Applied Intelligence
https://doi.org/10.1007/s10489-024-05826-y

Novel stochastic algorithms for privacy-preserving utility mining

Duc Nguyen[1,2] · **Bac Le**[1,2]

Accepted: 28 August 2024

Abstract
High-utility itemset mining (HUIM) is a technique for extracting valuable insights from data. When dealing with sensitive information, HUIM can raise privacy concerns. As a result, privacy-preserving utility mining (PPUM) has become an important research direction. PPUM involves transforming quantitative transactional databases into sanitized versions that protect sensitive data while retaining useful patterns. Researchers have previously employed stochastic optimization methods to conceal sensitive patterns in databases through the addition or deletion of transactions. However, these approaches alter the database structure. To address this issue, this paper introduces a novel approach for hiding data with stochastic optimization without changing the database structure. We design a flexible objective function to let users restrict the negative effects of PPUM according to their specific requirements. We also develop a general strategy for establishing constraint matrices. In addition, we present a stochastic algorithm that applies the ant lion optimizer along with a hybrid algorithm, which combines both exact and stochastic optimization methods, to resolve the hiding problem. The results of extensive experiments are presented, demonstrating the efficiency and flexibility of the proposed algorithms.

Keywords Ant lion · Optimizer · Privacy · Utility mining

1 Introduction

Data mining is the extraction of interesting patterns from databases, primarily targeting valuable insights from complex datasets [1]. Utility mining, a popular technique, focuses on identifying high-utility itemsets (HUIs) based on unit profits and quantities in transactions. This approach has gained significant attention because of its broad applicability. However, discovering patterns in business and health data raises privacy concerns [2]. Hidden patterns can expose personal identities, private relationships, or confidential trade secrets. An approach to addressing these concerns is to process data with decentralized methods such as federated learning [3]. However, there are several challenges; for example, the volume of data in distributed devices needs to be sufficient and the communication protocols need to be secure [4]. As a traditional approach, privacy-preserving utility mining (PPUM) techniques have been developed to protect sensitive information before sharing and mining data [5].

In most published studies, sensitive high-utility itemsets (SHUIs) are hidden using approaches that remove or reduce their utilities in the database. The earliest was developed by Yeh et al. [6], which involved two algorithms, namely, hiding high utility item first (HHUIF) and maximum sensitive itemsets conflict first (MSICF). These algorithms identify victim transactions and items, then iteratively reduce victim item quantities to diminish the utilities of SHUIs. Two additional heuristic algorithms were proposed by Lin et al. [7], namely, maximum sensitive utility-maximum item utility (MSU-MAU) and maximum sensitive utility-minimum item utility (MSU-MIU). These algorithms introduced the maximum sensitive utility concept as an effective heuristic. Recently, sorting techniques and new concepts [8, 9] have also been presented. A second approach was proposed by Lin et al. [10]. In this method, the original database is perturbed by inserting artificial transactions or removing current transactions.

These techniques use heuristic concepts to hide sensitive itemsets. However, they come with low generality and high

✉ Bac Le
lhbac@fit.hcmus.edu.vn

Duc Nguyen
nnduc@fit.hcmus.edu.vn

[1] Faculty of Information Technology, University of Science, Ho Chi Minh City, Vietnam

[2] Vietnam National University, Ho Chi Minh City, Vietnam

Published online: 27 September 2024

rates of negative effects. For a more formal strategy, Li et al. [11] presented the PPUM-ILP algorithm. They developed the HI-table to store the HUIs that will be affected during the hiding process. These itemsets' states are efficiently changed using integer programming techniques. Since removing constraints iteratively to form and solve a feasible model leads to slow execution, Duc et al. [12] designed a mathematical optimization relaxation. Furthermore, they employed a parallel preprocessing method to address the hiding problem.

Optimization is the process of finding a good solution from the feasible set of an optimization problem. The challenge is that most optimization problems are NP-hard. Instead of finding exact solutions, researchers use stochastic methods to create an iterative search process that can escape from the local minima [13]. In recent years, new stochastic optimization algorithms have emerged and been applied to various domains [14, 15], with some achieving remarkable performance. The stochastic behaviors in many of these algorithms are inspired by natural phenomena, including hunting behaviors (ant lions [16], chameleons [17], hyenas [18]), black holes [19], and the socio-political process [20], among others [21–23]. Stochastic optimization techniques are also widely applied for preserving privacy in data mining [24].

In this paper, two PPUM algorithms, called PPUM-ALO and PPUM-IALO, are proposed to address the sanitizing problem with a more flexible approach that allows the use of both exact and stochastic optimization methods to resolve the hiding problem. Our primary contributions are summarized as follows:

- Proposing a general strategy to formulate the optimization problem for hiding sensitive patterns.
- Introducing a novel optimization problem for data sanitization, which can be solved by stochastic algorithms. The novel optimization model can be modified by domain experts with hyperparameters for side effects, allowing them to customize the solution according to their requirements.
- Developing two stochastic approaches, namely, PPUM-ALO and PPUM-IALO, for resolving the hiding problem.
- Demonstrating the flexibility and strength of the proposed algorithms using experiments and comparisons with alternative methods.

The remaining sections of this paper are structured as follows. A review of published works on HUIM, PPUM, and stochastic optimization is presented in Section 2. Section 3 provides background information about the PPUM problem. The particulars of the proposed PPUM methodology are outlined in Section 4. Section 5 presents comparative experimental results on typical datasets and demonstrates the superior effectiveness and efficiency of the proposed algorithms. Lastly, Section 6 summarizes the paper and discusses future research directions.

2 Related works

2.1 High-utility itemset mining

Utility mining is a broad research topic in data mining, addressing the problem of finding useful information in quantitative transactional databases. The first work in this area was published in 2004 [25]. The authors developed a high-utility itemset mining (HUIM) algorithm by using two types of utility information to identify useful hidden patterns. Since HUIs do not have the downward closure property, Liu [26] introduced the transaction-weighted utilization (TWU) concept to prune the unnecessary itemsets.

To mine patterns efficiently, specialized data structures need to be designed. Lin et al. [27] presented HUP-Tree. Tseng [28] designed UP-Tree followed by two further algorithms, UP-growth [28] and UP-growth+ [29]. In addition, the utility-list [30] structure was developed to find HUIs directly. The EFIM [31] algorithm represents the database and merge transactions with two upper bounds: revised subtree utility and local utility. Kim et al. [32] applied the sliding window method by inserting flow data.

For mining problems with uncertain external utilities, Gan et al. [33] introduced a mining approach for changing patterns. Dynamic databases have been mined with the MCH-miner [34] algorithm. Moreover, the MCUI-miner [35] algorithm combines a genetic algorithm and the MapReduce method to search for closed patterns. In distributed environments, Lin et al. [36] adapted clustering concepts to group transactions based on their correlations.

2.2 Privacy-preserving utility mining

With the advent of HUIM algorithms, privacy-preserving utility mining (PPUM) has become a crucial research topic. The PPUM problem was initially addressed by Yeh et al. [6], who proposed two heuristic algorithms, HHUIF and MSICF. The aim of these algorithms was to conceal SHUIs by reducing the quantities of items in transactions.

To speed up the hiding process, Yun et al. [37] developed a tree structure called FPUTT. With FPUTT, the sanitization process requires only three database scans. Although faster than earlier algorithms, FPUTT can produce non-compact trees on large datasets, leading to high memory usage and computational costs. Since scanning a database multiple times is time consuming, Liu et al. [38] constructed a T-table and an HUI-table. As it updates both the T-table and

HUI-table for each modified item, the algorithm's runtime remains high. In another effort to improve the speed of the sanitizing process, Yin and Yi [39] presented the utility-list dictionary structure, which takes a similar approach to the HI table [11], IT-table [40], and GIT-table [12].

Lin et al. [7] introduced the maximum sensitive utility concept and two heuristic algorithms: MSU-MAU and MSU-MIU. They also proposed three measurements for the side effects of the hiding process. In addition, Lin et al. [10, 41] proposed two approaches that involve a genetic algorithm, PPUMGA+insert and PPUMGAT. Sensitive items are hidden by inserting artificial transactions or deleting existing transactions. However, their changes in the database can produce false insights.

To tackle the remaining challenge of minimizing side effects, Liu et al. [42] proposed an enhanced version of the MSICF algorithm, IMSICF. The sanitization process incurs higher computational costs because this algorithm's dynamic computation of the conflict considers each sensitive item. Recently, three methods were introduced by Jangra et al. [8] based on dataset sorting techniques. The efficiency of the sorting techniques was demonstrated with three more algorithms [9]. Experiments showed that these algorithms have a better ability to lower side effects than earlier heuristic algorithms. A new concept named real item sensitive utility (RISU) was also described in this work.

A more general approach was introduced by Li et al. [11], in which the sanitization process is formulated as an optimization problem and solved with integer programming solvers. Although PPUM-ILP's [11] overall performance is good, constructing constraint satisfaction problems with many variables results in a significant running time. Duc et al. [40] resolved this drawback by presenting a fast formulation process for ILP approaches. In addition, a parallel method for preprocessing and formulating the sanitizing problem has been proposed [12] along with a relaxation of the hiding problem.

2.3 Stochastic optimization

In stochastic optimization algorithms, stochastic operators add random adjustments to the solution candidates, assisting in their escape from local optima. These optimization techniques draw inspiration from the natural world. They follow a common framework that begins with a set of random solutions, which are then refined and enhanced through various methods unique to each algorithm.

What distinguishes these algorithms is the specific approach they take to improve their solution sets. Genetic algorithms (GAs) [43] mimic natural selection. Particle swarm optimization (PSO) [44] originates from the swarming habit. Ant colony optimization (ACO) [45] emulates the movement of ants. These algorithms are effective and can solve a wide range of real-world problems.

