

See discussions, stats, and author profiles for this publication at: <https://www.researchgate.net/publication/316945494>

Efficient hiding of confidential high-utility itemsets with minimal side effects

Article in Journal of Experimental & Theoretical Artificial Intelligence · May 2017

DOI: 10.1080/095213X.2017.1328462

CITATIONS

11

READS

119

6 authors, including:



Chun-Wei Jerry Lin

Høgskulen på Vestlandet

310 PUBLICATIONS 2,720 CITATIONS

[SEE PROFILE](#)



Tzung-Pei Hong

National University of Kaohsiung

688 PUBLICATIONS 7,827 CITATIONS

[SEE PROFILE](#)



Philippe Fournier Viger

Harbin Institute of Technology (Shenzhen)

258 PUBLICATIONS 3,230 CITATIONS

[SEE PROFILE](#)

Some of the authors of this publication are also working on these related projects:



Mining Utility Itemsets in SpatioTemporal Databases [View project](#)



Privacy Preserving Utility Mining [View project](#)



Efficient hiding of confidential high-utility itemsets with minimal side effects

Jerry Chun-Wei Lin, Tzung-Pei Hong, Philippe Fournier-Viger, Qiankun Liu, Jia-Wei Wong & Justin Zhan

To cite this article: Jerry Chun-Wei Lin, Tzung-Pei Hong, Philippe Fournier-Viger, Qiankun Liu, Jia-Wei Wong & Justin Zhan (2017): Efficient hiding of confidential high-utility itemsets with minimal side effects, Journal of Experimental & Theoretical Artificial Intelligence, DOI: [10.1080/0952813X.2017.1328462](https://doi.org/10.1080/0952813X.2017.1328462)

To link to this article: <http://dx.doi.org/10.1080/0952813X.2017.1328462>



Published online: 15 May 2017.



Submit your article to this journal [↗](#)



Article views: 9




View related articles [↗](#)



View Crossmark data [↗](#)



Efficient hiding of confidential high-utility itemsets with minimal side effects

Jerry Chun-Wei Lin^{a,b}, Tzung-Pei Hong^{c,d}, Philippe Fournier-Viger^e , Qiankun Liu^b,
Jia-Wei Wong^d and Justin Zhan^f

^aFujian Provincial Key Laboratory of Big Data Mining and Applications, Fujian University of Technology, Fujian, China;

^bSchool of Computer Science and Technology, Harbin Institute of Technology Shenzhen Graduate School, Shenzhen, China; ^cDepartment of Computer Science and Information Engineering, National University of Kaohsiung, Kaohsiung, Taiwan; ^dDepartment of Computer Science and Engineering, National Sun Yat-sen University, Kaohsiung, Taiwan; ^eSchool of Natural Sciences and Humanities, Harbin Institute of Technology Shenzhen Graduate School, Shenzhen, China; ^fDepartment of Computer Science, University of Nevada, Las Vegas, NV, USA

ABSTRACT

Privacy preserving data mining (PPDM) is an emerging research problem that has become critical in the last decades. PPDM consists of hiding sensitive information to ensure that it cannot be discovered by data mining algorithms. Several PPDM algorithms have been developed. Most of them are designed for hiding sensitive frequent itemsets or association rules. Hiding sensitive information in a database can have several side effects such as hiding other non-sensitive information and introducing redundant information. Finding the set of itemsets or transactions to be sanitised that minimises side effects is an NP-hard problem. In this paper, a genetic algorithm (GA) using transaction deletion is designed to hide sensitive high-utility itemsets for PPDM. A flexible fitness function with three adjustable weights is used to evaluate the goodness of each chromosome for hiding sensitive high-utility itemsets. To speed up the evolution process, the pre-large concept is adopted in the designed algorithm. It reduces the number of database scans required for verifying the goodness of an evaluated chromosome. Substantial experiments are conducted to compare the performance of the designed GA approach (with/without the pre-large concept), with a GA-based approach relying on transaction insertion and a non-evolutionary algorithm, in terms of execution time, side effects, database integrity and utility integrity. Results demonstrate that the proposed algorithm hides sensitive high-utility itemsets with fewer side effects than previous studies, while preserving high database and utility integrity.

ARTICLE HISTORY

Received 7 November 2015
Accepted 29 April 2017

KEYWORDS

Genetic algorithm;
transaction deletion;
high-utility mining; privacy
preserving data mining

1. Introduction

Frequent itemset mining (FIM), association rule mining (ARM), and sequential-pattern mining are fundamental techniques for knowledge discovery in databases (Agrawal & Srikant, 1994b; Fournier-Viger, Lin, Kiran, Koh, & Thomas, 2017; Han, Pei, Yin, & Mao, 2004; Lin, Hong, & Lu, 2009). Although these techniques are useful, an important issue is that information discovered by data mining techniques may reveal confidential or private information such as social security numbers, credit card numbers and health issues.

In recent decades, privacy-preserving data mining (PPDM) (Agrawal & Srikant, 2000; Atallah, Elmagarmid, Ibrahim, Bertino, & Verykios, 1999; Verykios et al., 2004) has attracted the attention of numerous researchers and practitioners as it allows to sanitise a database by hiding confidential and private information, while ensuring that other important information may still be extracted for decision-making. Agrawal and Srikant (2000) have shown that randomised Gaussian and Uniform perturbation functions can be applied to anonymise a database. Evfimievski, Srikant, Agrawal, and Gehrke (2002) presented a framework for mining association rules in transactions consisting of categorical items where the data are randomised to preserve the privacy of individual transactions. Clifton, Kantarcioglu, Vaidya, Lin, and Zhu (2002) designed a toolkit with several components to solve the problems of PPDM. Lindell and Pinkas (2000) introduced the concept of PPDM and used the ID3 principle in PPDM. Wu, Chiang, and Chen (2007) designed an algorithm to modify transactions to, respectively, decrease the support and confidence of sensitive rules for PPDM. Hong, Lin, Yang, and Wang (2013) designed a sensitive items frequency-inverse database frequency approach to sanitise a database by decreasing the support of sensitive itemsets. Lin, Hong, Chang, and Wang (2013) designed a greedy approach to increase the size of a database to hide sensitive information.

High-utility itemset mining (HUIM) (Lin, Hong, & Lu, 2011; Liu, Liao, & Choudhary, 2005; Yao, Hamilton, & Butz, 2004; Yao & Hamilton, 2006) is an emerging research topic which considers both the unit profits of items and their purchase quantities in transactions to mine high-utility itemsets (HUIs). Because HUIM suffers from the same security problems as traditional ARM, privacy-preserving utility mining (PPUM) (Rajalaxmi & Nataraja, 2012; Yeh & Hsu, 2010; Yun & Kim, 2015) has emerged as an important variation of PPDM. In PPUM, sensitive high-utility patterns are hidden by perturbing the original database by removing itemsets or reducing the utilities of sensitive itemsets in the database. Yeh and Hsu (2010) designed the state-of-the-art HHUIF and MSICF approaches for hiding sensitive high-utility patterns. Rajalaxmi and Nataraja (2012) presented two sanitisation algorithms named MSMU and MCRSU to hide both sensitive high-utility itemsets and frequent itemsets by modifying databases. Yun and Kim (2015) presented an efficient tree structure to hide sensitive high-utility patterns.

Since the purpose of PPDM and PPUM is to hide sensitive information in a database while ensuring that other important information or rules can still be revealed, these problems can be viewed as optimisation problems where an optimal solution must be found, which is NP-hard (Agrawal & Srikant, 2000; Aggarwal, Pei, & Zhang, 2006). A genetic algorithm (GA) Holland (1992) is an evolutionary model that can find nearly optimal feasible solutions in a limited amount of time. In this paper, a GA-based approach is developed to sanitise a database for hiding sensitive high-utility itemsets through transaction deletion. The key contributions of the designed algorithm are twofold and summarised as follows.

- (1) This is the first paper that addresses the problem of privacy-preserving HUIM by adopting a GA-based approach to find appropriate transactions to be deleted for hiding sensitive high-utility itemsets. A flexible fitness function is adopted in the evolution process to evaluate the goodness of each processed chromosome, and minimise side effects resulting from the sanitisation process.
- (2) The main drawback of evolutionary computation is that it is time-consuming. To address this issue, this paper proposes an enhanced pre-large concept, to avoid performing multiple database scans in the evolution process for quickly revealing the side effects of each processed chromosome. This procedure greatly reduces the runtime of the evolution process.

2. Related work

In this section, related work on HUIM and PPDM are reviewed and discussed.

2.1. High-utility itemset mining

HUIM (Chan, Yang, & Shen, 2003; Yao et al., 2004; Yao & Hamilton, 2006) is an emerging topic that has become increasingly important in recent years, since it can reveal more profitable and meaningful information than traditional ARM. HUIs can be used by managers and decision-makers to take informed business decisions and adapt sales strategies. Yao et al. (2004) and Yao and Hamilton (2006) developed algorithms to mine profitable itemsets by considering both purchase quantities and unit profits of items to reveal the desired HUIs. Because the *downward closure* property does not hold in traditional HUIM, the *transaction-weighted utilisation* model (Liu et al., 2005) with its designed *transaction-weighted downward closure* property was proposed for finding the *high transaction-weighted utilisation itemsets* (HTWUIs), and speed up the discovery of HUIs. This approach performs a level-wise search to find HUIs.

To avoid the drawbacks of level-wise approaches, Lin et al. designed the high-utility-pattern (HUP)-tree structure to mine HUIs (Lin et al., 2011). The HUP-tree algorithm first discovers the 1-HTWUIs (HTWUIs containing a single item) using the TWU model. Then, the discovered 1-HTWUIs are used to construct the HUP-tree structure, which is then used to discover all HUIs. A list-based algorithm named HUI-Miner (Liu & Qu, 2012) was also designed to directly mine HUIs without candidate generation. Several extensions of HUIM are, respectively, presented and discussed (Lin, Gan, Fournier-Viger, Hong, & Tseng, 2015; Lin, Gan, Fournier-Viger, Hong, & Zhan, 2016; Lin, Fournier-Viger, & Gan, 2016; Liu, Wang, & Fung, 2016).

