{2}^{m^m}\right)$	$O\left(\lvert R\rvert^{\lvert S\rvert^{m^m}+\lvert S\rvert^t+m\times t}\right)$

TABLE V
S-IT TABLE.

SHUI	Tidset	Utility
AB	$T_0T_1T_4T_6$	316

TABLE VI
N-IT TABLE.

NSHUI	Tidset	Utility
A	$T_0T_1T_4T_6T_7T_8T_9$	351
B	$T_0T_1T_4T_6$	136
D	$T_0T_2T_3T_4T_5T_7$	156
AC	$T_7T_8T_9$	187
AE	T_0T_9	125
AD	$T_0T_4T_7$	213
BD	T_0T_4	124
DE	$T_0T_3T_5$	134
ABD	T_0T_4	214
$ABDE$	T_0	127

TABLE VII
PREPROCESSED N-IT TABLE.

NSHUI	Tidset	Utility
A	$T_0T_1T_4T_6T_7T_8T_9$	351
AD	$T_0T_4T_7$	213
AE	T_0T_9	125
BD	T_0T_4	124
$ABDE$	T_0	127

TABLE VIII
THE SANITIZED DATABASE D'.

tid	A	B	C	D	E
0	3	5	0	4	5
1	1	1	0	0	0
2	0	0	4	6	0
3	0	0	6	4	3
4	1	1	0	6	0
5	0	0	0	3	9
6	1	1	0	0	0
7	5	0	2	3	0
8	8	0	6	0	0
9	6	0	8	0	6

(for our algorithm and PPUM-ILP). The time limit for all algorithm is set to 1200 seconds. The sensitive itemsets are randomly sampled from the set of high-utility itemsets H based on a sensitive information percentage. Execution time of an algorithm is the average of five running times. The experiments to analyze execution time were conducted under two contexts: increasing the minimum utility threshold and increasing the sensitive information percentage (SIP). Because the utility generation process is

TABLE IX
HIT TABLE FOR D'.

HUI	Tidset	Utility
A	$T_0T_1T_4T_6T_7T_8T_9$	225
D	$T_0T_2T_3T_4T_5T_7$	156
AC	$T_7T_8T_9$	187
AD	$T_0T_4T_7$	159
AE	T_0T_9	125
DE	$T_0T_3T_5$	134
ABD	T_0T_4	134

TABLE X
THE CHARACTERISTICS OF THE DATASETS.

Dataset	N	M	Density(%)
Mushrooms	8124	119	19.3
Foodmart	4141	1559	0.25
T20I6D100K	99922	929	2.22

more complicated than [12], we can see the weakness of the PPUM based ILP approach. In mushrooms datasets, the minimum utility threshold and the sensitive information percentage were set respectively to 10% and 9%. All the ILP based algorithms can not finish under the time limit (included the proposed algorithm). Because mushrooms is a dense dataset, their itemsets share many similar items. That causes many conflict constraints in CSP and moreover the CSP is harder to check whether it is feasible or not. In order to find a solution these algorithms need to iteratively remove a constraint and presolve the CSP, which makes their runtimes exceed the time limit.

2. Foodmart

The PPUM-ILP algorithm can only execute successfully in foodmart dataset. In the large and sparse T20I6D100K dataset, it failed to finish under the time limit because the computational cost for preprocessing and establishing constraints. The comparison with PPUM-ILP will be conducted separately in foodmart. Foodmart is a small and sparse datasets so that all PPUM algorithms can finish successfully the hiding. Although the length of high-utility itemsets in foodmart is small, we still gain performance advantages with the proposed IT table structure. As we can see in Figure 4 and Figure 5, all the time the proposed algorithm has lower execution time than PPUM-ILP. It also can be seen that, when the minimum utility threshold

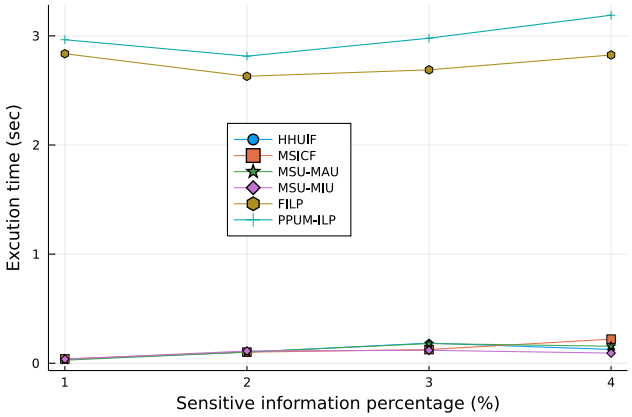

Figure 4. Execution times of algorithms in foodmart dataset under various SIP.