Some more recent optimization algorithms are the grey wolf optimizer (GWO) [46], spotted hyena optimizer (SHO) [18], ant lion optimizer (ALO) [16], and forensic-based investigation (FBI) [47]. Variants of these algorithms have been developed for solving specific problems. This paper proposes stochastic approaches for PPUM. The ALO is chosen to find the solution because of its simplicity and because it can be guided by an appropriate initialization step.

Table 1 Item quantities in transactions

id	1	2	3	4	5
1	10	5	0	7	2
2	6	0	2	0	0
3	0	0	0	5	1
4	0	0	3	0	8
5	1	6	0	10	0
6	0	0	0	1	3
7	8	7	0	0	0
8	10	0	6	3	0
9	2	0	8	0	0
10	10	0	1	0	0

3 Preliminaries

Let $\mathcal{T}$ be a set of n transactions: $\mathcal{T} = \{T_1, T_2, \ldots, T_h\}$. A transaction $T_h \in \mathcal{T}$ contains distinct items. Each item has quantities in supporting transactions. Let $\mathcal{I} = \{i_1, i_2, \ldots, i_k\}$ be the set of distinct items in the database. A transaction T_h is a subset of $\mathcal{I}$. An item i_k in the database has its own profit. A quantitative transactional database $\mathcal{D}$ can be represented with two tables: Table 1 lists the transaction details and Table 2 the utilities of items.

Definition 1 (itemset) The term "itemset" denotes a set that consists of items.

Definition 2 (tidset) A tidset represents a set of unique identifiers for transactions.

Table 2 Item profits

Item	Profit
1	7
2	3
3	9
4	5
5	6

Definition 3 (internal utility) The internal utility of an item in a transaction is determined by its quantity. We denote the internal utility of item i_k in transaction T_h by $q(i_k, T_h)$.

For example, in Table 1, the internal utility of item 1 in transaction T_1 is 10, so $q(1, T_1) = 10$.

Definition 4 (external utility) External utility represents the significance of an item within a database. We denote the external utility of the item i_k by $e(i_k)$.

For example, in Table 2, the external utility of item 2 is 3, so $e(2) = 3$.

Definition 5 (utility of an item in a transaction) The utility of item i_k in transaction T_h, denoted by $u(i_k, T_h)$, is the product of its internal and external utilities:

$$u(i_k, T_h) = q(i_k, T_h) \times e(i_k). \tag{1}$$

The utility of item 2 in transaction T_1 of database $\mathcal{D}$ is $u(2, T_1) = q(2, T_1) \times e(2) = 5 \times 3 = 15$.

Definition 6 (utility of an itemset in a transaction) Let $u(A, T_h)$ denote the utility of itemset A in transaction T_h. If the itemset A is not supported by T_h, $u(X, T_h)$ will be zero. Otherwise, $u(A, T_h)$ refers to the sum of the utilities of all items in A in transaction T_h:

$$u(A, T_h) = \sum_{i_k \in A} u(i_k, T_h). \tag{2}$$

In $\mathcal{D}$, the utility of itemset $\{3, 5\}$ in transaction T_4 is computed as follows: $u(\{3, 5\}, T_4) = u(3, T_4) + u(5, T_4) = 3 \times 9 + 8 \times 6 = 75$. Because $\{3, 5\} \nsubseteq T_1$, $u(\{3, 5\}, T_1) = 0$.

Definition 7 (utility of an itemset in the database) The utility of itemset A in database $\mathcal{D}$ is determined by the sum of its utilities in supporting transactions:

$$u(A) = \sum_{T \in \mathcal{D}, A \subseteq T_h} u(A, T_h). \tag{3}$$

For example, the utility of itemset $\{4, 5\}$ in database $\mathcal{D}$ is calculated as follows: $u(\{4, 5\}) = u(\{4, 5\}, T_1) + u(\{4, 5\}, T_3) + u(\{4, 5\}, T_6) = (7 \times 5 + 2 \times 6) + (5 \times 5 + 1 \times 6) + (1 \times 5 + 3 \times 6) = 101$.

Definition 8 (minimum utility threshold) The minimum utility threshold, denoted by δ, is a measure used to identify whether the itemsets yield high utility. Its value is determined by the user.

Definition 9 (high-utility itemset) If an itemset A has a utility in the database equal to or greater than the minimum threshold δ, it qualifies as a high-utility itemset (HUI).

Changes in the database occur due to the hiding process affecting both sensitive and non-sensitive patterns. Lin et al. [7] suggested three measures for negative effects. To clarify the affection of hiding process to NSHUIs, Li et al. [11] introduced the concept of hiding cost.

Given a database $\mathcal{D}$, let $\mathcal{D}'$ denote the sanitized database, $\mathcal{H}$ the set of HUIs in $\mathcal{D}$, and $\mathcal{H}'$ the set of HUIs in $\mathcal{D}'$. Let $\mathcal{N}$ denote the set of non-sensitive HUIs (NSHUIs) in $\mathcal{D}$ and $\mathcal{S}$ the set of SHUIs in $\mathcal{D}$.

Definition 10 (hiding failure) The hiding failure is denoted by α and indicates the proportion of SHUIs in the database before and after sanitization:

$$\alpha = \frac{|\mathcal{S} \cap \mathcal{H}'|}{|\mathcal{S}|}. \tag{4}$$

For example, if $\alpha = 0.5$, then 50% of the sensitive itemsets are not hidden.

Definition 11 (missing cost) The ratio β, referred to as the missing cost, captures the percentage of NSHUIs that lose their utilities and cannot be identified in the perturbed database:

$$\beta = \frac{|\mathcal{N} - \mathcal{N} \cap \mathcal{H}'|}{|\mathcal{N}|}. \tag{5}$$

For example, if $\beta = 0.6$, then 60% of non-sensitive HUIs become low-utility itemsets.

Definition 12 (artificial cost) The artificial cost γ is the proportion of low-utility itemsets that transform into HUIs in the sanitized database $\mathcal{D}'$. This can be computed as follows:

$$\gamma = \frac{|\mathcal{H}' - \mathcal{H} \cap \mathcal{H}'|}{|\mathcal{N}|}. \tag{6}$$

For example, if $\gamma = 0.2$, then there is a 20% surplus of redundant HUIs in comparison to the original database.

Definition 13 (hiding cost) The hiding cost ϵ represents the changes to non-sensitive itemsets in the database:

$$\epsilon = \frac{|\mathcal{N} - \mathcal{H}'|}{|\mathcal{N}|}. \tag{7}$$

In this study, we evaluate results using the hiding failure, missing cost, and artificial cost. Consequently, the assessment of results does not take into account the hiding cost.

Problem statement 1 Given a quantitative transactional database $\mathcal{D}$, the main goal for PPUM is to transform it into a sanitized database $\mathcal{D}'$, while minimizing the negative effects on useful patterns in $\mathcal{D}$.

Table 3 The GIT table structure [12]

HUIs	Tidset	Size	Utility
I_1	TS_1	s_1	u_1
I_2	TS_2	s_2	u_2
I_3	TS_3	s_3	u_3

Table 4 An example variable list

	Coefficient	Original value
v_1^2	$e(i_2)$	$q(i_2, T_1)$
v_2^3	$e(i_3)$	$q(i_3, T_2)$
v_1^5	$e(i_2)$	$q(i_5, T_1)$

4 The proposed algorithms

In this section, we design a soft approach for the PPUM problem. The objective of all PPUM algorithms is to conceal sensitive patterns while minimizing side effects. To address this problem, we formulate the sanitizing process as an optimization problem.

4.1 Constructing constraint matrices

The first step relates to retaining the correlations between HUIs and their transactions. The GIT table is an effective structure that can serve parallel preprocessing tasks [12]. After preprocessing the original database, we obtain two GIT tables, namely, S-GIT for storing the SHUIs and N-GIT for NSHUIs. Table 3 visualizes the GIT table structure. For example, the transactions with identities in the TS_1 set support the HUI I_1, which has size s_1 and utility u_1 in the database.

The second step is to formulate the hiding problem. Since external utilities are fixed values, to hide SHUIs, we perturb the original database with artificial internal utilities. The replaced internal utilities must reduce the utilities of SHUIs below the minimum utility threshold δ to hide the SHUIs from mining algorithms. Let those internal utilities be the variables and let them be represented by the vector $\boldsymbol{v}$:

$$\boldsymbol{v} = \left[v_h^k\right], v_h^k \in \mathbb{N}^*, T_h \in \mathcal{D}, i_k \in \mathcal{I}. \tag{8}$$

The original values of the variables are denoted by $\boldsymbol{o}$. The coefficient of a variable is the external utility of its represented item; for example, the coefficient of v_{hk} is $e(i_k)$. Let $\boldsymbol{p}$ denote the coefficients of the variables in $\boldsymbol{v}$.

Hiding the problem solution is an assignment for $\boldsymbol{v}$. All SHUIs that we need to hide are stored in the S-GIT table. By scanning the S-GIT table, we can define $\boldsymbol{v}$ and organize the positions of variables in the database by a hash structure named a variable list.

Definition 14 (variable list) A variable list is a hash structure that consists of the coefficients and original values of variables. The position of a variable in the database (the item and the transaction that supports it) acts as the index (Table 4).