2.2. Privacy-preserving techniques

Data mining techniques (Agrawal & Srikant, 1994b; Han et al., 2004) are used to discover and analyse implicit relationships between purchased items. Private or confidential information between items in itemsets can also be revealed by data mining techniques, which may result in security threats and information theft. PPDM (Agrawal & Srikant, 2000; Aggarwal et al., 2006; Verykios et al., 2004) has thus emerged as an important topic in recent years since private or confidential information can be hidden, while ensuring that the desired information about the purchase of itemsets can still be discovered. Sanitisation is a technique of PPDM, which can hide private or confidential information successfully using deletions or insertions. To compare the performance of sanitisation algorithms, three side effects named hiding failure, missing cost and artificial cost, have been proposed (Oliveria & Zaiane, 2002; Wu et al., 2007). Atallah et al. (1999) developed a protection mechanism to sanitise a database for hiding sensitive association rules. Wu et al. (2007) designed several heuristic approaches for hiding sensitive rules while minimising the number of modified entries. Giannotti, Lakshmanan, Monreale, Pedreschi, and Wang (2013) studied the problem of outsourcing the ARM task within a corporate privacy-preserving framework. Bonam, Reddy, and Kalyani (2014) addressed the issue of privacy preservation in ARM by developing a PSO-based approach using data distortion. Cheng, Lin, and Pan (2015) applied the multi-objective optimisation (EMO) mechanism to consider multiple factors for hiding sensitive itemsets. Cheng, Roddick, Chu, and Lin (2016) then proposed a deletion method to reduce the support or confidence of sensitive rules below specified thresholds for PPDM.

Besides PPDM, PPUM has also become a critical issue since it considers both the purchase quantities and unit profits of item to hide sensitive HUIs. Yeh and Hsu (2010) first designed two algorithms named HHUIF and MSICF for hiding sensitive high-utility itemsets. Lin, Wu et al. (2015) designed three similarity measurements to be used as a new standard for evaluating side effects in PPUM. Rajalaxmi and Nataraja (2012) presented the MSMU and MCRSU algorithms to hide both sensitive utility and frequent item/sets. Yun and Kim (2015) designed a tree structure and a FPUTT algorithm to speed up the HHUIF and MSICF algorithms. It is a NP-hard problem and a non-trivial task to find appropriate transactions or items to be modified for hiding sensitive HUIs in PPUM. Lin et al. (2014) first presented a GA-based approach for hiding sensitive HUIs by inserting dummy transactions to increase the total utility in the database. However, this approach results in a high artificial cost since the size of transactions increases and many non-HUIs become HUIs after the sanitisation process.

Table 1. A quantitative database.

TID	Items and their quantities
1	D:6, F:1
2	E:6
3	A:5, E:1
4	B:5, F:2
5	C:8, F:2
6	A:4, E:1
7	B:2, C:3, D:2
8	A:7, B:3, E:2
9	E:4
10	A:5, B:2, E:5
11	C:1, F:1
12	A:3, E:3

3. Preliminaries and problem statement

Let $I = \{i_1, i_2, \dots, i_m\}$ be a finite set of m distinct items. A quantitative database is a set of transactions $D = \{T_1, T_2, \dots, T_n\}$, where each transaction $T_q \in D$ ($1 \leq q \leq n$) is a subset of I and has a unique identifier q , called its *TID*. Besides, each item i_j in a transaction T_q has a purchase quantity (internal utility) denoted as $q(i_j, T_q)$. A profit table $ptable = \{pr_1, pr_2, \dots, pr_m\}$ indicates the profit value (unit profit) pr_j of each item i_j . A set of k distinct items $X = \{i_1, i_2, \dots, i_k\}$ such that $X \subseteq I$ is said to be a k -itemset, where k is the length of the itemset. An itemset X is said to be contained in a transaction T_q if $X \subseteq T_q$. A minimum utility threshold (MUT) δ is set by the user according to his preference.

An illustrated example is presented in Table 1. It will be used as running example for the rest of this paper. It contains 12 transactions and 5 distinct items, represented by letters from (A) to (F). The profit value (external utility) of each item is set as $ptable = \{A:7, B:15, C:10, D:6, E:2, F:1\}$. The MUT is set to ($\delta = 20\%$).

Definition 1: The utility of an item i_j in a transaction T_q is denoted as $u(i_j, T_q)$, and defined as:

$$u(i_j, T_q) = q(i_j, T_q) \times pr(i_j). \quad (1)$$

For example, the utility of item (A) in transaction T_3 is calculated as: $u(A, T_3) = 5 \times 7 (= 35)$.

Definition 2: The utility of an itemset X in a transaction T_q is denoted as $u(X, T_q)$, and defined as:

$$u(X, T_q) = \sum_{i_j \in X \wedge X \subseteq T_q} u(i_j, T_q). \quad (2)$$

For example, the utility of the itemset (AE) in transaction T_3 is calculated as: $u(AE, T_3) = 35 + 2 (= 37)$

Definition 3: The utility of an itemset X in a database D is denoted as $u(X)$, and defined as:

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

For example, the utility of itemset (AE) in D is calculated as: $u(AE) = 37 + 30 + 53 + 45 + 27 (= 192)$.

Definition 4: The transaction utility of a transaction T_q is denoted as $tu(T_q)$, and defined as:

$$tu(T_q) = \sum_{X \in T_q} u(X, T_q). \quad (4)$$

For example, the $tu(T_1)$ is calculated as: $tu(T_1) = 36 + 1 (= 37)$.

Definition 5: The transaction-weighted utilisation of an itemset X is denoted as $TWU(X)$, and defined as:

$$TWU(X) = \sum_{X \subseteq T_q \wedge T_q \in D} tu(T_q). \quad (5)$$

For example, $TWU(AE)$ is calculated as: $TWU(AE) = 37 + 30 + 98 + 75 + 27 (= 267)$.

Definition 6: The total utility of a database D is denoted as TU , and defined as:

$$TU = \sum_{T_q \in D} tu(T_q). \quad (6)$$

For example, the total utility of a database D is calculated as: $TU = 37 + 12 + 37 + 77 + 82 + 30 + 72 + 98 + 8 + 75 + 11 + 27 (= 566)$.

Definition 7: An itemset X is a HUI in a database D iff its utility is no less than the minimum utility count defined as $TU \times \delta$. The set of HUIs is thus defined as:

$$HUIs \leftarrow \{X | u(X) \geq TU \times \delta\}. \quad (7)$$

Assume that the MUT is set to $\delta (= 20\%)$. In this example, the discovered HUIs and their utilities are $\{A:168, B:180, C:120, AB:159, AE:192, ABE:173\}$.

Definition 8: Let $HS = \{s_1, s_2, \dots, s_k\}$ denotes the set of sensitive HUIs to be hidden in a database D .

For example, suppose that two itemsets (B) and (ABE) are considered to be sensitive HUIs that need to be hidden. Thus, $HS = \{B, ABE\}$.

In traditional PPDM (Wu et al., 2007), three standard side effects are considered to evaluate the performance of a sanitisation method, namely hiding failures (HF), missing cost (MC) and artificial cost (AC). HF is the set of sensitive itemsets that the sanitisation process failed to hide. If the number of itemsets in HF is large, it means that the sanitisation process cannot successfully hide the sensitive information. If the number of itemsets in HF is zero, it indicates that all sensitive information has been hidden. MC is the set of itemsets that were non-sensitive before the sanitisation process but have been hidden by that process. If the number of itemsets in MC is high, it means that many itemsets that were not sensitive but were considered important are hidden or missing. Thus, it leads to a loss of information, which may result in taking incorrect decisions or using inefficient sale strategies. AC is the set of itemsets that would not have been discovered by the data mining process before sanitisation but are discovered after the sanitisation process. If the number of itemsets in AC is high, it indicates that many redundant or unnecessary itemsets are discovered and considered as important when data mining techniques are applied to the sanitised database. For PPUM (Yeh & Hsu, 2010), criteria similar to PPDM (Wu et al., 2007) are used, which are defined as follows.

Definition 9: Let $\alpha (= HF)$ be the sensitive HUIs that the sanitisation process failed to hide, that is the number of sensitive HUIs that still appear in the database after the sanitisation process. Formally, it is defined as:

$$\alpha = HS \cap HUIs', \quad (8)$$

where HS is the set of sensitive HUIs before the sanitisation process and $HUIs'$ is the set of HUIs after the sanitisation process.

Definition 10: Let $\beta (= MC)$ be the missing HUIs, i.e. the HUIs that are non-sensitive but that the sanitisation process has hidden:

$$\beta = \sim HS - HUIs'. \quad (9)$$

Definition 11: Let $\gamma (= AC)$ be the itemsets that were not HUIs before the sanitisation process but are now HUIs, that is the difference between $HUIs'$ and $HUIs$:

$$\gamma = HUIs' - HUIs, \quad (10)$$

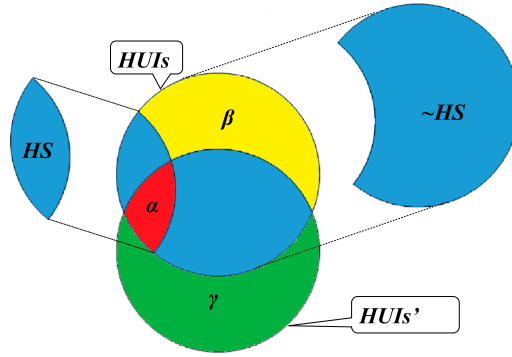


Figure 1. The relationships between the three side effects and the discovered HUIs before and after sanitisation.

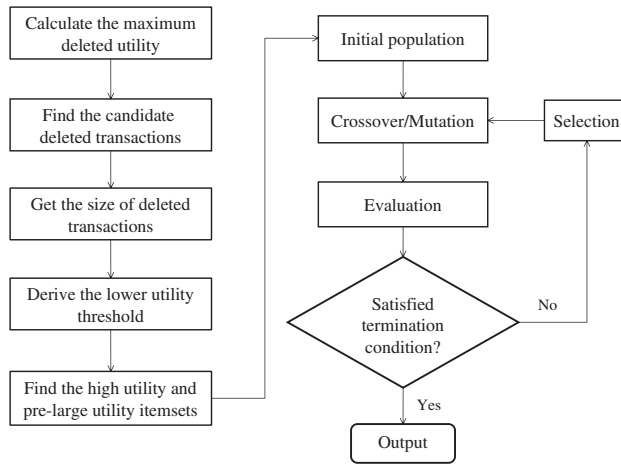


Figure 2. The flowchart of the designed PPUMGAT algorithm.

where $HUIs'$ is the set of HUIs obtained after the sanitisation process.