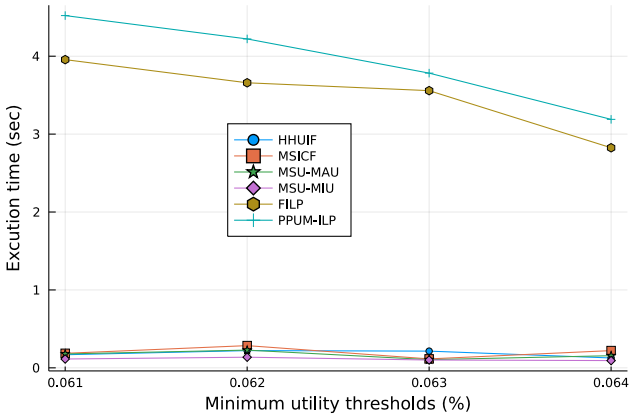

Figure 5. Execution times of algorithms in foodmart dataset under different minimum utility thresholds.

increases, the execution time decreases since the size of S become smaller. Table XI show the ability of algorithms to minimize negative effects in foodmart dataset. Since all the algorithms are able to hide all sensitive itemsets, we only compare the other two side effects. From table XI, we observed that even with the more general problem formulation the obtained solutions are not better. The reason behind this is the lack of accurate when finding a relaxation by iteratively removing constraints. As described in [12], the removed constraint of each iteration corresponds to the itemsets which has the longest length and lowest utility. But there is no mathematical proof for that constraint should be the most conflict in the CSP.

3. T20I6D100K

The execution times of algorithms in T20I6D100K is shown in Figure 6 and 7. Observing the results we see that in most cases, algorithm execution time increases as the sensitive information percentage (SIP) increases. This result

TABLE XI
SIDE EFFECTS OF ALGORITHMS IN FOODMART.

SIP	2		4	
Side effects	MC	AC	MC	AC
HHUIF	79.7%	0.0%	82.4%	0.0%
MSICF	74.5%	0.0%	81.8%	0.0%
MSU-MAU	79.7%	0.0%	82.3%	0.0%
MSU-MIU	76.3%	0.0%	80.7%	0.0%
PPUM-ILP	81.7%	2.6%	84.4%	7.1%
FILP	81.7%	2.6%	84.4%	2.8%

TABLE XII
SIDE EFFECTS OF ALGORITHMS IN T20I6D100K.

SIP	2		8	
Side effects	MC	AC	MC	AC
HHUIF	8%	0%	26.1%	0%
MSICF	8%	0%	24.2%	0%
MSU-MAU	8%	0%	26.1%	0%
MSU-MIU	4.3%	0%	20.3%	0%
FILP	0%	0.6%	8.5%	168%

is reasonable since as we increase the sensitive information percentage while maintaining a fixed minimum utility threshold, the number of sensitive high-utility itemsets increases. As a consequence, the CSP becomes harder to solve for PPUM based ILP approaches and the hiding iterations of heuristic algorithms increases. However, T20I6D100K is a sparse dataset, time for solving CSP problem is not changed much. Additionally, in worst cases, to hide a sensitive itemset, heuristic algorithms need to perform database scan for every item belong to the itemset. Too many database scans will slow down the hiding process and increase the running time. In some cases, when increases the minimum utility thresholds, the runtimes of algorithms increase. This is because the higher the minimum utility thresholds, the more frequent the itemsets appear in database and the more database scans for heuristic algorithms.

The side effects of algorithms in T20I6D100K are shown in Table XII. The FILP algorithm achieves better results for retaining non-sensitive information but in some cases it can introduce too much redundant information. Beyond that the proposed algorithm always runs faster than PPUM-ILP.