After that, we re-scan S-GIT, construct constraints to hide the SHUIs, and represent them by a constraint matrix $\boldsymbol{S}$ of size $|\mathcal{S}| \times \|\boldsymbol{v}\|_0$. Let $\boldsymbol{d}$ be the vector in which all values are the minimum utility threshold δ. To hide the SHUIs, we have

$$\boldsymbol{Sv} - \boldsymbol{d} \leq \boldsymbol{0}. \tag{9}$$

Lowering the utilities of SHUIs also affects NSHUIs. The N-GIT table stores the HUIs that are affected by the hiding process. For an HUI X and the set of transactions TS_X that support it, we combine each item $i_k \in X$ with transaction $T_h \in TS_X$. An item–transaction pair u_h^k is found in the variable list and has two states:

$$u_h^k = \begin{cases} v_h^k, & \text{If } v_h^k \in \boldsymbol{v}, \\ q(i_k, T_h), & \text{otherwise.} \end{cases} \tag{10}$$

After scanning N-GIT, we obtain a constraint matrix N with a size of $|\mathcal{N}| \times \|\boldsymbol{v}\|_0$. Algorithm 1 describes in detail the whole process of establishing constraint matrices (Table 5).

4.2 Formulating the optimization problem

Suppose the elements of $\boldsymbol{r}$ are the remaining utilities of itemsets in the NSHUIs after variable replacement. To retain the NSHUIs, we have

$$\boldsymbol{Nv} + \boldsymbol{r} - \boldsymbol{d} \geq \boldsymbol{0}. \tag{11}$$

Li et al. [11] suggested an integer linear programming model:

$$\begin{aligned} &\arg\min_{\boldsymbol{v}} \|\boldsymbol{v}\|_1 && v_h^k \in \mathbb{N}^* \\ &\text{s.t.} \quad \boldsymbol{Sv} - \boldsymbol{d} \leq \boldsymbol{0}, \\ &\qquad \boldsymbol{Nv} + \boldsymbol{r} - \boldsymbol{d} \geq \boldsymbol{0}. \end{aligned} \tag{12}$$

However, this model does not always have a feasible solution. Recently, Duc et al. [12] proposed a relaxation:

$$\begin{aligned} &\arg\min_{\boldsymbol{v}} \|\boldsymbol{v}\|_1 + \|\boldsymbol{q}\|_1 && v_h^k \in \mathbb{N}^*, q_i \in \mathbb{R} \\ &\text{s.t.} \quad \boldsymbol{Sv} - \boldsymbol{d} \leq \boldsymbol{0}, \\ &\qquad \boldsymbol{Nv} + \boldsymbol{r} - \boldsymbol{d} \geq \boldsymbol{q}. \end{aligned} \tag{13}$$

The constraint satisfaction problems (12) and (13) have been shown to be efficient approaches to minimizing the hiding

Table 5 Side effects of heuristic algorithms on the mushrooms and foodmart datasets

Dataset	SIP(%)	SMRF β(%)	SDIF β(%)	SLRF β(%)	HHUIF β(%)	MSICF β(%)	MSU-MAU β(%)	MSU-MIU β(%)
Mushrooms	0.5	71.0	46.4	62.1	77.2	56.2	76.8	62.9
	0.6	87.7	84.1	89.6	99.5	83.5	99.5	97.5
	0.7	92.1	97.5	94.1	99.9	92.4	99.9	98.8
	0.8	83.9	86.3	90.7	99.7	95.0	99.9	99.6
Foodmart	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	0.6	22.9	24.9	25.7	26.1	26.9	26.1	21.0
	0.7	12.1	3.3	3.3	12.1	2.9	12.1	13.7
	0.8	12.8	14.0	14.0	12.1	7.4	12.1	12.8

Algorithm 1 Establishing constraint matrices.

Input: N-GIT, S-GIT tables
Output: Variable vector $\boldsymbol{v}$, coefficient vector $\boldsymbol{p}$, constraint matrices $\boldsymbol{S}$, $\boldsymbol{N}$, remaining utility vector $\boldsymbol{r}$

Initialize variable vector $\boldsymbol{v}$
Initialize matrices $\boldsymbol{S}$, $\boldsymbol{N}$
for itemset, tidset, size, util in S-GIT **do**
  Combine itemset, tidset into $\boldsymbol{t}$
  for variable $v_h^k = t_i$ in $\boldsymbol{t}$ **do**
    Add v_h^k to $\boldsymbol{v}$
    Add $e(i_k)$ to $\boldsymbol{p}$
  end for
end for
for itemset, tidset, size, util in S-GIT **do**
  Combine itemset, tidset into $\boldsymbol{t}$
  Initialize vector $\boldsymbol{t_1}$ with $\|\boldsymbol{v}\|_0$ zero values
  for t_i in $\boldsymbol{t}$ **do**
    if $v_i = t_i$ in $\boldsymbol{v}$ **then**
      $t_1^i = 1$ ▷ $t_1^i \in \boldsymbol{t_1}$
    end if
  end for
  Add $\boldsymbol{t_1}$ to $\boldsymbol{S}$
end for
Initialize remaining util vector $\boldsymbol{r}$
for itemset, tidset, size, util in N-GIT **do**
  Combine itemset, tidset into $\boldsymbol{t}$
  Initialize vector $\boldsymbol{t_1}$ with $\|\boldsymbol{v}\|_0$ zero values
  for t_i in $\boldsymbol{t}$ **do**
    if $v_i = t_i$ in $\boldsymbol{v}$ **then**
      $t_1^i = 1$ ▷ $t_1^i \in \boldsymbol{t_1}$
      $r = \text{util} - p[t_i] \times o[t_i]$
    end if
  end for
  Add $\boldsymbol{t_1}$ to $\boldsymbol{N}$
  Add r to $\boldsymbol{r}$
end for
$\boldsymbol{S} = \boldsymbol{Sp}$; $\boldsymbol{N} = \boldsymbol{Np}$
return $\boldsymbol{v}$, $\boldsymbol{p}$, $\boldsymbol{r}$, $\boldsymbol{S}$, $\boldsymbol{N}$

cost of the sanitizing process. However, there are no specific restrictions on the redundant information. Furthermore, they are incompatible with stochastic optimization algorithms. In [11, 12, 40], the authors used integer programming solvers to find solutions.

To tackle this problem, we have designed a new model compatible with a wide range of optimization methods. Let $\min(\cdot)$ and $\max(\cdot)$ be the element-wise minimum and maximum functions, so that

$$\boldsymbol{m} = \min(\boldsymbol{a}, \boldsymbol{b}) \qquad \boldsymbol{a}, \boldsymbol{b}, \boldsymbol{m} \in \mathbb{R}^d$$
$$= \begin{bmatrix} \min(a_1, b_1) \\ \min(a_2, b_2) \\ \vdots \\ \min(a_n, b_n) \end{bmatrix} \tag{14}$$

and

$$\boldsymbol{n} = \max(\boldsymbol{a}, \boldsymbol{b}) \qquad \boldsymbol{a}, \boldsymbol{b}, \boldsymbol{n} \in \mathbb{R}^d$$
$$= \begin{bmatrix} \max(a_1, b_1) \\ \max(a_2, b_2) \\ \vdots \\ \max(a_n, b_n) \end{bmatrix}. \tag{15}$$

Let $\boldsymbol{w}_1$, $\boldsymbol{w}_2$, and $\boldsymbol{w}_3$ be adjustable weight vectors defined according to the user's preference. These respectively control the hiding failure, missing cost, and artificial cost of the hiding process. In particular, each value w_1^i in $\boldsymbol{w}_1$ corresponds to an SHUI. Formally, we have

$$\arg\min_{\boldsymbol{v}} \quad \|\boldsymbol{w}_1 \circ \max(\boldsymbol{0}, \boldsymbol{Sv} - \boldsymbol{d})\|_1 \qquad v_h^k \in \mathbb{N}^*. \tag{16}$$

Similarly, for the missing cost,

$$\arg\min_{\boldsymbol{v}} \quad \|\boldsymbol{w}_2 \circ \max(\boldsymbol{0}, \boldsymbol{d} - (\boldsymbol{Nv} + \boldsymbol{r})\|_1 \quad v_h^k \in \mathbb{N}^*. \tag{17}$$

Finding an optimal solution to PPUM is an NP-hard problem. The number of itemsets is $2^{|\mathcal{I}|}$, which means there can be the same number of constraints. Therefore, it is impossible to declare a model that fully restricts the side effects. We define a soft approach to limit the redundant patterns:

$$\arg\min_{\boldsymbol{v}} \quad \|\boldsymbol{w}_3 \circ \boldsymbol{p} \circ \boldsymbol{v} - \boldsymbol{p} \circ \boldsymbol{o}\|_1 \qquad v_h^k \in \mathbb{N}^*. \tag{18}$$

Formally, our soft model can be described as follows:

$$\begin{aligned} \arg\min_{\boldsymbol{v}} \quad & \|\boldsymbol{w}_1 \circ \max(\boldsymbol{0}, \boldsymbol{S}\boldsymbol{v} - \boldsymbol{d})\|_1 + \qquad v_h^k \in \mathbb{N}^* \\ & \|\boldsymbol{w}_2 \circ \max(\boldsymbol{0}, \boldsymbol{d} - (\boldsymbol{N}\boldsymbol{v} + \boldsymbol{r}))\|_1 + \\ & \|\boldsymbol{w}_3 \circ (\boldsymbol{p} \circ \boldsymbol{v} - \boldsymbol{p} \circ \boldsymbol{o})\|_1. \end{aligned} \tag{19}$$

4.3 PPUM-ALO and PPUM-IALO algorithms

After establishing the optimization model, the next step is finding the optimal solution. In this paper, we propose two algorithms: PPUM-ALO and PPUM-IALO. The PPUM-ALO algorithm applies the ALO to find the solution of the optimization problem (19). The ALO [16] is an optimization algorithm inspired by the hunting behavior of ant lions. These creatures are known for creating conical pits in sandy soil and waiting at the bottom for their prey to fall in. Figure 1 illustrates a conical trap made by an ant lion.