The relationships between these three side effects, and the discovered high-utility itemsets before ($HUIs$) and after sanitisation ($HUIs'$) are illustrated in Figure 1.

Problem: Given a set of sensitive HUIs, the problem of PPUM (Yeh & Hsu, 2010) using a GA (Holland, 1992) and transaction deletion is to find an appropriate set of transactions to be deleted as an optimal solution for hiding as many sensitive HUIs as possible, while minimising the three side effects (HF , MC and AC).

4. Proposed GA-based algorithm of PPUM

Finding an optimal solution to the problem of PPUM is an NP-hard problem when considering HF , MC and AC . Traditional approaches have been designed to hide sensitive HUIs by reducing the purchase quantities of itemsets, and as a result reducing the utilities of sensitive itemsets. This approach may be efficient but can produce many misleading rules or information since it has no mechanism to select which transactions should be modified to minimise side effects. This paper thus presents a GA-based algorithm (Holland, 1992) to sanitise a database for hiding sensitive HUIs by finding appropriate transactions for deletion. A GA-based algorithm for PPUM using transaction deletion named PPUMGAT

is thus designed to hide sensitive HUIs with minimal side effects. Each chromosome consists of several genes and each gene represents a transaction to be deleted by the designed approach. The flowchart of the proposed algorithm is shown in Figure 2.

4.1. Fitness function

In the designed PPUMGAT algorithm, a flexible fitness function is given to evaluate the goodness of each processed chromosome. In the developed fitness function, three weights are respectively, assigned to the three side effects, according to the user's preferences. The number of instances of each side effect (HF , MC and AC) is considered in the designed fitness function, used by the evolution process.

Definition 12: The proposed fitness function to evaluate the goodness of a processed chromosome in PPUM is defined as:

$$fitness(c_i) = |\alpha| \times w_1 + |\beta| \times w_2 + |\gamma| \times w_3, \quad (11)$$

where w_1 , w_2 and w_3 are three adjustable weights defined according to the user's preference.

As stated in the problem statement, the purpose of PPUM as defined in this paper is to find appropriate transactions to be deleted for hiding the sensitive HUIs, while minimising the three side effects as a result of the evolution process.

4.2. Proposed PPUMGAT algorithm

The full pseudo-code of the designed PPUMGAT algorithm is presented in Algorithm 1.

Algorithm 1: Proposed PPUMGAT algorithm

Input: D , a quantitative database; $ptable$, a profit table, S_U , the upper (minimum) utility threshold; M , the number of chromosomes in a population; N , the number of iterations in the evolution process.

Output: D' , a sanitised database.

```

1 for each  $s_j \in HS$  do
2    $MDU \leftarrow MDU + TWU(s_j) - TU \times S_U$ ;
3 for each  $T_q \in D$  do
4   calculate  $tu(T_q)$ ;
5   for each  $s_j \in HS$  do
6     if  $s_j \in T_q$  and  $tu(T_q) < MDU$  then
7        $Candi\_Delete \leftarrow T_q$ ;
8 sort the transactions in  $Candi\_Delete$  in ascending order of  $tu$ ;
9 set  $m = 0$ ,  $sum = 0$ ;
10 for each  $T_q$  in  $Candi\_Delete$  do
11   if  $sum < MDU$  then
12      $sum \leftarrow sum + tu(T_q)$ ;
13      $m \leftarrow m + 1$ ;
14 set the size of a chromosome to  $m$ ;
15 randomly generate  $M$  chromosomes as the initial population;
16 while termination criteria is not reached do
17   for each chromosome  $c_i$  among  $M$  chromosomes in a population do
18     perform crossover operation;
19     perform mutation operation;
20     evaluate  $fitness(c_i)$ ;
21     select top  $M/2$  chromosomes in a population;
22     randomly generate  $M/2$  chromosomes as the next generation;
23 obtain the optimal chromosome  $c_i$  with minimal fitness value from  $M$ ;
24 delete  $T_q$  of  $c_i$  from  $D$  as  $D'$ ;
25 return  $D'$ ;

```

First, the Maximum Deleted Utility is calculated (MDU), which is defined as the sum of the difference between the TWU of each sensitive itemset and the minimum utility count (Lines 1 to 2).

Definition 13: The maximal deleted utility is denoted as MDU and defined as:

$$MDU = \sum_{\forall s_i \in HS} (TWU(s_i) - TU \times \delta). \quad (12)$$

In the designed algorithm, transactions are selected to be deleted for hiding the sensitive HUIs. In this transaction deletion process, each transaction is evaluated and projected as a candidate transaction for deletion in *Candi_Delete* if it contains at least a sensitive HUI. Besides, the maximal deleted utility value is calculated from the predefined sensitive HUIs. To minimise the MC side effect, it is desirable to delete transactions having transaction utilities no less than the MDU (Lines 3–7). This process can help finding an optimal solution in terms of transactions to be deleted to minimise the AC side effect.

Definition 14: Each transaction containing at least a sensitive high-utility itemset in HS and having a transaction utility no less than MDU is projected as a candidate deleted transaction:

$$Candi_Delete \leftarrow \{T_q | T_q \in D, \exists s_i \in HS, s_i \subseteq T_q \wedge tu(T_q) < MDU\}. \quad (13)$$

Transactions in *Candid_Delete* are sorted in ascending order of their tu (Line 8). Transaction utilities in *Candid_Delete* are then summed until the value is greater than MDU . The number of transactions that has been summed is then used as the chromosome length for the evolution process (Lines 9–14). After that, a set of chromosomes is generated as the initial population, where the genes of each chromosome are randomly selected from *Candi_Delete* (Line 15). In the proposed PPUMGAT algorithm, a chromosome represents a possible solution, that is a set of transactions to be deleted for hiding the sensitive HUIs. Each gene of a chromosome represents the ID of a transaction in *Candi_Delete*. Note that it is a NP-hard problem to find an optimal solution for PPUM from the generated chromosomes in a population, and a chromosome gene is allowed to take the **null** value. Besides, a transaction can be repeatedly selected in the same chromosome but it can only be deleted once by the evolution process.

The crossover and mutation operations are also performed to update the chromosomes for the next iteration of the evolution process (Lines 18 to 19). After that, each chromosome is evaluated using the designed fitness function (Line 20). Half of the chromosomes, having the lowest fitness function values, are kept for the next generation, and another half is randomly generated (Lines 21 to 22). This procedure is then repeatedly performed until the termination criterion (i.e. the given maximum number of iterations in GA) is met (Lines 16–22). After that, the chromosome with the lowest fitness function value (Line 23) is projected and the transactions IDs in this chromosome are selected as the transactions for deletion, thus successfully hiding the sensitive HUIs (Line 24). Finally, the database is updated, and the sanitisation process is completed (Line 24).

4.3. Pre-large concept

In the evolution process, evaluating the goodness of chromosomes at each iteration is very time-consuming. This process requires to repeatedly scan the original database to evaluate the three side effects for each chromosome to calculate the designed fitness function, and especially to evaluate the AC side effect. To speed up this evaluation process, the pre-large concept is adopted in the designed algorithm. Thanks to this concept, multiple database scans can be avoided at each evaluation. The pre-large concept was proposed by [Hong and Wang \(2006\)](#) for efficiently maintaining discovered information in dynamic situations. It uses two thresholds called the upper (S_u) and lower (S_l) support thresholds to maintain a buffer of promising itemsets having a support in the $[S_u, S_l]$ interval, which have a high probability of being frequent itemsets. When transactions are deleted, the AC side effect

Table 2. Candidate deleted transactions (*Candi_Delete*) and their transaction utilities.

TID	Items and their quantities	<i>tu</i>
4	<i>B</i> :5, <i>F</i> :2	77
7	<i>B</i> :2, <i>C</i> :3, <i>D</i> :2	72
8	<i>A</i> :7, <i>B</i> :3, <i>E</i> :2	98
10	<i>A</i> :5, <i>B</i> :2, <i>E</i> :5	75

can be directly calculated using the pre-large buffer and thus multiple database scans can be avoided. This strategy is simple but can be used to efficiently maintain updated information.

In the designed PPUMGAT algorithm, the maximal deleted utility (*MDU*) is first calculated using the sensitive HUIs. The *MDU* can be considered as the safety bound to be used by the pre-large concept to avoid multiple database scans at each iteration and the total utility (*TU*) in the database can be considered as the database size in the original database. Based on the pre-large concept, we can calculate the S_I value to keep the set of promising pre-large utility itemsets (PUIs) in a buffer to avoid performing database scans.

Definition 15: Let S_u be the user-defined upper support threshold, and *MDU* be the maximal deleted utility obtained from the sensitive HUIs, and *TU* be the total utility in the original database. The S_I value of the pre-large concept is redefined in the proposed approach as:

$$S_I = S_u \times \left(1 - \frac{MDU}{TU}\right). \quad (14)$$

Thanks to the modified pre-large concept, it is possible to calculate the side effects without rescanning the database. After that, the generated chromosomes are then evaluated by the designed fitness function. This procedure is repeated until the termination criterion (i.e. a maximum of 100 iterations) is met. For each generation, half of the chromosomes, having the lowest fitness values, are selected as the remaining chromosomes for the next generation. The other half of the chromosomes is generated using the transactions in the projected database *Candi_Delete*. Finally, the chromosome having the lowest fitness value is kept and the transaction *IDs* within that chromosome are deleted from the original database for sanitisation. The final sanitised database is then output as the final updated database.

5. An illustrated example

An example is given to illustrate the proposed algorithm step-by-step. Consider the database of Table 1, and assume that (*B*) and (*ABE*) are sensitive high-utility itemsets that need to be hidden. Therefore, the *MDU* of Table 1 is calculated as: $MDU = (TWU(B) - 113) + (TWU(ABE) - 113) (= 269)$. After that, the projected transactions of *Candi_Delete* from Table 1 are shown in Table 2.

In the running example, $MDU (= 269)$, and $tu(T_7) + tu(T_{10}) + tu(T_4) (= 224) < 269$, and $tu(T_7) + tu(T_{10}) + tu(T_4) + tu(T_8) (= 322) > 269$. Thus, the number of summed transactions is 3, which is used as the chromosome length (number of genes) for the evolution process.

