Mining Potential High-Utility Itemsets over Uncertain Databases

Journal:	*Transactions on Knowledge and Data Engineering*
Manuscript ID:	TKDE-2015-01-0105
Manuscript Type:	Regular
Keywords:	H.2.8.i Mining methods and algorithms < H.2.8 Database Applications < H.2 Database Management < H Information Technology and Systems, H.2.8.d Data mining < H.2.8 Database Applications < H.2 Database Management < H Information Technology and Systems

JOURNAL OF LATEX CLASS FILES, VOL. 6, NO. 1, NOVMEMBER 2014

Mining Potential High-Utility Itemsets over Uncertain Databases

Jerry Chun-Wei Lin, *Member, IEEE,* Wensheng Gan, Tzung-Pei Hong, *Member, IEEE,* and Vincent S. Tseng, *Senior Member, IEEE*

Abstract—High-utility itemsets mining (HUIM) is an extension of frequent itemsets mining that incorporates the concept of utility (e.g., profit) over a certain database. In real-life applications, however, an item or itemset is not only present or absent in the transactions but also associated with an existing probability. The topic of mining HUIs from uncertain databases has not yet been addressed though it is commonly seen in real-world applications. In this paper, we proposed novel algorithms for mining potential high-utility itemsets (PHUIs) over uncertain databases. An itemset of PHUIs indicated that it has not only the high utility but also the high probability of existence based on the tuple uncertainly mechanism. The apriori-based potential high-utility itemsets (PHUI-apriori) mining algorithm is firstly presented to level-wisely mine PHUIs. Since PHUI-apriori applies the generate-and-test framework to mine PHUIs, the secondly list-based potential high-utility itemsets (PHUI-list) mining algorithm is then developed to directly mine PHUIs based on the designed probability-utility (PU)-list structure without candidate generation, thus greatly improving the scalability for mining PHUIs. Substantial experiments were conducted on both real-life and synthetic datasets to show the performance of two designed algorithms in terms of runtime, number of patterns, memory consumption, and scalability.

Index Terms—High-utility itemsets mining, uncertain databases, probabilistic-based, level-wise, PU-list structure.

1 Introduction

The main purpose of Knowledge Discovery in Database (KDD) is to discover meaningful and useful information from a collection of data. Depending on different requirements in various domains and applications, association-rule mining (ARM) [1, 2, 3] is an important and common issue in KDD. Agrawal et al. first designed the well-known Apriori algorithm to mine ARs in a level-wise way [4]. Han et al. then presented FP-growth algorithm to directly mine frequent itemsets without candidate generation [5]. Traditional algorithms of ARM only consider whether or not the item or itemset is present in a transaction.

High-utility itemsets mining (HUIM) [6, 7, 8, 9, 10, 11] incorporates the concept of utility (e.g., worth, profit, etc.) of an item or itemset to measure how useful an item or itemset is. An itemset is defined as a high-utility itemset (HUI) if its utility value in the databases is no less than a user-specified minimum utility count. The goal of HUIM is to identify the rare items or itemsets in the transactions, but it can bring valuable profits for retailers.

In real-life applications, data may be uncertainly collected from incomplete data sources, such as RFID, GPS, wireless sensors, or WiFi systems [12, 13]. In basket analysis of uncertain databases, each transaction contains several items with their purchase probabilities for the customers. For instance, the transaction (A:2, C:3, E:2, 90%) indicates that three items (A, C, E) with quantities (2, 3, 2) can be bought with 90% probability. Several algorithms were proposed to mine frequent itemsets under uncertain databases [14, 15, 16, 17, 18].

The existing algorithms of HUIM have been developed to handle a precise database, which is insufficient in real-life applications. An item or itemset is, however, not only present or absent in the transactions but also associated with an existential probability, especially the collected data from experimental measurements or noisy sensors. In particular, numerous discovered HUIs may not be the interest patterns to help managers or retailers for making the efficient decisions without considering the probability factor. It may be misleading if the discovered HUIs with low existential probability. In fact, it is more interested in finding the high existential probability and high profitable patterns. In this paper, a novel potential high-utility itemsets mining (PHUIM) model is designed to mine the condensed and meaningful patterns named