VII. CONCLUSION AND DISCUSSION

Data mining can reveal implicit sensitive pattern and private information in a database. Privacy concerns about data mining are becoming more and more serious. Protecting privacy in data mining, especially for utility mining is a critical problem. However, hiding sensitive patterns while retaining non-sensitive information is a NP-hard problem. In this paper, we introduced a faster PPUM method based on ILP approach with the utilization of hash structures. The conducted experiments have shown the efficiency of proposed algorithm. Besides, we still have some problems

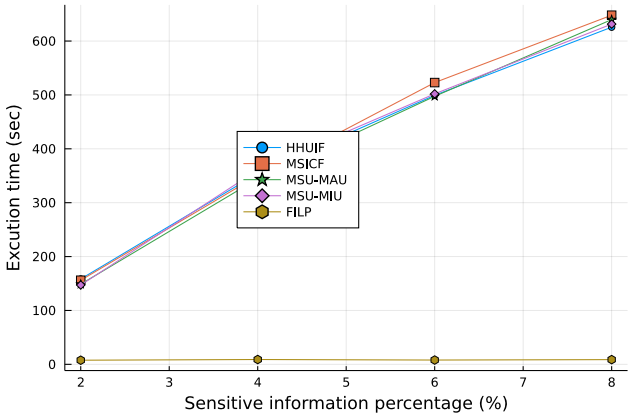

Figure 6. Execution times of algorithms in T20I6D100K dataset under various SIP.

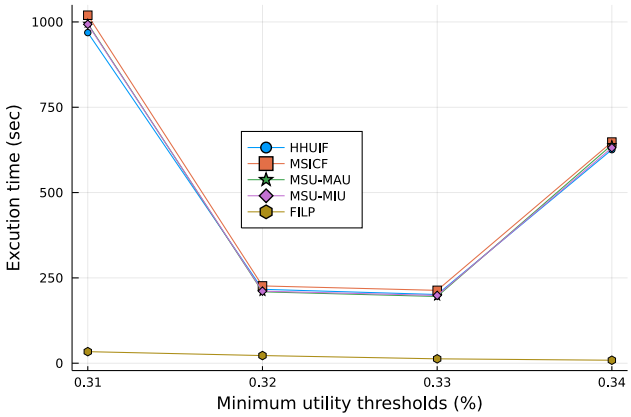

Figure 7. Execution times of algorithms in T20I6D100K dataset under different minimum utility thresholds.

with the ILP based approach. The CSP of hiding process does not have specific constraints to limit artificial cost. The time for solving the CSP is too high, so we need a better way to find a feasible solution. In addition, the preprocessing can be run asynchronously using parallel programming methods.

References

[1] A. Mantelero and G. Vaciago, "The "Dark Side" of Big Data: Private and Public Interaction in Social Surveillance, How data collections by private entities affect governmental social control and how the EU reform on data protection responds," *Computer Law Review International*, vol. 14, pp. 161–169, 12 2013.

[2] R. Agrawal and S. Ramakrishnan, "Privacy-preserving data mining," in *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '00. New York, NY, USA: Association for Computing Machinery, 2000, pp. 439–450. [Online]. Available: https://doi.org/10.1145/342009.335438

[3] A. V. Evfimievski, S. Ramakrishnan, R. Agrawal, and J. Gehrke, "Privacy preserving mining of association rules," in *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, July 23-26, 2002, Edmonton, Alberta, Canada*. ACM, 2002, pp. 217–228. [Online]. Available: https://doi.org/10.1145/775047.775080

[4] J. Vaidya and C. Clifton, "Privacy preserving association rule mining in vertically partitioned data," in *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '02. New York, NY, USA: Association for Computing Machinery, 2002, pp. 639–644. [Online]. Available: https://doi.org/10.1145/775047.775142

[5] V. S. Verykios, E. Bertino, I. N. Fovino, L. P. Provenza, Y. Saygin, and Y. Theodoridis, "State-of-the-art in privacy preserving data mining," *SIGMOD Record*, vol. 33, no. 1, pp. 50–57, March 2004. [Online]. Available: https://doi.org/10.1145/974121.974131

[6] H. Yao, H. J. Hamilton, and C. J. Butz, "A foundational approach to mining itemset utilities from databases," in *Proceedings of the 2004 SIAM International Conference on Data Mining*, 2004, pp. 482–486.

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

[8] U. Yun and D. Kim, "Analysis of privacy preserving approaches in high utility pattern mining," in *Advances in Computer Science and Ubiquitous Computing*, J. J. J. H. Park, Y. Pan, G. Yi, and V. Loia, Eds., Singapore, 2017, pp. 883–887.

[9] C.-W. Lin, T.-P. Hong, J.-W. Wong, G.-C. Lan, and W.-Y. Lin, "A GA-based approach to hide sensitive high utility itemsets," *The Scientific World Journal*, vol. 2014, pp. 2356–6140, March 2014.