First, we model the random walk of ants when searching for food. Let g be the maximum number of iterations and t the random walk step. A stochastic function $r(\cdot)$ is defined as follows:

$$r(\cdot) = \begin{cases} 1, & \text{if } x \sim U(0,1) > 0.5, \\ 0, & \text{if } x \sim U(0,1) \leq 0.5. \end{cases} \tag{20}$$

Letting $c(\cdot)$ be the cumulative sum function, a random walk at step t is

$$\boldsymbol{x}_t = \left[0, c\left(2r(t_1) - 1\right), c\left(2r(t_2) - 1\right), \ldots, c\left(2r(t_g) - 1\right)\right]. \tag{21}$$

Suppose the optimization problem has v variables. The positions of the ants form a $g \times v$ matrix:

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{a}_1 \\ \boldsymbol{a}_2 \\ \vdots \\ \boldsymbol{a}_g \end{bmatrix} = \begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,v} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,v} \\ \vdots & \vdots & \vdots & \vdots \\ a_{g,1} & a_{g,2} & \cdots & a_{g,v} \end{bmatrix}. \tag{22}$$

Fig. 1 The ALO mimics the hunting behavior of an ant lion inside a conical trap

Second, assuming that ant lions are somewhere in the search space,

$$\boldsymbol{L} = \begin{bmatrix} \boldsymbol{l}_1 \\ \boldsymbol{l}_2 \\ \vdots \\ \boldsymbol{l}_g \end{bmatrix} = \begin{bmatrix} l_{1,1}\ l_{1,2} \cdots l_{1,v} \\ l_{2,1}\ l_{2,2} \cdots l_{2,v} \\ \vdots\ \ \vdots\ \ \vdots\ \ \vdots \\ l_{g,1}\ l_{g,2} \cdots l_{g,v} \end{bmatrix}. \tag{23}$$

To formulate the bounds of the optimization problem, we limit the random walks inside the search space by normalizing $\boldsymbol{x}$ for each variable j:

$$\hat{\boldsymbol{x}_j} = \frac{\boldsymbol{x}_j}{\|\boldsymbol{x}_j\|}. \tag{24}$$

To mimic the hunting behavior of ant lions, we define a hypersphere by two vectors:

$$\boldsymbol{c} = \min(\boldsymbol{x}) \tag{25}$$

$$\boldsymbol{d} = \max(\boldsymbol{x}). \tag{26}$$

Ant lion hunting is modeled with a roulette wheel selection method. When prey enters the trap, the ant lion throws sand from the center of the pit. This means that the hypersphere radius is decreased adaptively at step t by a ratio $r = 10^{w\frac{t}{g}}$. The value of w changes over time as suggested in [16]:

$$\boldsymbol{c} = \frac{1}{r}\boldsymbol{c} \tag{27}$$

$$\boldsymbol{d} = \frac{1}{r}\boldsymbol{c}. \tag{28}$$

Figure 2 shows the random walk of an ant in the ant lion's trap. When the ant lion catches an ant, it moves to its position to consume it. The ant lion yielding the best solution is the elite. The elite will affect the movements of all ants. Letting $\boldsymbol{R}_l$ be the random walks around the ant lion, $\boldsymbol{R}_e$ consists of the random walks around the elite:

$$\boldsymbol{A} = \frac{\boldsymbol{R}_l + \boldsymbol{R}_e}{2}. \tag{29}$$

The solution of the optimization problem is the position of the elite after the ALO algorithm is executed. Variable values are used to replace the internal utilities in the original database to obtain the sanitized one.

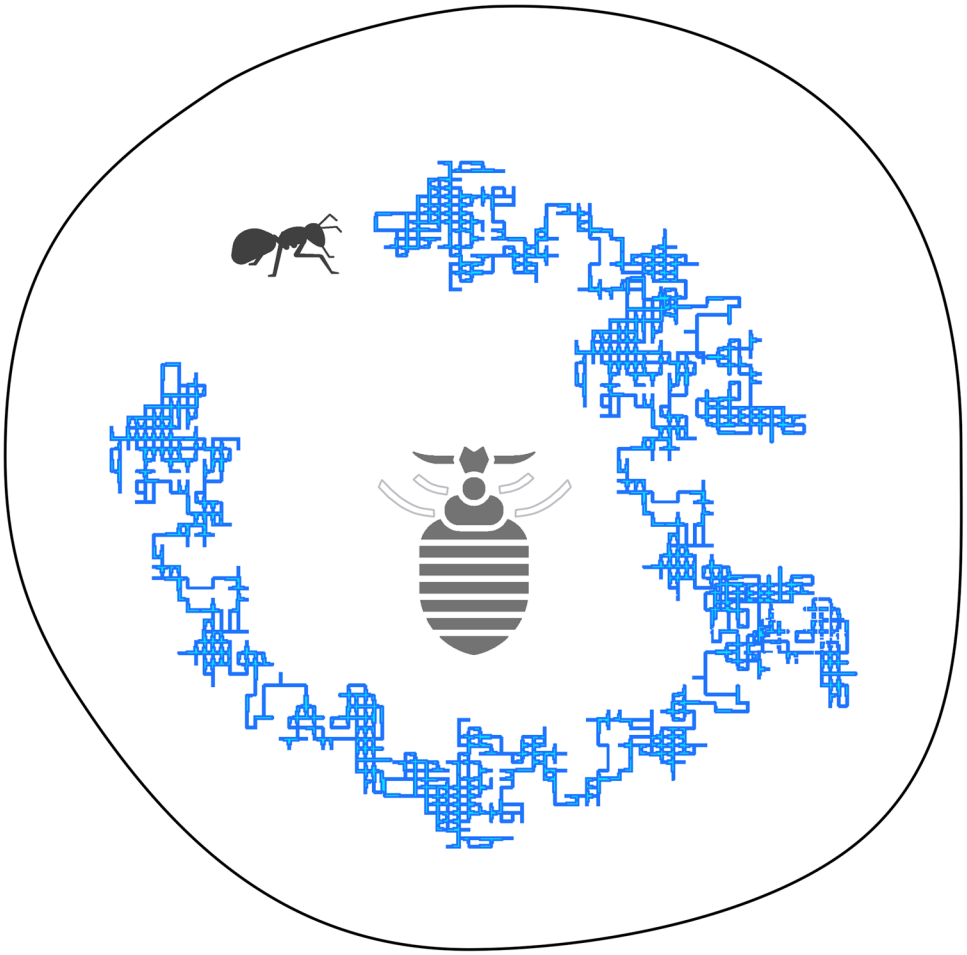

Fig. 2 An ant's random walk inside a trap

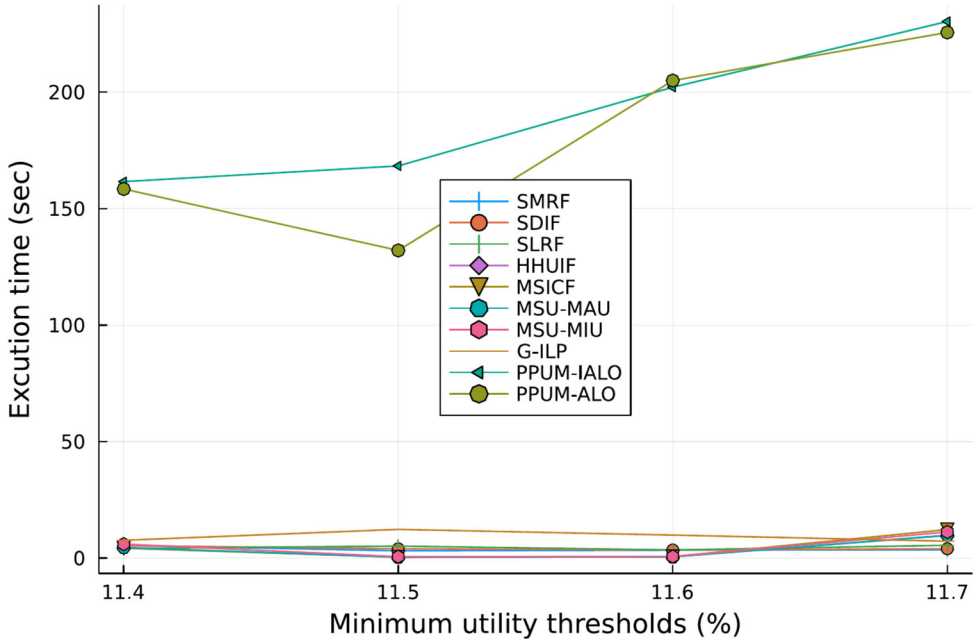

Fig. 3 Variation of algorithm runtimes with the minimum utility thresholds on the mushrooms dataset

We combine the exact method of a mathematical solver with the stochastic method of the ALO to form a hybrid algorithm. First, we solve the optimization model with a mathematical optimization method, namely, the Gurobi optimizer [48]. The solution is used to initialize the elite ant lion since it will affect the movements of all the ants during the iterations. We execute the ALO for only twenty iterations. This algorithm is named PPUM-IALO.