From the above example, the chromosome length was set to 3, which indicates that at most three transactions can be deleted for hiding the sensitive HUIs. The number of chromosomes in a population is set to 5. Thus, the generated chromosomes in a population are shown in Figure 3. For the first chromosome in Figure 3, the transactions T_4 and T_7 are the transactions to be deleted by the proposed algorithm.

Suppose that the upper support threshold is set to 20% by the user, and that the maximal deleted utility is calculated as $MDU (= 113)$ using the predefined sensitive HUIs. The lower utility threshold is

c_1	4	7	<i>null</i>
c_2	8	<i>null</i>	7
c_3	4	8	10
c_4	10	<i>null</i>	8
c_5	4	10	8

Figure 3. The generated chromosomes in a population.

calculated as:

$$S_l = S_u \times \left(1 - \frac{MDU}{TU}\right) = 0.2 \times \left(1 - \frac{269}{566}\right) = 10.4\%.$$

Thus, each itemset having a TWU between $566 \times 0.2 (= 113.2)$ and $566 \times 0.104 (= 58.86)$ by the process of HUIM, are added to the buffer of PUIs. In this example, the discovered HUIs and PUIs are respectively as: HUIs = {A:168, B:180, C:120, AB:159, AE:192, ABE:173} and PUIs = {BE:89, CF:93}. Based on this improved pre-large concept, multiple database scans can be avoided for evaluating the side effects, and especially the AC.

Consider the database of the running example, and assume that one of the chromosome in Figure 3 is updated as (4, 7, *null*), which indicates that transactions T_4 and T_7 would be deleted by the sanitisation process. Thanks to the modified pre-large concept, it is possible to calculate the side effects without rescanning the database. The discovered HUIs and PUIs are then, respectively, updated.

In the example, the total transaction utility is updated to $(566 - 77 - 72) (= 412)$, and that the upper utility count is updated to $(412 \times 0.2) (= 82.4)$. In this example, the sensitive HUI (B) is thus hidden since its utility is updated to $(180 - 75 - 30) (= 75)$, which is lower than 82.4. The itemset (BE) becomes a HUI since its utility is now larger than $(89 > 82.4)$. After that, the chromosome is evaluated by the designed fitness function.

In this example, the *HF* side effect is calculated as: $\{B, ABE\} - \{B, ABE\} = \{ABE\}$, which is $(|\alpha| = 1)$. The *MC* side effect is calculated as: $\{A, B, C, AB, AE, ABE\} - \{A, C, AB, AE, BE, CF, ABE\} = \{B\}$, and thus $(|\beta| = 1)$. The *AC* side effect is calculated as: $\{A, C, AB, AE, BE, CF, ABE\} - \{A, B, C, AB, AE, ABE\} = \{BE, CF\}$, and thus $(|\gamma| = 2)$. Note that for the new HUIs (BE and CF), it is unnecessary to rescan the database to calculate its utility since this itemset was buffered in the set of PUIs. Thus, computations can be avoided. Assume that the three weights of the fitness function are, respectively, set to 0.5, 0.25 and 0.25. The goodness of the first generated chromosome in the population is then evaluated by the designed fitness function as:

$$fitness(4, 7, \mathbf{null}) = 0.5 \times 1 + 0.25 \times 1 + 0.25 \times 2 = 1.25.$$

The other chromosomes are then evaluated by the same way to find the optimal transactions to be deleted for hiding the sensitive HUIs. The above procedures are then repeated until the termination criterion, and the sanitised database is then output as the final updated database.

6. Experimental results

In the experiments, three evolutionary algorithms such as **PPUMGA+insert** (Lin et al., 2014) (an evolutionary sanitisation algorithm using transaction insertion), **PPUMGAT+** (the PPUMGAT algorithms with the pre-large concept) and **PPUMGAT** (the PPUMGAT algorithm without the pre-large concept)

Table 3. Characteristics of the data-sets.

Dataset	# D	# I	AvgLen	MaxLen	Type
Chess	3196	75	35	35	dense
Mushroom	8124	120	23	23	dense
Accidents	340,183	468	33.8	42	dense
Foodmart	21,556	1559	4	11	sparse
Retail	88,162	16,470	10	176	sparse
T10I4D100K	100,000	870	10.1	29	sparse

and **the non-evolutionary HHUIF (Yeh & Hsu, 2010) algorithm** are compared in terms of runtime, three side effects, database integrity and utility integrity. The difference between PPUMGAT+ and PPUMGAT- is only the adoption of the pre-large concept. The pre-large concept can reduce the runtime of the sanitisation process, but has no effect on the side effects of database integrity and utility integrity. Therefore, the results in terms of side effects, database integrity and utility integrity are not shown for PPUMGAT-. Five real-world data-sets (Fournier-Viger et al., 2016) and a synthetic database (Agrawal and Srikant, 1994a) generated by the IBM database generator were used in the experiments to show the performance of the designed approaches. Characteristics of the data-sets used in the experiments are shown in Table 3. The parameters used in Table 3 are, respectively, #|D|: total number of transactions; #|I|: number of distinct items; **AvgLen**: average transaction length; **MaxLen**: maximal transaction length; and **Type**: data-set type. In the experiments, the MUT and the percentage of sensitive HUIs (SP) are varied to evaluate the performance of the compared algorithms.

For the designed approach, the population size of the GA is set to 20 and the maximum number of iterations is set to 100. Single-point crossover is adopted and the mutation rate is set to 0.1. For the three dense data-sets (chess, mushroom and accidents), the weights of the three side effects in the fitness function are, respectively, set to 0.98, 0.01 and 0.01. For the three sparse data-sets (foodmart, retail and T10I4D100K), the weights of the three side effects are set to 0.8, 0.1 and 0.1. To evaluate how significant are the differences between results obtained by the compared evolutionary algorithms, a two-way ANOVA analysis was done. ANOVA is a commonly used method for statistical analysis. The two-way ANOVA determines how a response is affected by two factors (the x and y axis). The *F* value is the ratio of the mean-square value for that source of variation to the residual mean square. If the *F* value is large, it indicates that the difference between the compared algorithms is large. If the *F* value is close to 1, it indicates that the difference between the compared algorithms is small. The *P* value indicates if the difference between the compared algorithms is significant or not. If the *P* value is less than 0.05, it indicates that there is a significant difference between the compared algorithms. If the *P* value is greater than 0.05, it means that there is no significant difference between the compared algorithms. Since HHUIF is a non-evolutionary algorithm, it is not suitable to analyse results of this algorithm using a two-way ANOVA. Since the designed algorithm is an evolutionary algorithm, the two-way ANOVA analysis is only used to compare evolutionary algorithms.

6.1. Runtime

This subsection compares the runtime of the algorithms for all six data-sets. The runtime results of the algorithms w.r.t. various MUTs and a fixed percentage of sensitive HUIs are shown in Figure 4.

From Figure 4, it can be observed that the proposed PPUMGAT+ algorithm runs faster than PPUMGAT-. The reason is that PPUMGAT+ can avoid performing multiple unnecessary database scans to calculate the three side effects for evaluating chromosomes during the evolution process. But the proposed PPUMGAT+ algorithm requires more runtime than PPUMGA+insert for the three dense data-sets. The reason is that the PPUMGA+insert algorithm also adopts both the pre-large concepts and a GA approach to find the optimal solutions for adding dummy transactions to hide sensitive HUIs.

The maximum utility for insertion in PPUMGA+insert depends on the ratio between the maximum sensitive itemset utility and the MUT. For dense data-sets, the average transaction utility is higher

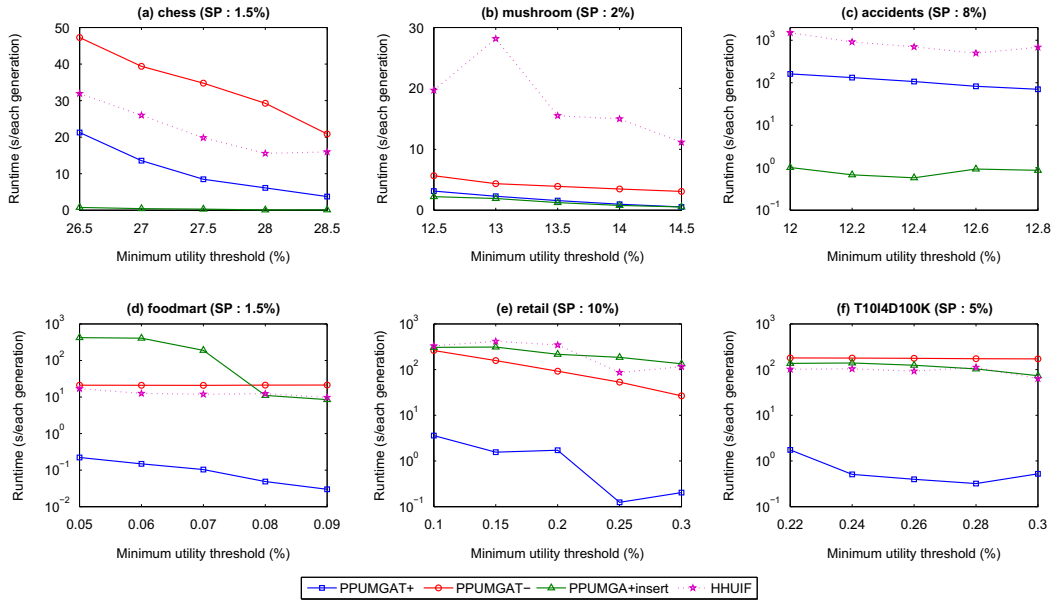


Figure 4. Runtime w.r.t. various MUTs.