- *Jerry Chun-Wei Lin and Wensheng Gan are with the Innovative Information Industry Research Center (IIIRC), School of Computer Science and Technology, Harbin Institute of Technology Shenzhen Graduate School, Shenzhen, China, 518055. E-mail: jerrylin@ieee.org; wsgan001@gmail.com Website: http://ikelab.net*
- *Tzung-Pei Hong is with the Department of Computer Science and Information Engineering, National University of Kaohsiung, Kaohsiung, Taiwan, R.O.C. 700, and with the Department of Computer Science and Engineering, National Sun Yat-sen University, Kaohsiung, Taiwan, R.O.C., 80424. E-mail: tphong@nuk.edu.tw*
- *Vincent S. Tseng is with the Department of Computer Science and Information Engineering, National Cheng Kung University, Tainan, Taiwan, R.O.C., 701. E-mail: tsengsm@mail.ncku.edu.tw*

potential high-utility itemsets (PHUIs). The apriori-based potential high-utility itemsets (PHUI-apriori) mining algorithm and list-based potential high-utility itemsets (PHUI-list) mining algorithm are respectively developed based on the level-wise approach and the designed probability-utility (PU)-list structure to mine PHUIs. Major contributions of this paper are summarized as follows:

1) Previous works on HUIM have addressed the mining of HUIs from precise databases. This is the first paper to address the issue of mining high-utility itemsets from uncertain databases.
2) Two mining algorithms, PHUI-apriori and PHUI-list, are respectively designed to efficiently mine PHUIs over uncertain databases, which can be used as the state-of-the-art algorithms for the further research of HUIM or PHUIM.
3) PHUI-apriori algorithm is proposed as a baseline algorithm for level-wisely mining PHUIs over uncertain databases based on the Apriori-like mechanism and two-phase model. As an improved algorithm, PHUI-list algorithm is proposed for directly discovering PHUIs without candidate generation in a level-wise way.
4) Substantial experiments on both real-life and synthetic datasets showed that the two proposed algorithms can effectively discover the complete PHUIs over uncertain databases.

The remainder of this paper is organized as follows. Related works are briefly reviewed in Section 2. The preliminaries and problem statement are presented in Section 3. Two proposed algorithms are described in Section 4. The conducted experiments of two proposed algorithms are provided in Section 5. Conclusions and future works are finally presented in Section 6.

2 Related Works

In this section, some related works of mining frequent itemsets over uncertain databases and high-utility itemsets mining (HUIM) are briefly reviewed.

2.1 Mining Frequent Itemsets over Uncertain Databases

Most approaches for mining frequent itemsets of ARs are processed to handle binary databases, which only indicates that an item or itemset is present or absent in the transaction. In real-life applications, a huge amount of the data collected from wireless sensor networks may be inaccurate or incomplete [12, 13]. Discovering the frequent itemsets over uncertain databases has emerged as an important issue in recent years [17, 18]. Depending on the specific applications, approaches for mining uncertain frequent itemsets over uncertain databases can be generally classified into two classes: the expected support-based model and the probabilistic frequency model.

For the expected support-based model, Chui et al. first designed UApriori algorithm with the defined expected support threshold [14] to level-wisely mine frequent itemsets over uncertain databases based on the well-known Apriori algorithm. Leung et al. designed the UFP-growth algorithm [15] to mine the uncertain frequent itemsets without candidate generation based on the extended frequent pattern (FP)-tree structure, divide-and-conquer model, and depth-first search strategy. Aggarwal et al. extended the H-Mine algorithm to present the UH-mine algorithm [19] to recursively mine the uncertain frequent itemsets from the designed UH-Struct structure as well as adopting the divide-and-conquer model and the depth-first search strategy. Lin et al. then designed a compressed uncertain frequent pattern (CUFP)-tree structure to efficiently mine the uncertain frequent itemsets [16].

For the probabilistic frequency model, Bernecker et al. first proposed a new probabilistic formulation for mining frequent itemsets based on *possible world semantics* model [20]. Sun et al. also developed p-FP structure and proposed two efficient algorithms to discover frequent patterns based on bottom-up (p-Apriori) and top-down (TODIS) manners [21]. Besides the frequent itemsets mining over uncertain databases, approaches for mining uncertain frequent itemsets over data streams have been proposed [22], such as UF-streaming and SUF-growth. Tong et al. then combined the expected support-based model and probabilistic frequency model to present a new way for mining frequent itemsets over uncertain databases with the uniform measure [23].