5 Experiments

We conducted experiments on an Intel Core i7-6700 CPU with 32 GB of RAM and an NVIDIA GeForce RTX2080 GPU. The algorithms were executed on four typical datasets. The density of each dataset was calculated using the following formula:

$$\text{Density} = \frac{\text{AvgLen}}{|\mathcal{I}|}.$$

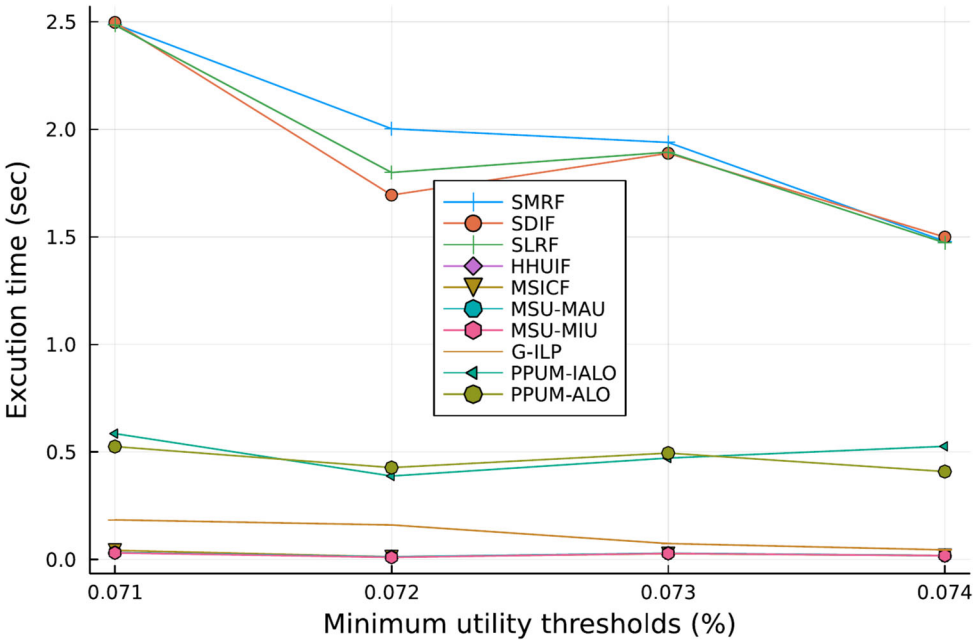

Fig. 4 Variation of algorithm runtimes with the minimum utility threshold on the foodmart dataset

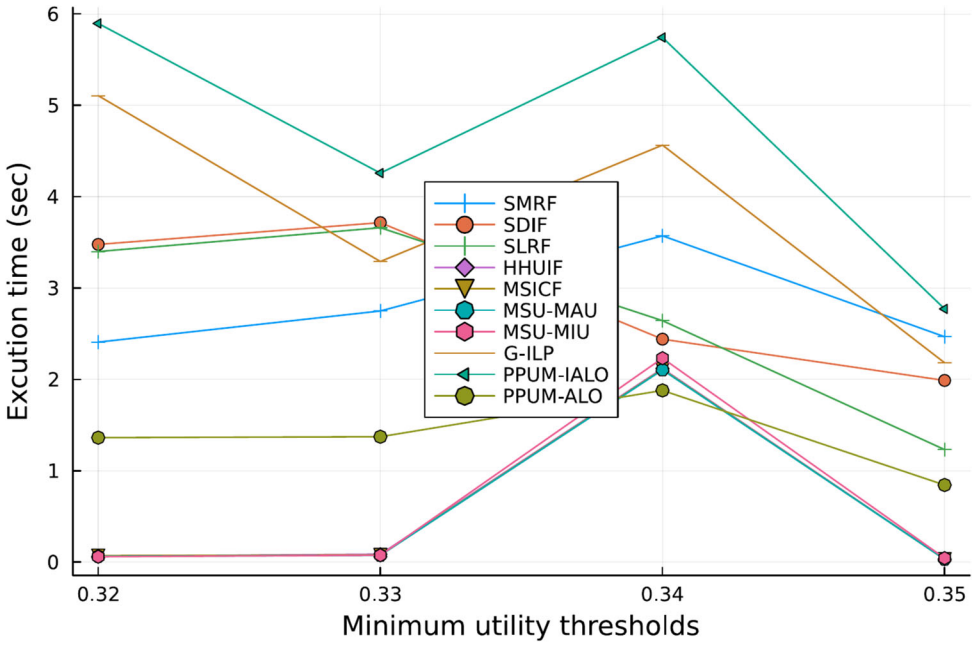

Fig. 5 Variation of algorithm runtimes with the minimum utility threshold on the t25i10d10k dataset

The properties of the experimental datasets are described in Table 9. The external utility of each item was drawn from a log-normal distribution truncated to the interval $\mathbb{R} \cap [1, 10]$. The internal utilities of the items in a transaction were taken to be integers sampled from $U(1, 10)$. According to the survey in Zhang et al. [49], the d2HUP algorithm [50] is preferred as the mining technique. The performance of the proposed algorithm was evaluated and compared with other algorithms on different datasets with regard to running time, hiding failure (α), missing cost (β), and artificial cost (γ). Several heuristic algorithms, HHUIF, MSICF [6], MSU-MAU, MSU-MIU [7], SMRF, SLRF, and SDIF [9], and the newest integer programming approach, GILP [12], were selected for comparison with the proposed algorithms. The integer programming problem was solved using the Gurobi solver [48]. All algorithms were implemented in Julia version 1.6.6, a high-performance language for scientific computing. We conducted experiments under varying minimum utility thresholds and different numbers of sensitive patterns (NSPs) and sensitive information percentages (SIPs). Each algorithm evaluation measure was averaged over three runs.

Both the PPUM-ALO and PPUM-IALO algorithms were used with ten ants and were stopped early at twenty iterations. To control the side effects, the vector $\boldsymbol{w_1}$ was filled with positive infinity to eliminate hiding failure completely. Since introducing false information can cause serious problems, we let $\boldsymbol{w_2}$ be an all-ones vector and set all values of the vector $\boldsymbol{w_3}$ to 20 to minimize the artificial cost. The maximum iterations of PPUM-ALO and PPUM-IALO were set to 100 and 20, respectively.

5.1 Running time

As shown in Fig. 3, the stochastic methods required a high execution time on the mushrooms dataset, which is small and dense. A high-density dataset tends to have more items shared between NSHUIs and SHUIs, creating more conflicts between the objectives of reducing the missing cost and hid-

Table 6 Side effects of heuristic algorithms on the t25i10d10k and t20i6d100k datasets

Dataset	NSPs	SMRF β(%)	SDIF β(%)	SLRF β(%)	HHUIF β(%)	MSICF β(%)	MSU-MAU β(%)	MSU-MIU β(%)
t25i10d10k	1	50.6	50.6	68.3	68.9	67.3	74.3	56.7
	2	48.8	51.3	44.2	38.2	39.0	43.1	37.4
	3	41.1	41.0	71.4	59.0	58.9	51.6	40.2
	4	66.4	53.0	56.2	47.3	47.3	56.1	40.7
t20i6d100k	2	9.3	10.4	10.4	9.3	9.3	8.8	7.7
	3	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	4	8.3	7.8	7.8	7.8	5.6	8.9	6.7
	5	12.8	11.2	10.6	12.8	11.2	12.3	10.1

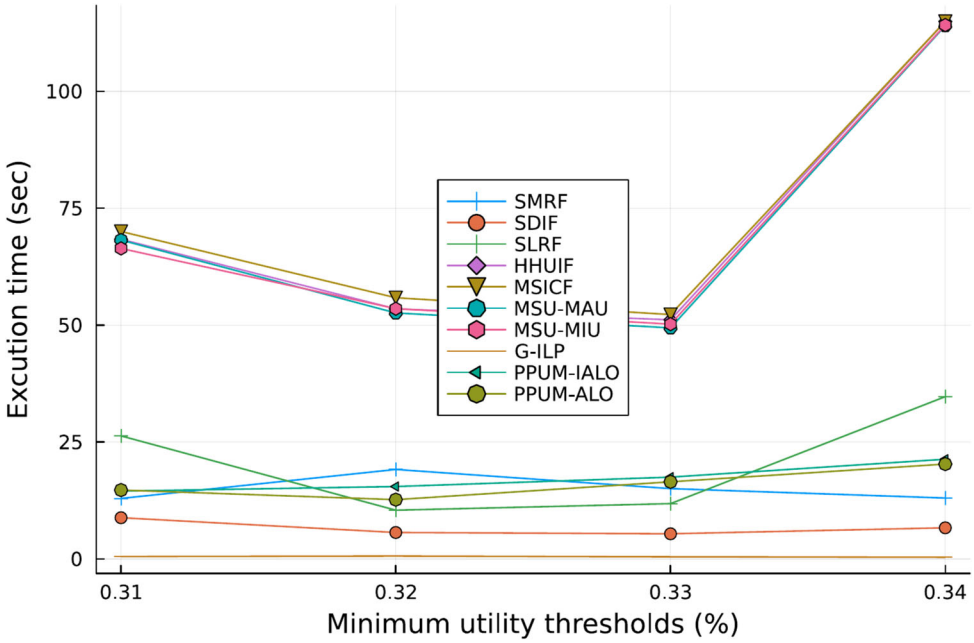

Fig. 6 Variation of algorithm runtimes with the minimum utility threshold on the t20i6d100k dataset

ing sensitive data. This makes the formulated hiding problem more difficult to solve. Heuristic algorithms hide sensitive patterns by iteratively removing or reducing target internal utilities, giving them advantages on small datasets. However, the G-ILP algorithm performs well on mushrooms. As the minimum utility threshold was set to 11.7% of the total utility of the dataset, we increased the percentages of sensitive information (Fig. 7). We can observe that the proposed algorithms have difficulties in dense datasets. Because the number of variables in a dense dataset can be large, computing the fitness function for the whole population is a time-consuming task.

As shown in Figs. 4 and 8, all algorithms had low running times on the foodmart dataset, but the SMRF, SDIF, and SLRF algorithms seemed to run slightly more slowly. This outcome is expected because the algorithms scan the dataset and SHUIs to construct data structures that preserve transaction weights and real itemset utility values (RISUs) before concealing the SHUIs. With a fixed minimum utility threshold, hiding the more sensitive patterns requires higher algorithm execution times.