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

[11] J. C.-W. Lin, T.-Y. Wu, P. Fournier-Viger, G. Lin, J. Zhan, and M. Voznak, "Fast algorithms for hiding sensitive high-utility itemsets in privacy-preserving utility mining," *Engineering Applications of Artificial Intelligence*, vol. 55, pp. 269–284, 2016.

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

[13] R. Chan, Q. Yang, and Y.-D. Shen, "Mining high utility itemsets," in *Third IEEE International Conference on Data Mining*, 2003, pp. 19–26.

[14] Y. Liu, W.-K. Liao, and A. Choudhary, "A two-phase algorithm for fast discovery of high utility itemsets," in *Advances in Knowledge Discovery and Data Mining*, T. B. Ho, D. Cheung, and H. Liu, Eds., Berlin, Heidelberg, 2005, pp. 689–695.

[15] C.-W. Lin, T.-P. Hong, and W.-H. Lu, "An effective tree structure for mining high utility itemsets," *Expert Systems with Applications*, vol. 38, no. 6, pp. 7419–7424, 2011.

[16] M. Liu and J. Qu, "Mining high utility itemsets without candidate generation," in *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, ser. CIKM '12, New York, NY, USA, 2012, pp. 55–64.

[17] J. C.-W. Lin, W. Gan, P. Fournier-Viger, T.-P. Hong, and V. S. Tseng, "Efficient algorithms for mining high-utility itemsets in uncertain databases," *Knowledge-Based Systems*, vol. 96, pp. 171–187, 2016.

[18] J. C.-W. Lin, P. Fournier-Viger, and W. Gan, "FHN: An

efficient algorithm for mining high-utility itemsets with negative unit profits," *Knowledge-Based Systems*, vol. 111, pp. 283–298, 2016.
[19] J. C.-W. Lin, W. Gan, P. Fournier-Viger, and T.-P. Hong, "Mining high-utility itemsets with multiple minimum utility thresholds," in *Proceedings of the Eighth International C* Conference on Computer Science & Software Engineering*, ser. C3S2E '15, New York, NY, USA, 2015, pp. 9–17.
[20] H.-F. Li, H.-Y. Huang, and S.-Y. Lee, "Fast and memory efficient mining of high-utility itemsets from data streams: with and without negative item profits," *Knowledge and Information Systems*, vol. 28, no. 3, pp. 495–522, 2011.
[21] C. W. Wu, P. Fournier-Viger, P. S. Yu, and V. S. Tseng, "Efficient mining of a concise and lossless representation of high utility itemsets," in *2011 IEEE 11th International Conference on Data Mining*, 2011, pp. 824–833.
[22] C. Wu, P. Fournier-Viger, J. Gu, and V. S. Tseng, "Mining closed+ high utility itemsets without candidate generation," in *2015 Conference on Technologies and Applications of Artificial Intelligence (TAAI)*, 2015, pp. 187–194.
[23] U. Yun and J. Kim, "A fast perturbation algorithm using tree structure for privacy preserving utility mining," *Expert Systems with Applications*, vol. 42, no. 3, pp. 1149–1165, 2015.
[24] X. Liu, G. Chen, S. Wen, and G. Song, "An improved sanitization algorithm in privacy-preserving utility mining," *Mathematical Problems in Engineering*, vol. 2020, pp. 1–14, April 2020.
[25] J. Liu, K. Wang, and B. C. Fung, "Direct discovery of high utility itemsets without candidate generation," *2012 IEEE 12th International Conference on Data Mining*, Dec. 2012.
[26] C. Zhang, G. Almpanidis, W. Wang, and C. Liu, "An empirical evaluation of high utility itemset mining algorithms," *Expert Systems with Applications*, vol. 101, pp. 91–115, Jul. 2018.
[27] L. Gurobi Optimization, "Gurobi optimizer reference manual," 2020. [Online]. Available: http://www.gurobi.com

Duc Nguyen Ngoc Duc N. Nguyen received B.S in 2018 and M.S in 2021 from University of Science-VNUHCM. He is currently working at Department of Computer Science, University of Science - VNUHCM. His research interests are machine learning, pattern recognition, data mining and privacy-preserving in data mining.
Email: nnduc@fit.hcmus.edu.vn

Bac Le Hoai is currently a Professor and Head of Department of Computer Science, Faculty of Information Technology, University of Science, Vietnam National University, Ho Chi Minh City, Vietnam. His main research includes artificial intelligence, soft computing, machine learning and data mining.
Email: lhbac@fit.hcmus.edu.vn