2.2 High-Utility Itemsets Mining

High-utility itemsets mining (HUIM) is based on the measurement of local utility (quantity) and external utility (profit) to find the rare frequencies of itemsets with high profits, which is an extension of frequent itemset mining. Chan et al. designed top-k high-utility closed patterns for deriving both positive and negative utilities [6]. Yao et al. defined the problem of utility mining by analyzing the utility relationships of the itemsets [11]. The utility bound and support bound properties are defined, forming the mathematical mode for mining HUIs.

Since the downward closure (DC) property is no longer kept for HUIM, Liu et al. presented a two-phase model [8] to efficiently discover HUIs based on the designed transaction-weighted downward closure (TWDC) property. Based on two-phase model for mining HUIs, Lin et al. proposed the HUP-tree algorithm [24] to find HUIs without candidate generation. Vincent et al. proposed the UP-tree structure and developed two mining algorithms, UP-growth [10] and UP-growth+ [25], to efficiently derive HUIs. Liu et al. proposed the HUI-Miner algorithm [7] to build utility-list structure and to develop an enumeration tree to both prune the search space and directly extract HUIs without either candidate generation or an additional database rescan. Fournier-Viger et al. further designed a FHM algorithm by enhancing the property of HUI-Miner for analyzing the co-occurrences among 2-itemsets [9].

The development of other algorithms for HUIM is still in progress, but most of them are processed to handle precise databases. To the best of our knowledge, however, mining high-utility itemsets over uncertain databases has not yet been developed. This is the first work to present the problem of mining potential high-utility itemsets (PHUIs) over uncertain databases.

3 Preliminaries and Problem Statement

In this section, preliminary and problem statement related to potential high-utility itemsets mining (PHUIM) over uncertain databases are given below.

3.1 Preliminaries

In this paper, the tuple uncertainty model and the expected support-based frequent pattern mining model are adopted in two proposed algorithms. Let $I = \{i_1, i_2, \ldots, i_m\}$ be a finite set of m distinct items in uncertain quantitative databases $D = \{T_1, T_2, \ldots, T_n\}$, where each transaction $T_q \in D$ is a subset of I, contains several items with their purchase quantities $q(i_j, T_q)$, and has an unique identifier, *TID*. In addition, each transaction has a unique probability of existence $p(T_q)$, which indicates that T_q exists in D with probability $p(T_q)$ based on a tuple uncertainly model. A corresponding profit table, $ptable = \{pr_1, pr_2, \ldots, pr_m\}$, in which pr_j is the profit value of an item i_j, is created. An itemset X is a set of k distinct items $\{i_1, i_2, \ldots, i_k\}$, where k is the length of an itemset called k-itemset. An itemset X is said to be contained in a transaction T_q if $X \subseteq T_q$. Two thresholds, the minimum utility threshold and the minimum expected support threshold, are respectively defined as ε and μ.

An example of uncertain quantitative databases with its probabilistic values is shown in Table 1. The corresponding utility table is shown in Table 2. In this example, the minimum utility threshold and the minimum expected support threshold are respectively set at ε (= 25%) and μ (= 15%).

Definition 1. The utility of an item i_j in T_q is defined as:

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

For example, the utility of an item (C) in T_1 is $u(C, T_1) = q(C, T_1) \times pr(C) = (3 \times 12) = 36$.

Definition 2. The probability of an itemset X occurring in T_q is denoted as $p(X, T_q)$, which can be defined as:

$$p(X, T_q) = p(T_q),$$

where $p(T_q)$ is the corresponding probability of T_q.

For example, $p(C, T_1) = p(T_1) = 0.9$, and $p(AC, T_1) = p(T_1) = 0.9$.

TABLE 1: An uncertain database

TID	A	B	C	D	E	Probability
1	2	0	3	0	2	0.9
2	0	1	0	2	0	0.7
3	1	2	1	0	3	0.85
4	0	0	2	0	0	0.5
5	0	3	0	2	1	0.75
6	2	0	2	5	0	0.7
7	1	1	0	4	1	0.45
8	0	4	0	0	1	0.36
9	3	0	3	2	0	0.81
10	0	2	3	0	1	0.6

TABLE 2: An utility table

TID	A	B	C	D	E
Profit	4	1	12	6	15

Definition 3. The utility of an itemset X in transaction T_q is denoted as $u(X, T_q)$, which can be defined as:

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

For example, the utility of (AC) in T_1 is calculated as $u(AC, T_1) = u(A, T_1) + u(C, T_1) = q(A, T_1) \times pr(A) + q(C, T_1) \times pr(C) = (2 \times 4) + (3 \times 12) = 44$.