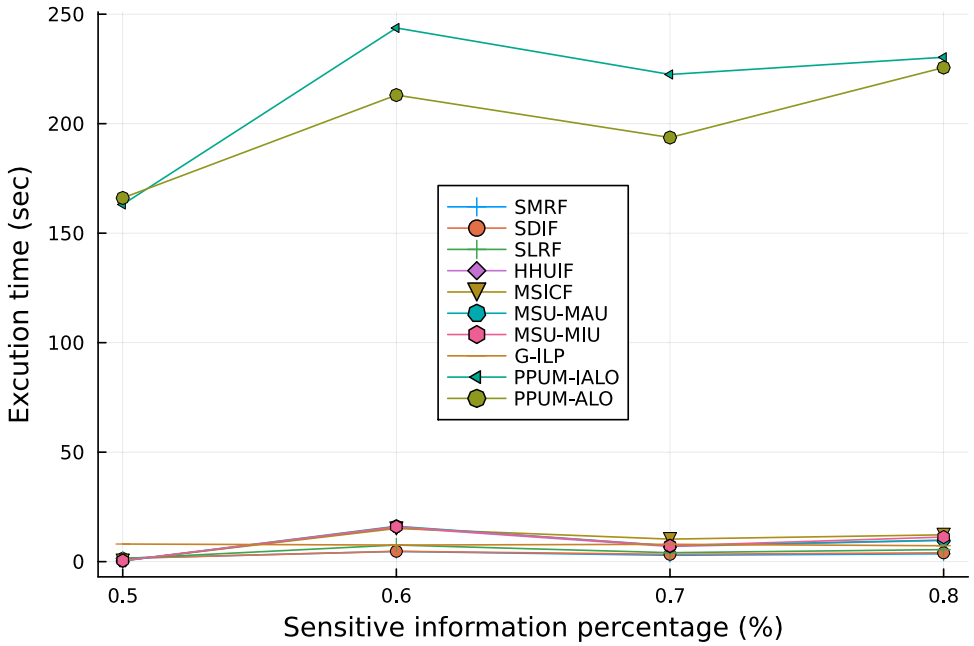

Fig. 7 Variation of algorithm runtimes with the number of sensitive patterns on the mushrooms dataset

Table 7 Side effects of optimization-based algorithms on the mushrooms and foodmart datasets

Dataset	SIP(%)	G-ILP		PPUM-IALO		PPUM-ALO	
		$\beta(\%)$	$\gamma(\%)$	$\beta(\%)$	$\gamma(\%)$	$\beta(\%)$	$\gamma(\%)$
Mushrooms	0.5	4.7	472.3	4.7	472.3	n/a	n/a
	0.6	9.6	466.5	9.6	466.5	n/a	n/a
	0.7	24.4	636.4	24.4	636.4	n/a	n/a
	0.8	5.9	112.5	5.9	112.5	n/a	n/a
Foodmart	0.5	0.0	0.0	0.0	0.0	0.0	0.0
	0.6	5.5	7.8	27.9	0.0	27.9	0.0
	0.7	0.0	399.3	25.2	0.0	25.3	0.0
	0.8	0.8	321.9	24.7	0.0	24.7	0.0

Table 8 Side effects of optimization-based algorithms on the t25i10d10k and t20i6d100k datasets

Dataset	NSPs	G-ILP		PPUM-IALO		PPUM-ALO	
		$\beta(\%)$	$\gamma(\%)$	$\beta(\%)$	$\gamma(\%)$	$\beta(\%)$	$\gamma(\%)$
t25i10d10k	1	31.6	8.1	31.6	8.1	95.2	0.0
	2	1.3	81.7	1.3	81.7	95.2	0.0
	3	10.9	21.2	10.9	21.2	95.2	0.0
	4	n/a	n/a	95.2	0.0	95.2	0.0
t20i6d100k	2	0.0	9.9	0.0	9.9	23.6	0.0
	3	0.0	0.0	0.0	0.0	0.0	0.0
	4	0.6	23.9	11.7	0.0	11.7	0.0
	5	0.0	12.3	0.0	12.3	24.0	0.0

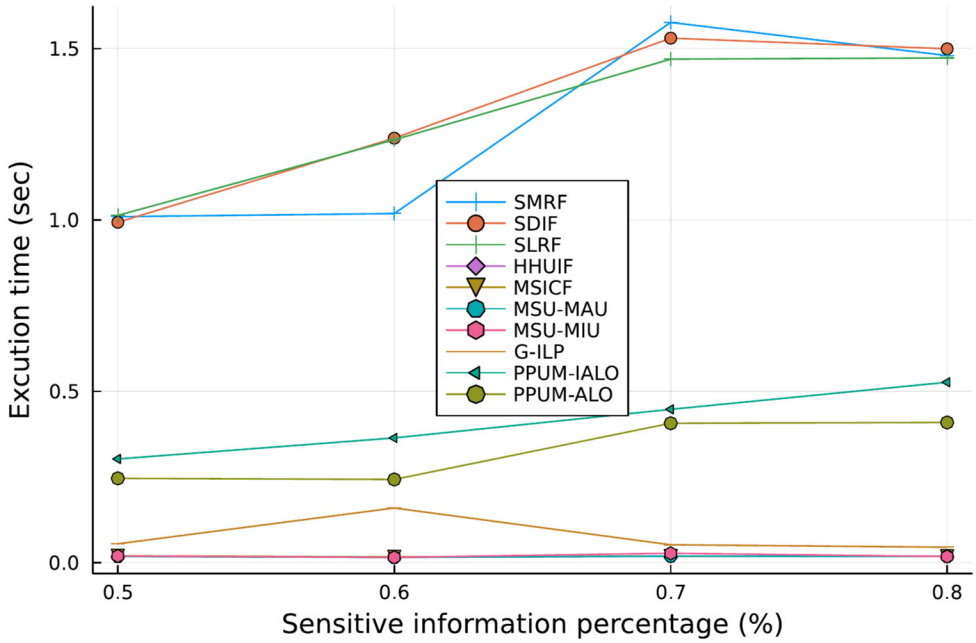

Fig. 8 Variation of algorithm runtimes with the number of sensitive patterns on the foodmart dataset

Table 9 Datasets and their properties

Dataset	$\|\mathcal{D}\|$	$\|\mathcal{I}\|$	Density(%)
Mushrooms	8124	119	19.3
Foodmart	4141	1559	0.25
t25i10d10k	9976	893	2.66
t20i6d100k	99922	929	2.22

The algorithms tend to run more slowly on large datasets. In Fig. 5, the ILP-based approaches, that is, G-ILP and PPUM-IALO, have high overall execution times. The stochastic approach PPUM-ALO has reasonable results. In Fig. 9, there is a case in which the ILP problem is mathematically harder to solve. This causes unexpectedly high running times with ILP-based approaches (Table 6).

The disadvantages of heuristic algorithms are apparent in large and sparse datasets (Fig. 6). Their running times were high on the t20i6d100k dataset. We can also find that the execution time of an algorithm is affected by the chosen sensitive patterns. Because the heuristic algorithms only modify an internal utility in an iteration, SHUIs that appear in many transactions need more database scans to hide. As can be observed in Figs. 10 and 6, algorithms based on optimization problems perform more effectively on large and sparse datasets.

5.2 Side effects

The missing and artificial costs of the algorithms on the mushrooms and foodmart datasets are shown in Tables 5 and 7. On the mushrooms dataset, the proposed PPUM-ALO algorithm failed to hide SHUIs and the PPUM-IALO algorithm achieved similar results to G-ILP. This means that without an elite as a guide, the ALO cannot escape the local minima of the hiding problem. The PPUM-IALO was initialized with the G-ILP solution and took fewer steps to find a more suitable solution for specific requirements corresponding to user-defined hyperparameters. This behavior can be attributed to the side effects of algorithms on the foodmart dataset (Fig. 7).

Table 11 TMRs of optimization-based algorithms on the mushrooms and foodmart datasets

Dataset	SIP(%)	G-ILP	PPUM-IALO	PPUM-ALO
Mushrooms	0.5	20.5	20.5	0.0
	0.6	67.3	67.3	0.0
	0.7	51.7	51.7	0.0
	0.8	49.1	49.1	0.0
Foodmart	0.5	1.1	1.1	1.2
	0.6	0.5	0.5	0.5
	0.7	1.0	1.0	1.1
	0.8	1.5	1.5	81.1

For large datasets (see Tables 6 and 8), optimization-based algorithms usually perform better with a trade-off in the artificial cost. However, due to the strict constraints of G-ILP, there was a case in which it could not solve the hiding problem properly. In addition, the proposed PPUM-IALO obtained more suitable results than G-ILP on the t20i6d100k dataset (Fig. 8).

5.3 Transaction modification ratio

To measure the impact of the hiding process, we used the transaction modification ratio (TMR) to determine the percentage of modified transactions for each experimental dataset. The TMR is lower for heuristic approaches, as indicated by Tables 9, 10 and 11. There was an unforeseen case in the foodmart dataset that led these approaches to have unexpectedly high TMRs. In the foodmart dataset, the PPUM-ALO algorithm also had a perturbed solution that affected 80% of its transactions. The PPUM-ALO algorithm failed to find hiding solutions for the mushrooms dataset, resulting in TMRs of zero (Fig. 9).