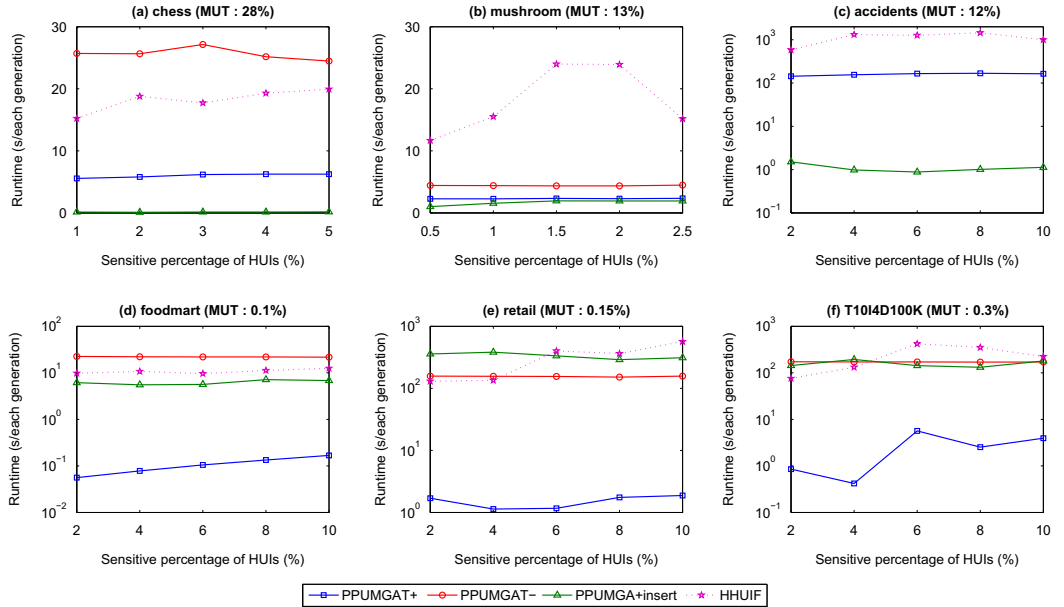


Figure 5. Runtime w.r.t. various SPs.

than for sparse data-sets. The PPUMGA+insert algorithm only adds a small number of transactions for hiding the sensitive HUIs. Thus, the runtime of the PPUMGA+insert algorithm is less than the PPUMGAT+ algorithm. Based on a two-way ANOVA analysis, there is a significant difference between the runtime of the PPUMGAT+ and the PPUMGA+insert algorithms ($F = 58.901$, $P < 0.001$, in Figure 4(a); $F = 216.684$, $P < 0.01$, in Figure 4(b); $F = 43.823$, $P = 0.003 < 0.05$, in Figure 4(c)). There

is also a significant difference between the runtimes of the PPUMGAT+ and PPUMGAT- algorithms for the chess data-set ($T = 7.376$, $P < 0.001$, in Figure 4(a)). For the accidents data-set, results are shown in Figure 4(c). Since PPUMGAT- exceeds 10^4 s on that data-set, it is ignored for the purpose of comparisons.

For sparse data-sets, results are shown in Figure 4(d)–(f). The designed PPUMGA+ is up to two order of magnitude faster than PPUMGA+insert. Based on a two-way ANOVA analysis, there is a significant difference between the runtimes of the PPUMGA+ and PPUMGA+insert algorithms ($F = 4.760$, $P = 0.043 < 0.05$, in Figure 4(d); $F = 24.408$, $P < 0.001$, in Figure 4(e); $F = 178.373$, $P < 0.001$, in Figure 4(f)). For the foormart data-set, there is no significant difference between the PPUMGAT- and PPUMGAT+insert algorithms ($P = 0.108$). However, there is a slightly significant difference between the PPUMGAT- and the PPUMGAT+insert algorithms ($P = 0.027 < 0.05$). For all experiments in Figure 4, the runtime of PPUMGAT+ decreases as the MUT is increased. These results are reasonable since when the MUT is increased with a fixed SP, the number of sensitive HUIs is decreased. As a result, it takes less time for finding the optimal solutions for sanitisation. Results of experiments under varied SPs with a fixed MUT for the six data-sets are shown in Figure 5.

It can be observed in Figure 5, that the designed PPUMGAT+ algorithm runs faster than the PPUMGAT- algorithm for all data-sets and is faster than PPUMGA+insert for sparse data-sets ($F = 3018.041$, $P < 0.001$, in Figure 5(d); $F = 318.946$, $P < 0.001$, in Figure 5(e); $F = 178.237$, $P < 0.001$, in Figure 5(f)). As for the results of Figure 4, the runtime of PPUMGAT+ is greater than for PPUMGA+insert for dense data-sets ($F = 2423.296$, $P < 0.001$, in Figure 5(a); $F = 194.929$, $P < 0.001$, in Figure 5(b); $F = 1188.355$, $P < 0.001$, in Figure 5(c)). Besides, there is a significant difference between runtimes of the PPUMGAT+ and PPUMGAT- algorithms ($P < 0.001$, in Figure 5(a); $P = 0.007 < 0.05$, in Figure 5(b)), which shows that the designed pre-large concept is helpful to decrease runtime based on a two-way ANOVA analysis. In general, runtime increases when the SP is increased. This is reasonable since when the SP is increased, more sensitive information needs to be hidden. For the accidents data-set, the PPUMGAT- algorithm exceeds 10^4 s, and is thus ignored in Figure 5(c). From results shown in Figures 4 and 5, it can be concluded that the proposed PPUMGAT+ algorithm outperforms PPUMGAT- both under different MUTs and under various SPs. Thus, the adapted pre-large concept to speed up the sanitisation process for hiding sensitive HUIs is acceptable. In addition, the proposed PPUMGAT+ algorithm is also better than PPUMGA+insert for sparse data-sets, which have characteristics that are more similar to real-life data-sets.

6.2. Side effects

In this section, the ratios of side effects in terms of hiding failure (HF_r), missing cost (MC_r), and artificial cost (AC_r) are used as criteria to evaluate the performance of the proposed approach. (HF_r) is the ratio of sensitive HUIs that the sanitisation process failed to hide. (MC_r) is the ratio of non-sensitive HUIs that are missing after sanitisation, and (AC_r) is the ratio of itemsets that have become HUIs as a result of sanitisation. These three ratios corresponding to the three side effects are formally defined as:

$$HF_r = \frac{|HS'|}{|HS|}, \quad (15)$$

$$MC_r = \frac{|\sim HS - \sim HS'|}{|\sim HS|}, \quad (16)$$

$$AC_r = \frac{|\sim HS' - \sim HS|}{|\sim HS|}, \quad (17)$$

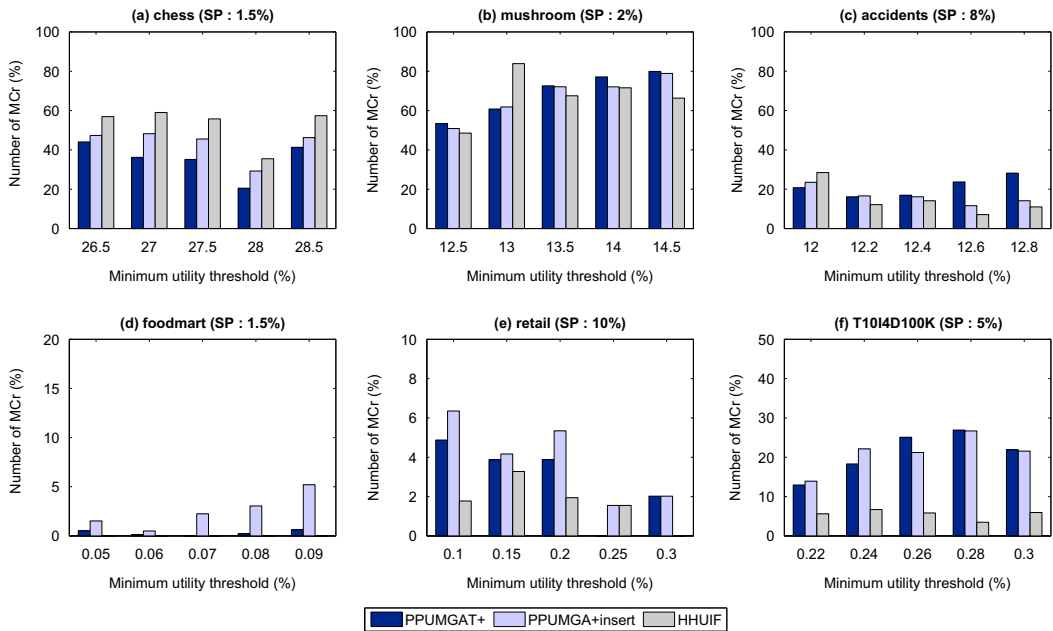
where (HS) is the set of sensitive HUIs before sanitisation and HS' is the set of sensitive HUIs after sanitisation.

Table 4. The HF_r results of HHUIF on foodmart for various SPs.

(MUT = 0.1%)					
SP(%)	2	4	6	8	10
HF_r (%)	0	11.1	0	11.1	9.091

Table 5. The HF_r results of HHUIF on foodmart for various MUTs.

(SP = 1.5%)					
MUT(%)	0.05	0.06	0.07	0.08	0.09
HF_r (%)	20	15.385	20	0	20

**Figure 6.** Missing cost w.r.t. various MUTs.

6.2.1. Hiding failure

All the compared evolutionary algorithms successfully hide all the sensitive HUIs itemsets, while the non-evolutionary HHUIF algorithm did not on the foodmart data-set. Detailed results for the HHUIF algorithm are thus shown in Tables 4 and 5.

Thus, it can be found that HHUIF cannot achieve the purpose of PPUM since some sensitive HUIs are not hidden by its sanitisation process.

6.2.2. Missing cost

The results of MC_r for various MUTs when the SP is fixed are shown in Figure 6.