Tables 12 and 13 show the TMRs of algorithms on large datasets. In Table 12, it can be seen that even with smaller TMRs, heuristic algorithms still have higher exe-

Table 10 TMRs of heuristic algorithms on the mushrooms and foodmart datasets

Dataset	SIP(%)	SMRF	SDIF	SLRF	HHUIF	MSICF	MSU-MAU	MSU-MIU
Mushrooms	0.5	3.2	1.7	1.7	1.6	1.9	1.3	1.5
	0.6	67.3	67.3	67.3	67.3	67.3	67.3	67.3
	0.7	29.2	29.2	29.3	29.2	29.2	29.2	29.6
	0.8	40.8	40.8	40.8	40.8	41.1	40.8	40.8
Foodmart	0.5	0.5	0.5	0.5	0.4	0.4	0.4	0.4
	0.6	46.4	46.4	46.4	46.3	46.3	46.3	46.3
	0.7	0.5	0.5	0.5	0.4	0.4	0.4	0.4
	0.8	0.6	0.5	0.6	0.4	0.4	0.4	0.4

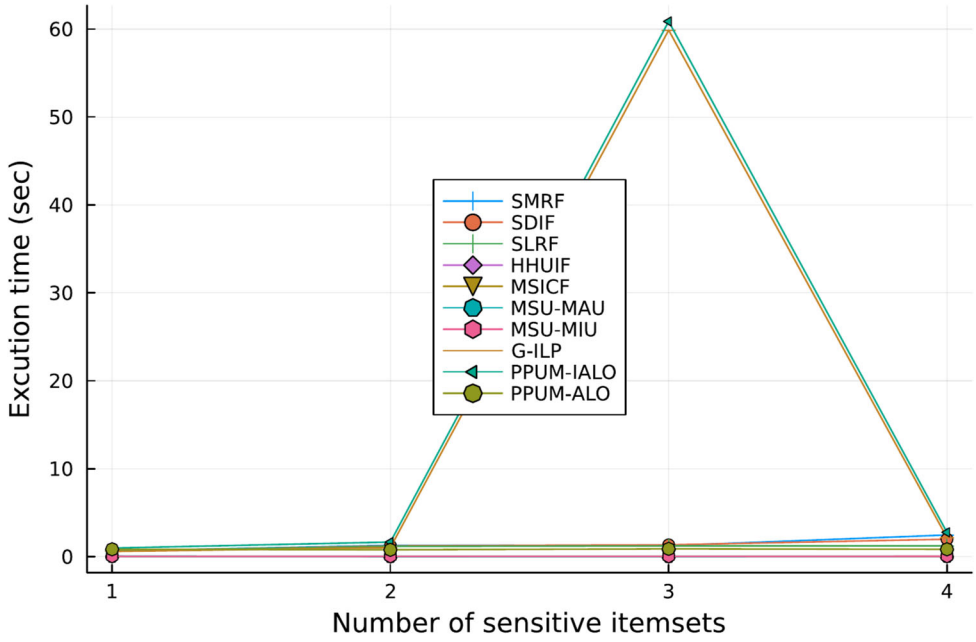

Fig. 9 Variation of algorithm runtimes with the number of sensitive patterns on the t25i10d10k dataset

Table 12 TMRs of heuristic algorithms on the t25i10d10k and t20i6d100k datasets

Dataset	NSP	SMRF	SDIF	SLRF	HHUIF	MSICF	MSU-MAU	MSU-MIU
t25i10d10k	1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
	2	0.1	0.1	0.1	0.1	0.1	0.1	0.1
	3	0.1	0.1	0.2	0.1	0.1	0.1	0.1
	4	0.2	0.1	0.1	0.1	0.1	0.1	0.1
t20i6d100k	2	0.0	0.1	0.1	0.0	0.0	0.0	0.0
	3	3.1	0.5	3.1	1.9	1.9	1.9	1.9
	4	0.8	0.8	0.8	0.6	0.6	0.6	0.6
	5	3.3	3.3	3.3	2.0	2.0	2.0	2.0

Table 13 TMRs of optimization-based algorithms on the t25i10d10k and t20i6d100k datasets

Dataset	NSP	G-ILP	PPUM-IALO	PPUM-ALO
t25i10d10k	1	0.9	0.9	0.9
	2	0.9	0.9	0.9
	3	1.0	1.0	1.0
	4	0.0	1.0	1.0
t20i6d100k	2	0.6	0.6	0.6
	3	12.3	12.3	12.3
	4	8.4	8.4	8.4
	5	17.4	17.4	17.4

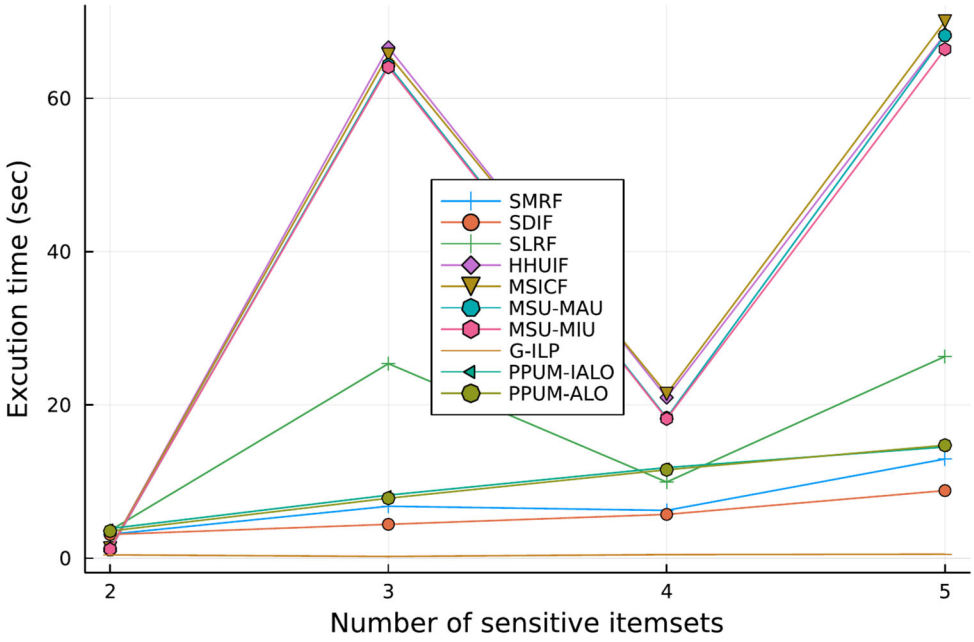

Fig. 10 Variation of algorithm runtimes with the number of sensitive patterns on the t20i6d100k dataset

cution times than optimization-based algorithms. The side effects of optimization-based algorithms can be different, but their TMRs remain the same (Fig. 10).

6 Conclusion

With the growing concern for privacy, researchers are focusing on the potential privacy breaches caused by high-utility itemset mining (HUIM). In this study, we propose two algorithms, PPUM-ALO and PPUM-IALO, for PPUM. To construct the hiding problems, we design a strategy for establishing constraint matrices and introduce a novel hiding model. Stochastic optimization is the approach we use to find the hiding solution. Our algorithms show stable performance in running time and side effects through extensive experimental analysis and comparison. However, hiding sensitive information is an NP-hard problem with many unexpected cases. Therefore, it is crucial to experiment with different optimization algorithms. Moreover, the optimization approach does not allow deleting items from transactions. There are cases in which reducing internal utilities is not enough to hide itemsets. Developing a perturbation algorithm that allows removing items from transactions is necessary for future research.

Acknowledgements This research is funded by University of Science, VNU-HCM under grant number CNTT 2024-01.

Author Contributions Duc Nguyen: Conceptualization, Methodology, Validation, Software, Writing—original draft. Bac Le: Conceptualization, Methodology, Validation, Supervision, Review, Declarations.

Data Availability The datasets generated and/or analyzed during the current study are available in the GitHub repository, https://github.com/4ree/ILP.

Code Availability Code for preprocessing and analysis is provided in the GitHub repository, https://github.com/4ree/ILP.

Declarations

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

Informed Consent Informed consent was obtained from all of the subjects involved in this study.

References

1. Gheisari M, Hamidpour H, Liu Y, Saedi P, Raza A, Jalili A, Rokhsati H, Amin R (2022) Data mining techniques for web mining: a survey. Artif Intell Appl 1(1):3–10. https://doi.org/10.47852/bonviewAIA2202290
2. Wu JM-T, Gautam S, Jolfaei A, Fournier-Viger P, Lin JC-W (2021) Hiding sensitive information in ehealth datasets. Futur Gener Comput Syst 117:169–180. https://doi.org/10.1016/j.future.2020.11.026
3. Zhang C, Xie Y, Bai H, Yu B, Li W, Gao Y (2021) A survey on federated learning. Knowl-Based Syst 216:106775. https://doi.org/10.1016/j.knosys.2021.106775
4. Liu Y, Kang Y, Xing C, Chen T, Yang Q (2020) A secure federated transfer learning framework. IEEE Intell Syst 35(4):70–82. https://doi.org/10.1109/MIS.2020.2988525
5. Yun U, Kim D (2017) Analysis of privacy preserving approaches in high utility pattern mining. In: Park JJJH, Pan Y, Yi G, Loia V (eds.) Advances in computer science and ubiquitous comput-