It can be observed in Figure 6 that the proposed algorithm can generally achieve better results in terms of (MC_r) compared to the PPUMGA+insert algorithm. Although HHUIF directly deletes items (not transactions) to achieve the purpose of PPUM, the proposed algorithm can still obtain better results in some cases in terms of (MC_r). The (MC_r) is high for most data-sets except in Figure 6(c)–(e). These results are convincing since there is a trade-off relationship between the side effects of hiding failure and missing cost. When sensitive HUIs are hidden, more information about non-sensitive HUIs may

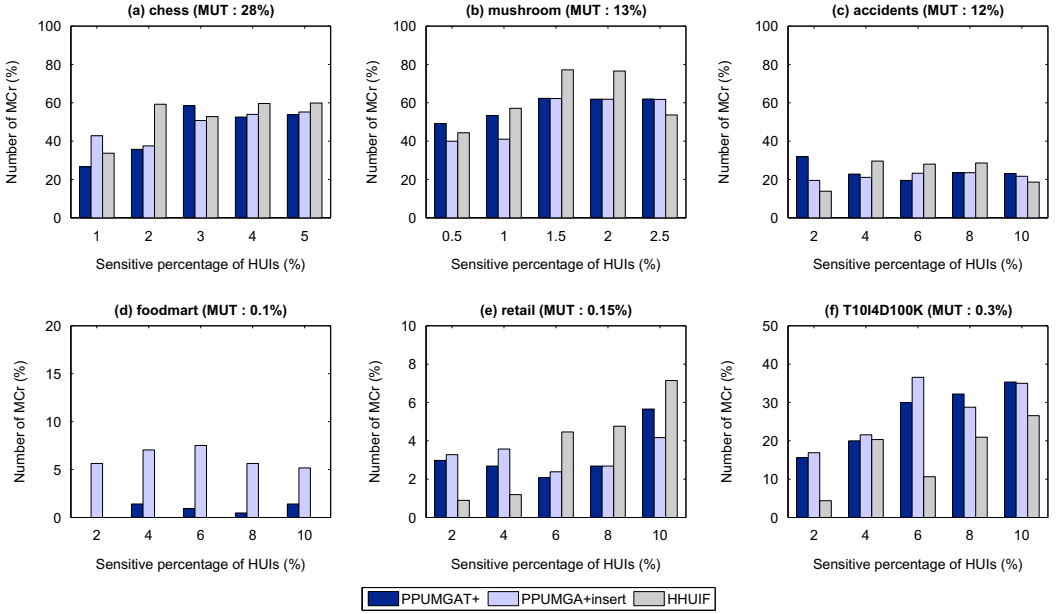


Figure 7. Missing cost for various SPs.

also be hidden at the same time. Based on a two-way ANOVA analysis, there is a significant difference in terms of missing cost for the chess, foodmart and retail data-sets ($F = 22.956$, $P < 0.001$, in Figure 6(a); $F = 8.994$, $P = 0.009 < 0.05$, in Figure 6(d); $F = 8.230$, $P = 0.011 < 0.05$, in Figure 6(e)). However, there is no significant differences in terms of missing cost for the mushroom, accidents, and T10I4D100K data-sets ($F = 2.451$, $P = 0.148 > 0.05$, in Figure 6(b); $F = 1.888$, $P = 0.241 > 0.05$, in Figure 6(c); $F = 0.00384$, $P = 0.996 > 0.05$, in Figure 6(f)).

From the results shown in 6(a), (b) and (f), it can be observed that the (MC_r) ratio can reach values as high as 20%. The reason is that the distribution of the HUIs for these data-sets is too dense. When sensitive HUIs are hidden by the sanitisation process, more non-sensitive HUIs are also hidden as a side effect. For very sparse data-sets such as foodmart and retail, the similarity between transactions is low. As a result, the sensitive HUIs can be completely hidden with a low (MC_r). The results of MC_r for various SPs and a fixed MUT are shown in Figure 7.

From the results presented in Figure 7, it can be observed that the proposed PPUMGAT+ algorithm can achieve better results than the PPUMGA+insert algorithm and HHUIF. As the SP increases, the proposed algorithm and the PPUMGA+insert algorithm can still completely hide all sensitive HUIs and the (HF_r) ratio is thus zero for all data-sets. However, HHUIF cannot hide all sensitive HUIs for the foodmart data-set. The reason was mentioned previously. The proposed PPUMGAT+ algorithm achieves better results in terms of (MC_r) in most cases, compared to the other two algorithms. However, based on a two-way ANOVA analysis, there are no significant differences in terms of missing cost for all data-sets ($F = 0.471$, $P = 0.641 > 0.05$, in Figure 7(a); $F = 2.744$, $P = 0.124 > 0.05$, in Figure 7(b); $F = 0.772$, $P = 0.429 > 0.05$, in Figure 7(c); $F = 0.0000000628$, $P = 1$, in Figure 7(e); $F = 0.481$, $P = 0.639 > 0.01$, in Figure 7(f)) but the foodmart data-set ($F = 135.375$, $P < 0.001$, in Figure 7(d)). The (MC_r) ratio increases when the SP is increased since there is a trade-off relationship between hiding failures and missing cost.

6.2.3. Artificial cost

For the side effect of artificial cost (AC_r), the compared algorithms produce a ratio of nearly zero or zero for (AC_r) for the three dense data-sets (chess, mushroom and accidents). Therefore, the results of

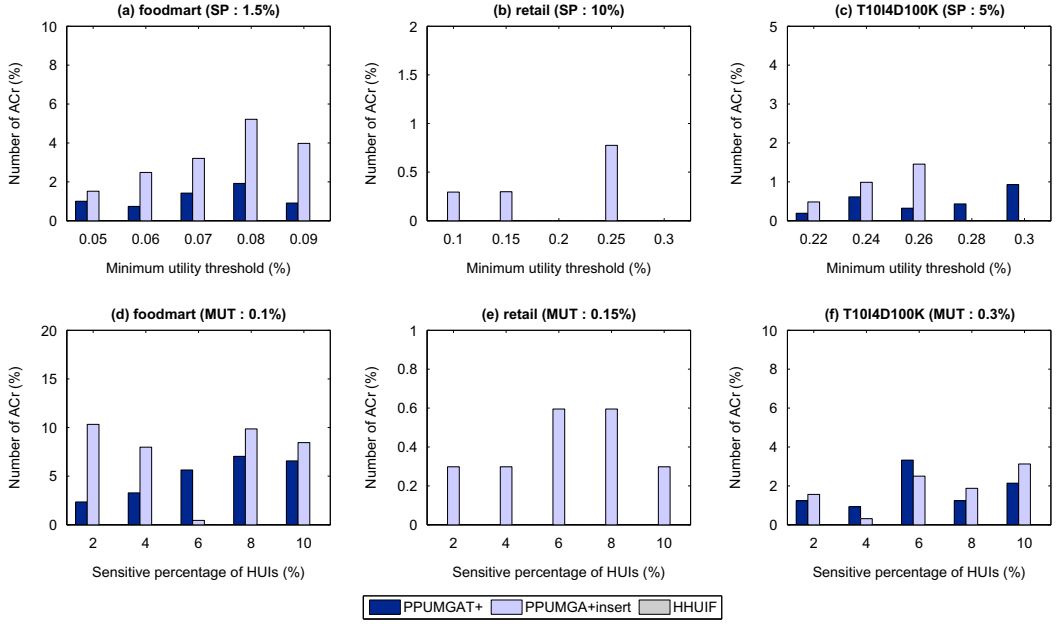


Figure 8. Artificial cost for various SPs and MUTs.

(AC_r) for the dense data-sets are not given. The results of (AC_r) for various MUTs and SPs for the three sparse data-sets are shown in Figure 8.

From the results shown in Figure 8, it is obvious that the proposed PPUMGAT+ algorithm has better results compared to PPUMGA+insert in terms of (AC_r) in most cases. However, HHUIF has not generated any artificial cost in all cases. The reason is that the three data-sets are very sparse. Thus, performing item deletions could not have much influence on other non-sensitive itemsets. Although the proposed PPUMGAT+ algorithm and the compared PPUMGA+insert algorithm generate the side effects of (AC_r) for sparse data-sets (foodmart, retail and T10I4D100K), as shown in Figure 8(a) and (d), (b) and (e), and (c) and (f), the obtained (AC_r) ratio for foodmart, retail and T10I4D100K is, respectively, less than 10, 1 and 4% for the compared algorithms and the proposed PPUMGAT+ algorithm always outperforms the PPUMGA+insert algorithm. Besides, based on a two-way ANOVA analysis, there is a significant difference between PPUMGAT+ and PPUMGA+insert in terms of artificial cost for the foodmart data-set ($F = 16.979$, $P = 0.001 < 0.05$, in Figure 8(a); $F = 113.797$, $P = 0.003 < 0.05$, in Figure 8(d)) but for retail ($F = 0.723$, $P = 0.072 > 0.05$, in Figure 8(b); $F = 32.824$, $P < 0.001$, in Figure 8(e)), and T10I4D100K ($F = 0.0608$, $P = 0.941 > 0.05$, in Figure 8(c); $F = 0.0848$, $P = 0.919 > 0.05$, in Figure 8(f)). Overall, the proposed algorithm has good performance in term of the three side effects, and especially in terms of (HF_r) and (AC_r) for all data-sets.

6.3. Data integrity

Besides the three side effects used in PPUM and PPDMA, a new criterion of database integrity (DI) shows the influence of transaction deletion on the original database, which can be used to verify the similarity between the original database. It is defined as:

$$DI = \frac{|D^*|}{|D|}, \quad (18)$$

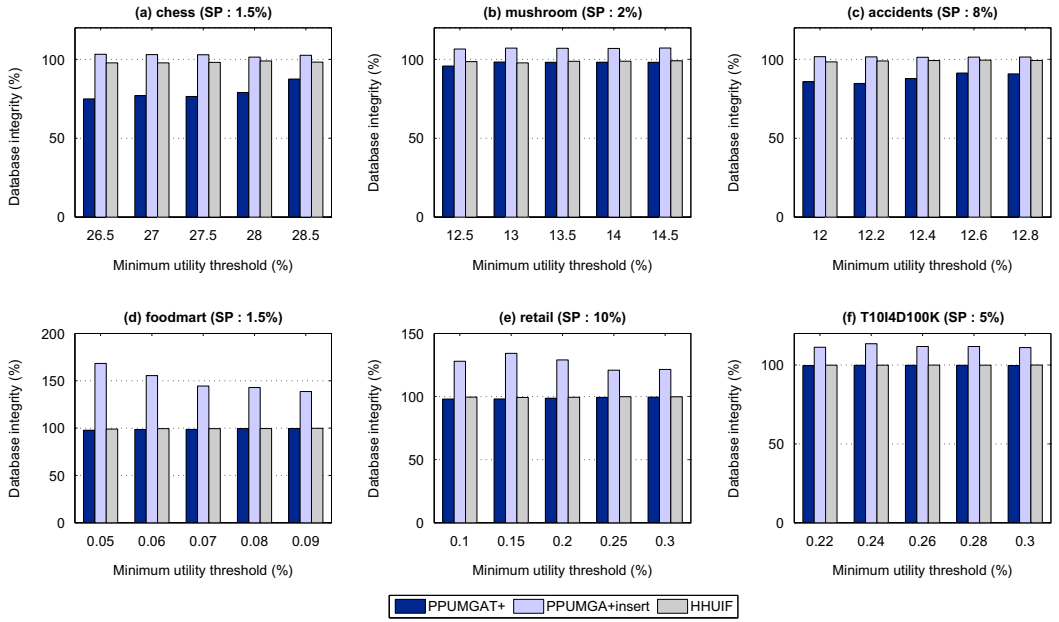


Figure 9. Database integrity for various MUTs.

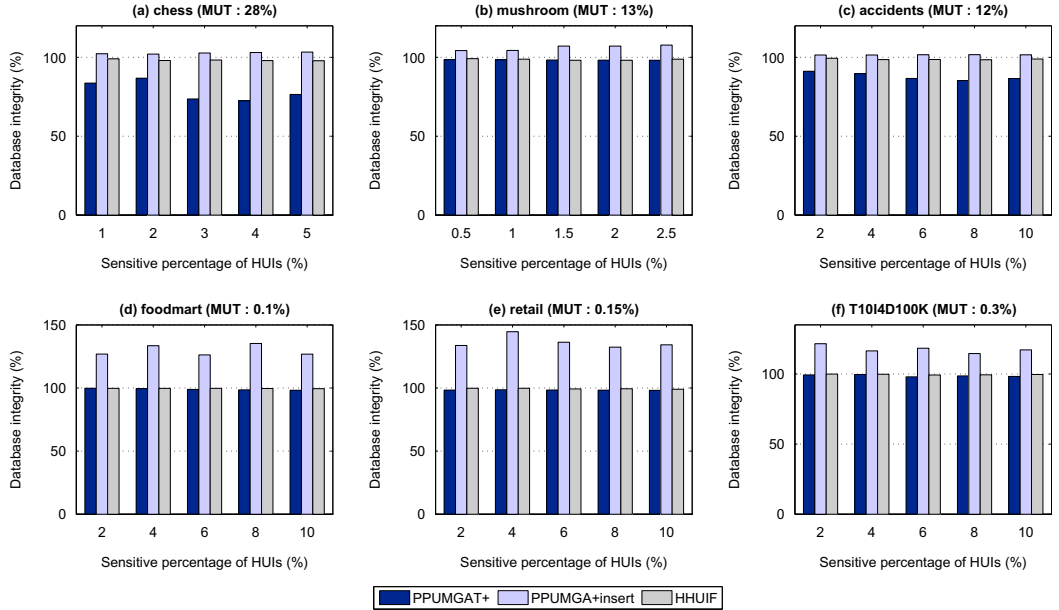


Figure 10. Database integrity for various SPs.

where $|D^*|$ is the database size after sanitisation and $|D|$ is the original database size before sanitisation. The results for the compared algorithms for various MUTs and various sensitive percentages are, respectively, shown in Figures 9 and 10.

In these figures, it is obvious that the PPUMGAT+ algorithm preserves high database integrity (almost 100%) for the three sparse data-sets and also for the dense mushroom data-set. Based on a two-way ANOVA analysis, it can be concluded that the PPUMGAT+ algorithm outperforms the PPUMGAT+insert algorithm for sparse data-sets and for the mushroom data-set ($F = 63.034, P < 0.001$, in Figure 9(b); $F = 42.579, P < 0.001$, in Figure 10(b); $F = 93.435, P < 0.001$, in Figure 9(d); $F = 217.824, P < 0.001$, in Figure 10(d); $F = 131.714, P < 0.001$, in Figure 9(e); $F = 235.843, P < 0.001$, in Figure 10(e); $F = 624.219, P < 0.001$, in Figure 9(f); $F = 167.343, P < 0.001$, in Figure 10(f)). On the other hand, the PPUMGA+insert algorithm influences database integrity by up to 20% or even 50%, especially for the foodmart and retail data-sets.

Also, it can be clearly seen that HHUIF always provides high database integrity for all six data-sets. The reason is that the HHUIF algorithm only deletes the partial sensitive HUIs to achieve the purpose of PPDm, while our algorithm deletes a set of transactions with minimal size effects. For three sparse data-sets and the dense mushroom data-set, the results of the proposed PPUMGAT+ algorithm and the HHUIF algorithm are almost the same. For the chess and accidents data-sets, the performance of the PPUMGAT+ algorithm is worse than HHUIF and the PPUMGA+insert algorithm ($F = 15.777, P : 0.002 < 0.05$, in Figure 9(a); $F = 66.185, P = 0.001 < 0.05$, in Figure 9(c); $F = 50.399, P < 0.001$, in Figure 10(a); $F = 101.505, P < 0.001$, in Figure 10(c)), but it still preserves database integrity by almost 70–80%. In general, the designed algorithm outperforms the PPUMGA+insert algorithm in terms of database integrity for sparse data-sets.

6.4. Utility integrity

Besides database integrity, utility should also be considered in PPUM as an important criterion to evaluate the performance of the compared algorithms. Thus, the concept of utility integrity (UI) is proposed to evaluate the difference in terms of total utility before and after sanitisation.

$$UI = \frac{TU^*}{TU}, \quad (19)$$

where TU^* is the total utility of the database after applying the sanitisation process and TU is the total utility of the database before the sanitisation process.

The results of the compared algorithms for various MUTs and various sensitive percentages are presented in Figures 11 and 12, respectively.

It is obvious that the PPUMGA+insert algorithm cannot ensure high utility integrity for sparse data-sets when it is compared to the designed PPUMGA+ algorithm and HHUIF algorithm. It can be seen that the total utility of the designed algorithm remains stable and close to 100% for most data-sets, and outperforms the PPUMGA+insert algorithm except for the dense chess and accidents data-sets. Also, there is a significant difference between PPUMGAT+ and PPUMGA+insert based on a two-way ANOVA analysis for the dense mushroom database and the three sparse databases ($F = 55.073, P < 0.001$, in Figure 11(b); $F = 41.557, P < 0.001$, in Figure 12(b); $F = 5.834, P < 0.001$, in Figure 11(d); $F = 169.826, P < 0.001$, in Figure 11(e); $F = 561.375, P < 0.001$, in Figure 11(f); $F = 151.971, P < 0.001$, in Figure 12(d); $F = 189.492, P < 0.001$, in Figure 12(e); $F = 139.751, P < 0.001$, in Figure 12(f)). Since the HHUIF algorithm preserves the privacy only by deleting partial sensitive HUIs, its utility integrity still remains very high (almost 100%) for all databases. The results of the proposed PPUMGAT+ algorithm are almost the same for the dense mushroom data-set and the three sparse data-sets. However, for very dense data-sets such as chess and accidents, the designed PPUMGA+ algorithm performs worse than the HHUIF and the PPUMGA+insert algorithms. This is reasonable since for very dense data-sets, when sensitive HUIs are deleted, many related HUIs are also removed and the total utility in the sanitised data-set decreases. Based on a two-way ANOVA analysis, there is a significant difference between the two GA-based compared algorithms ($F = 15.704, P = 0.002 < 0.05$, in Figure 11(a); $F = 70.094, P = 0.001 < 0.05$, in Figure 11(c); $F = 50.306, P < 0.001$, in Figure 12(a); $F = 103.499, P < 0.001$, in Figure 12(c)).

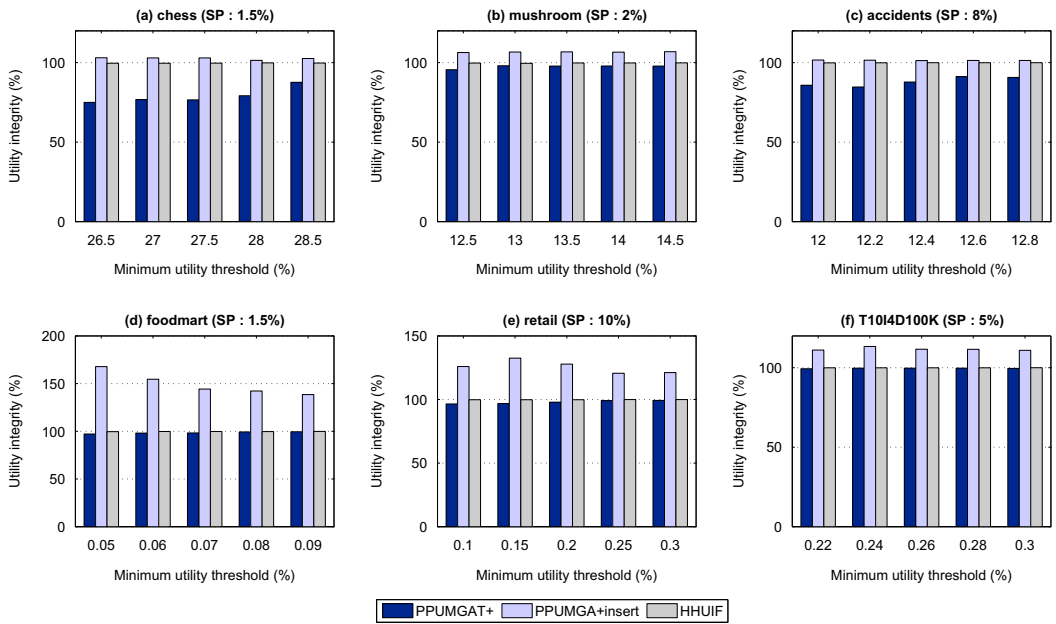


Figure 11. Utility integrity for various MUTs.

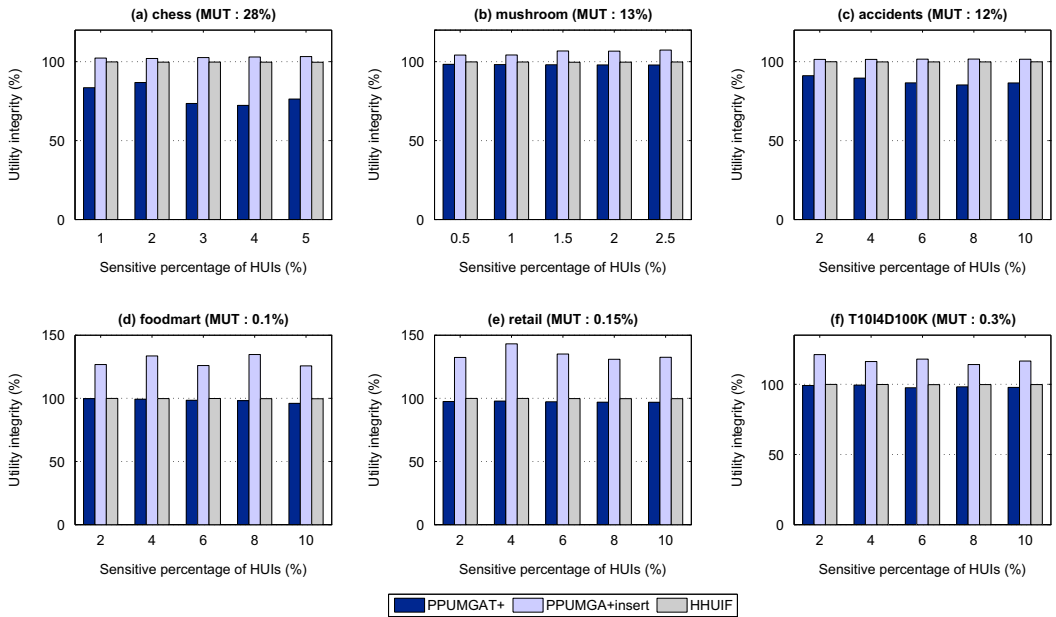


Figure 12. Utility integrity for various SPs.