ing, Singapore, pp 883–887. https://doi.org/10.1007/978-981-10-3023-9_137
6. Yeh J-S, Hsu P-C (2010) HHUIF and MSICF: novel algorithms for privacy preserving utility mining. Expert Syst Appl 37(7):4779–4786. https://doi.org/10.1016/j.eswa.2009.12.038
7. Lin JC-W, Wu T-Y, Fournier-Viger P, Lin G, Zhan J, Voznak M (2016) Fast algorithms for hiding sensitive high-utility itemsets in privacy-preserving utility mining. Eng Appl Artif Intell 55:269–284. https://doi.org/10.1016/j.engappai.2016.07.003
8. Jangra S, Toshniwal D (2022) Efficient algorithms for victim item selection in privacy-preserving utility mining. Futur Gener Comput Syst 128:219–234. https://doi.org/10.1016/j.future.2021.10.008
9. Ashraf M, Rady S, Abdelkader T, Gharib TF (2023) Efficient privacy preserving algorithms for hiding sensitive high utility itemsets. Computers & Security 132:103360. https://doi.org/10.1016/j.cose.2023.103360
10. Lin JC-W, Hong T-P, Fournier-Viger P, Liu Q, Wong J-W, Zhan J (2017) Efficient hiding of confidential high-utility itemsets with minimal side effects. Journal of Experimental & Theoretical Artificial Intelligence. 29(6):1225–1245. https://doi.org/10.1080/0952813X.2017.1328462
11. Li S, Mu N, Le J, Liao X (2019) A novel algorithm for privacy preserving utility mining based on integer linear programming. Eng Appl Artif Intell 81:300–312. https://doi.org/10.1016/j.engappai.2018.12.006
12. Nguyen D, Tran M-T, Le B (2023) A new algorithm using integer programming relaxation for privacy-preserving in utility mining. Appl Intell. https://doi.org/10.1007/s10489-023-04913-w
13. Karimi-Mamaghan M, Mohammadi M, Meyer P, Karimi-Mamaghan AM, Talbi E-G (2022) Machine learning at the service of meta-heuristics for solving combinatorial optimization problems: a state-of-the-art. Eur J Oper Res 296(2):393–422. https://doi.org/10.1016/j.ejor.2021.04.032
14. Bao X, Kang H, Li H (2024) An improved binary snake optimizer with gaussian mutation transfer function and hamming distance for feature selection. Neural Comput Appl 36(16):9567–9589. https://doi.org/10.1007/s00521-024-09581-6
15. Kovačević A, Luburić N, Slivka J, Prokić S, Grujić K-G, Vidaković D, Sladić G (2024) Automatic detection of code smells using metrics and codet5 embeddings: a case study in c#. Neural Comput Appl 36(16):9203–9220. https://doi.org/10.1007/s00521-024-09551-y
16. Mirjalili S (2015) The ant lion optimizer. Adv Eng Softw 83:80–98. https://doi.org/10.1016/j.advengsoft.2015.01.010
17. Braik MS (2021) Chameleon swarm algorithm: a bio-inspired optimizer for solving engineering design problems. Expert Syst Appl 174:114685. https://doi.org/10.1016/j.eswa.2021.114685
18. Dhiman G, Kumar V (2017) Spotted hyena optimizer: a novel bio-inspired based metaheuristic technique for engineering applications. Adv Eng Softw 114:48–70. https://doi.org/10.1016/j.advengsoft.2017.05.014
19. Hatamlou A (2013) Black hole: a new heuristic optimization approach for data clustering. Inf Sci 222:175–184. https://doi.org/10.1016/j.ins.2012.08.023 . Including Special Section on New Trends in Ambient Intelligence and Bio-inspired Systems
20. Ramezani F, Lotfi S (2013) Social-based algorithm (sba). Appl Soft Comput 13(5):2837–2856. https://doi.org/10.1016/j.asoc.2012.05.018
21. Mohammadi-Balani A, Dehghan Nayeri M, Azar A, Taghizadeh-Yazdi M (2021) Golden eagle optimizer: a nature-inspired metaheuristic algorithm. Computers & Industrial Engineering. 152:107050. https://doi.org/10.1016/j.cie.2020.107050
22. Mirjalili S, Gandomi AH, Mirjalili SZ, Saremi S, Faris H, Mirjalili SM (2017) Salp swarm algorithm: a bio-inspired optimizer for engineering design problems. Adv Eng Softw 114:163–191. https://doi.org/10.1016/j.advengsoft.2017.07.002
23. Kaur A, Jain S, Goel S (2019) Sandpiper optimization algorithm: a novel approach for solving real-life engineering problems. Appl Intell 50(2):582–619. https://doi.org/10.1007/s10489-019-01507-3
24. Afshari MH, Dehkordi MN, Akbari M (2016) Association rule hiding using cuckoo optimization algorithm. Expert Syst Appl 64:340–351. https://doi.org/10.1016/j.eswa.2016.08.005
25. Yao H, Hamilton HJ, Butz CJ (2004) A foundational approach to mining itemset utilities from databases. In: Proceedings of the 2004 SIAM international conference on data mining, pp 482–486. https://doi.org/10.1137/1.9781611972740.51
26. Liu Y, Liao W-k, Choudhary A (2005) A two-phase algorithm for fast discovery of high utility itemsets. In: Ho TB, Cheung D, Liu H (eds) Advances in knowledge discovery and data mining, pp 689–695. Springer, Berlin, Heidelberg. https://doi.org/10.1007/11430919_79
27. Lin C-W, Hong T-P, Lu W-H (2011) An effective tree structure for mining high utility itemsets. Expert Syst Appl 38(6):7419–7424. https://doi.org/10.1016/j.eswa.2010.12.082
28. Tseng VS, Wu C-W, Shie B-E, Yu PS (2010) Up-growth: an efficient algorithm for high utility itemset mining. In: Proceedings of the 16th ACM SIGKDD international conference on knowledge discovery and data mining. KDD '10, pp 253–262. Association for Computing Machinery, New York, USA. https://doi.org/10.1145/1835804.1835839
29. Tseng VS, Shie B, Wu C, Yu PS (2013) Efficient algorithms for mining high utility itemsets from transactional databases. IEEE Trans Knowl Data Eng 25(8):1772–1786. https://doi.org/10.1109/TKDE.2012.59
30. Liu M, Qu J (2012) Mining high utility itemsets without candidate generation. In: Proceedings of the 21st ACM international conference on information and knowledge management. CIKM '12, pp 55–64, New York, USA. https://doi.org/10.1145/2396761.2396773
31. Zida S, Fournier Viger P, Lin C-W, Wu C-W, Tseng V (2016) EFIM: a fast and memory efficient algorithm for high-utility itemset mining. Knowl Inf Syst 51:595–625. https://doi.org/10.1007/s10115-016-0986-0
32. Kim H, Yun U, Baek Y, Kim H, Nam H, Lin JC-W, Fournier-Viger P (2021) Damped sliding based utility oriented pattern mining over stream data. Knowl-Based Syst 213:106653. https://doi.org/10.1016/j.knosys.2020.106653
33. Gan W, Lin JC-W, Chao H-C, Fournier-Viger P, Wang X, Yu PS (2020) Utility-driven mining of trend information for intelligent system. ACM Trans Manag Inf Syst 11(3):1–28. https://doi.org/10.1145/3391251
34. Vo B, Nguyen LTT, Nguyen TDD, Fournier-Viger P, Yun U (2020) A multi-core approach to efficiently mining high-utility itemsets in dynamic profit databases. IEEE Access. 8:85890–85899. https://doi.org/10.1109/ACCESS.2020.2992729
35. Lin JC-W, Djenouri Y, Gautam S, Fourier-Viger P (2022) Efficient evolutionary computation model of closed high-utility itemset mining. Appl Intell 52(9):10604–10616. https://doi.org/10.1007/s10489-021-03134-3
36. Lin JC-W, Djenouri Y, Gautam S, Yun U, Fournier-Viger P (2021) A predictive ga-based model for closed high-utility itemset mining. Appl Soft Comput 108:107422. https://doi.org/10.1016/j.asoc.2021.107422
37. Yun U, Kim J (2015) A fast perturbation algorithm using tree structure for privacy preserving utility mining. Expert Syst Appl 42(3):1149–1165. https://doi.org/10.1016/j.eswa.2014.08.037
38. Liu X, Wen S, Zuo W (2020) Effective sanitization approaches to protect sensitive knowledge in high-utility itemset mining. Appl Intell 50(1):169–191. https://doi.org/10.1007/s10489-019-01524-2

39. Yin C, Li Y (2023) Fast privacy-preserving utility mining algorithm based on utility-list dictionary. Appl Intell 53(23):29363–29377. https://doi.org/10.1007/s10489-023-04791-2
40. Nguyen D, Le B (2022) A fast algorithm for privacy-preserving utility mining. Journal on Information Technologies & Communications. 2022(1):12–22. https://doi.org/10.32913/mic-ict-research.v2022.n1.1026
41. Lin C-W, Hong T-P, Wong J-W, Lan G-C, Lin W-Y (2014) A GA-based approach to hide sensitive high utility itemsets. Scientific World Journal 2014:2356–6140. https://doi.org/10.1155/2014/804629
42. Liu X, Chen G, Wen S, Song G (2020) An improved sanitization algorithm in privacy-preserving utility mining. Math Probl Eng 2020:1–14. https://doi.org/10.1155/2020/7489045
43. Hatjimihail AT (1993) Genetic algorithms-based design and optimization of statistical quality-control procedures. Clin Chem 39(9):1972–1978
44. Eberhart R, Kennedy J (1995) A new optimizer using particle swarm theory. In: MHS'95. Proceedings of the Sixth international symposium on micro machine and human science, pp 39–43. https://doi.org/10.1109/MHS.1995.494215 . IEEE
45. Colorni A, Dorigo M, Maniezzo V et al (1991) Distributed optimization by ant colonies. In: Proceedings of the first European conference on artificial life, vol 142, pp 134–142. Paris, France
46. Mirjalili S, Mirjalili SM, Lewis A (2014) Grey wolf optimizer. Adv Eng Softw 69:46–61. https://doi.org/10.1016/j.advengsoft.2013.12.007
47. Chou J-S, Nguyen N-M (2020) Fbi inspired meta-optimization. Appl. Soft Comput 93:106339. https://doi.org/10.1016/j.asoc.2020.106339
48. Gurobi Optimization L (2020) Gurobi Optimizer Reference Manual. http://www.gurobi.com
49. Zhang C, Almpanidis G, Wang W, Liu C (2018) An empirical evaluation of high utility itemset mining algorithms. Expert Syst Appl 101:91–115. https://doi.org/10.1016/j.eswa.2018.02.008
50. Liu J, Wang K, Fung BC (2012) Direct discovery of high utility itemsets without candidate generation. In: 2012 IEEE 12th international conference on data mining, pp 984–989. https://doi.org/10.1109/ICDM.2012.20. IEEE

Duc 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.

Bac Le 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.