7. Conclusion and discussion

Using data mining techniques, implicit or potential relationships between items in a database can be easily discovered. As a result, private or confidential information can also be revealed, which may cause security threats and lead to privacy issues. Thus, PPDM has emerged as an important issue in

recent years. Recently, PPUM has also become a critical issue as it addresses the problem of preserving confidential information for HUIM. Since the purpose of PPDM or PPUM is to hide sensitive information while ensuring that important non sensitive information can still be discovered, finding an optimal solution to this problem is NP-hard.

In this paper, we first proposed an optimisation approach to hide sensitive HUIs based on a GA and the operation of transaction deletion. An algorithm named PPUMGAT has been presented to find an optimal set of transactions to be deleted that minimises the three side effects based on the designed fitness function. An improved pre-large concept strategy has also been developed to speed up the evolution process. Based on the designed pre-large concept strategy, multiple database scans can be avoided and runtime is thus decreased. The conducted experiments have shown that the proposed PPUMGAT+ algorithm can successfully hides all sensitive HUIs, while maintaining high database and utility integrity.

This paper has focused on transaction deletion as a mechanism for hiding sensitive high-utility itemsets. As an improvement to be considered in future work, each item within a sensitive HUIs in transactions could be individually encoded as a gene for the sanitisation process. Moreover, the transaction-weighted downward closure property can be further applied in PPUM if some sensitive itemsets in the database can be modified to reduce side effects. Since PPUM is an emerging research problem that has attracted the attention of many researchers in recent years, it is also a critical issue to develop an efficient anonymity mechanism for PPUM.

Disclosure statement

No potential conflict of interest was reported by the authors.

Funding

This work was partially supported by the open fund of Fujian Provincial Key Laboratory of Big Data Mining and Applications (Fujian University of Technology; the National Natural Science Foundation of China (NSFC) under [grant number 61503092]; the Tencent Project under [grant number CCF-Tencent IAGR20160115].

ORCID

Philippe Fournier-Viger  <http://orcid.org/0000-0002-7680-9899>

References

- Aggarwal, C. C., Pei, J., & Zhang, B. (2006). On privacy preservation against adversarial data mining. *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 510–516.
- Agrawal, R., & Srikant, R. (1994a). Quest synthetic data generator. Retrieved from <http://www.Almaden.ibm.com/cs/quest/syndata.html>
- Agrawal, R., & Srikant, R. (1994b). Fast algorithms for mining association rules in large databases. *The International Conference on Very Large Data Bases*, Morgan Kaufmann Publishers, San Francisco, CA, USA, pp. 487–499.
- Agrawal, R., & Srikant, R. (2000). Privacy-preserving data mining. *ACM SIGMOD Record*, 29, 439–450.
- Atallah, M., Elmagarmid, A., Ibrahim, M., Bertino, E., & Verykios, V. (1999). Disclosure limitation of sensitive rule. *The Workshop on Knowledge and Data Engineering Exchange*, Chicago, IL, pp. 45–52.
- Bonam, J., Reddy, A. R., & Kalyani, G. (2014). Privacy preserving in association rule mining by data distortion using pso. *Advances in Intelligent Systems and Computing*, 249, 551–558.
- Chan, R., Yang, Q., & Shen, Y. D. (2003). Mining high utility itemsets. *IEEE International Conference on Data Mining*, Melbourne, FL, pp. 19–26.
- Cheng, P., Lin, C. W., & Pan, J. S. (2015). Use hype to hide association rules by adding items. *PLOS One*, 10(6), 1–19.
- Cheng, P., Roddick, J. F., Chu, S. C., & Lin, C. W. (2016). Privacy preservation through a greedy, distortion-based rule-hiding method. *Applied Intelligence*, 44, 295–306.
- Clifton, C., Kantarcioglu, M., Vaidya, J., Lin, X., & Zhu, M. Y. (2002). Tools for privacy preserving distributed data mining. *ACM SIGKDD Explorations Newsletter*, 4, 28–34.

- Evfimievski, A., Srikant, R., Agrawal, R., & Gehrke, J. (2002). Privacy preserving mining of association rules. *ACM International Conference on Knowledge Discovery and Data Mining*, pp. 217–228.
- Fournier-Viger, P., Lin, J. C. W., Gomariz, A., Gueniche, T., Soltani, A., Deng, Z., & Lam, H. T. (2016). The spmf open-source data mining library version 2. *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 36–40.
- Fournier-Viger, P., Lin, J. C. W., Kiran, R. U., Koh, Y. S., & Thomas, R. (2017). A survey of sequential pattern mining. *Data Science and Pattern Recognition*, 1, 54–77.
- Giannotti, F., Lakshmanan, L. V. S., Monreale, A., Pedreschi, D., & Wang, H. (2013). Privacy-preserving mining of association rules from outsourced transaction databases. *IEEE Systems Journal*, 7, 385–395.
- Han, J., Pei, J., Yin, Y., & Mao, R. (2004). Mining frequent patterns without candidate generation: A frequent-pattern tree approach. *Data Mining and Knowledge Discovery*, 8, 53–87.
- Holland, J. H. (1992). *Adaptation in natural and artificial systems*. Cambridge, MA: MIT Press.
- Hong, T. P., & Wang, C. Y. (2006). Maintenance of association rules using pre-large itemsets. *Intelligent Databases: Technologies and Applications*, pp. 44–60.
- Hong, T. P., Lin, C. W., Yang, K. T., & Wang, S. L. (2013). Using tf-idf to hide sensitive itemsets. *Applied Intelligence*, 38, 502–510.
- Lin, C. W., Hong, T. P., & Lu, W. H. (2009). The pre-fufp algorithm for incremental mining. *Expert Systems with Applications*, 36, 9498–9505.
- Lin, C. W., Hong, T. P., & Lu, W. H. (2011). An effective tree structure for mining high utility itemsets. *Expert Systems with Applications*, 38, 7419–7424.
- Lin, C. W., Hong, T. P., Chang, C. C., & Wang, S. L. (2013). A greedy-based approach for hiding sensitive itemsets by transaction insertion. *Journal of Information Hiding and Multimedia Signal Processing*, 4, 201–214.
- Lin, C. W., Hong, T. P., Wong, J. W., Lan, G. C. & Lin, W. Y. (2014). Ga-based approach to hide sensitive high utility itemsets. *Scientific World Journal*, 12 p. Article ID 804629.
- Lin, J. C. W., Gan, W., Fournier-Viger, P., Hong, T. P., & Tseng, V. S. (2015). Efficient algorithms for mining high-utility itemsets in uncertain databases. *Knowledge-based Systems*, 96, 171–187.
- Lin, J. C. W., Wu, T. Y., Fournier-Viger, P., Lin, G., Hong, T. P., & Pan, J. S. (2015). A sanitization approach of privacy preserving utility mining. *Advances in Intelligent Systems and Computing*, 388, 47–57.
- Lin, J. C. W., Fournier-Viger, P., & Gan, W. (2016). Fhn: An efficient algorithm for mining high-utility itemsets with negative unit profits. *Knowledge-Based Systems*, 111, 283–298.
- Lin, J. C. W., Gan, W., Fournier-Viger, P., Hong, T. P., & Zhan, J. (2016). Efficient mining of high-utility itemsets using multiple minimum utility thresholds. *Knowledge-Based Systems*, 113, 100–115.
- Lindell, Y., & Pinkas, B. (2000). Privacy preserving data mining. *The Annual International Cryptology Conference on Advances in Cryptology*, Santa Barbara, CA, USA, pp. 36–54.
- Liu, J., Wang, K., & Fung, B. (2016). Mining high utility patterns in one phase without generating candidates. *IEEE Transactions on Knowledge and Data Engineering*, 28, 1245–1257.
- Liu, M., & Qu, J. (2012). Mining high utility itemsets without candidate generation. *ACM International Conference on Information and Knowledge Management*, pp. 55–64.
- Liu, Y., Liao, W. K., & Choudhary, A. (2005). A two-phase algorithm for fast discovery of high utility itemsets. *Lecture Notes in Computer Science*, 3518, 689–695.
- Oliveria, S. R. M., & Zaiane, O. R. (2002). Privacy preserving frequent itemset mining. *IEEE International Conference on Privacy, Security and Data Mining*, 14, 43–54.
- Rajalaxmi, R. R., & Nataraja, A. M. (2012). Effective sanitization approaches to hide sensitive utility and frequent itemsets. *Intelligent Data Analysis*, 16, 933–951.
- Verykios, V. S., Bertino, E., Fovino, I. N., Provenza, L. P., Saygin, Y., & Theodoridis, Y. (2004). State-of-the-art in privacy preserving data mining. *ACM SIGMOD Record*, 33, 50–57.
- Wu, Y. H., Chiang, C. M., & Chen, A. L. P. (2007). Hiding sensitive association rules with limited side effects. *IEEE Transactions on Knowledge and Data Engineering*, 19, 29–42.
- Yao, H., & Hamilton, H. J. (2006). Mining itemset utilities from transaction databases. *Data & Knowledge Engineering*, 59, 603–626.
- Yao, H., Hamilton, H. J., & Butz, C. J. (2004). A foundational approach to mining itemset utilities from databases. *SIAM International Conference on Data Mining*, Orlando, FL, pp. 482–486.
- Yeh, J. S., & Hsu, P. C. (2010). Hhuif and msic: novel algorithms for privacy preserving utility mining. *Expert Systems with Applications*, 37, 4779–4786.
- Yun, U., & Kim, J. (2015). A fast perturbation algorithm using tree structure for privacy preserving utility mining. *Expert Systems with Applications*, 42, 1149–1165.