Definition 4. The utility of an itemset X in D is denoted as $u(X)$, which can be defined as:

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

For example, $u(A) = u(A, T_1) + u(A, T_3) + u(A, T_6) + u(A, T_7) + u(A, T_9) = (8 + 4 + 8 + 4 + 12) = 36$, and $u(AC) = u(AC, T_1) + u(AC, T_3) + u(AC, T_6) + u(AC, T_9) = (44 + 16 + 32 + 48) = 140$.

Based on the expected support-based model in uncertain data mining [14], the PHUIs in uncertain databases are defined as follows.

Definition 5. The expected support count of an itemset X in D is denoted as $expSup(X)$, which can be defined as:

$$expSup(X) = \sum_{X \subseteq T_q \wedge T_q \in D} p(X, T_q).$$

For example, $expSup(A) = p(A, T_1) + p(A, T_3) + p(A, T_6) + p(A, T_7) + p(A, T_9) = (0.9 + 0.85 + 0.7 + 0.45 + 0.81) = 3.71$, and $expSup(ABE) = p(ABE, T_3) + p(ABE, T_7) = (0.85 + 0.45) = 1.3$.

Definition 6. The transaction utility of transaction T_q is denoted as $tu(T_q)$, which can be defined as:

$$tu(T_q) = \sum_{j=1}^{m} u(i_j, T_q),$$

in which m is the number of items in T_q.

For example, $tu(T_1) = u(A, T_1) + u(C, T_1) + u(E, T_1)$ (= 8 + 36 + 30) = 74.

Definition 7. The total utility in D is the sum of all transaction utilities in D and is denoted as TU, which can be defined as:

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

For example, the transaction utilities for T_1 to T_{10} are respectively calculated as $tu(T_1) = 74$, $tu(T_2) = 13$, $tu(T_3) = 63$, $tu(T_4) = 24$, $tu(T_5) = 30$, $tu(T_6) = 62$, $tu(T_7) = 44$, $tu(T_8) = 19$, $tu(T_9) = 60$, and $tu(T_{10}) = 53$. The total utility in D is the sum of all transaction utilities in D, which is calculated as: TU (= 74 + 13 + 63 + 24 + 30 + 62 + 44 + 19 + 60 + 53) = 442.

Definition 8. An itemset X is defined as a high-utility itemset (HUI) if its utility value $u(X)$ is larger than or equals to the minimum utility count as:

$$\sum_{X \subseteq T_q \wedge T_q \in D} u(X, T_q) = u(X) \geq \varepsilon \times TU.$$

For example, suppose that the minimum utility threshold ε is set at 25%. An item (A) is not considered as a HUI since $u(A) = 36$, which is smaller than (0.25 $\times$ 442) = 110.5. An itemset (AC) is considered as a HUI in D since $u(AC) = 140$, which is larger than the minimum utility count (= 110.5).

Definition 9. An itemset X is denoted as a high probability itemset (HPI) if the expected support count of an itemset X is larger than or equal to the minimum expected support count, which is defined as:

$$\sum_{X \subseteq T_q \wedge T_q \in D} p(X, T_q) = expSup(X) \geq \mu \times |D|.$$

For example, since μ is set at 15%, the minimum expected support count is calculated as (15% $\times$ 10) = 1.5. Thus, an item (A) is considered as a high probability itemset since $expSup(A) = 3.71 > 1.5$. An itemset (ABE) is, however, not considered as a high probability itemset since its expected support count is calculated as $expSup(ABE) = 1.3$, which is smaller than the minimum expected support count (= 1.5).

Definition 10. An itemset X in D is defined as a potential high-utility itemset (PHUI) if it satisfies the conditions: (1) X is a HUI; (2) X is a HPI.

Based on the above definitions, the problem statement of mining PHUIs over uncertain databases can be formulated as follows.

3.2 Problem Statement

Given an uncertain database D with total utility is TU, the minimum utility threshold and the minimum expected support threshold are respectively set as ε. The problem of PHUIM over uncertain databases is to mine PHUIs whose utilities are larger than or equal to $(\varepsilon \times TU)$, and its expected support count is larger than or equal to $(\mu \times |D|)$.

From the example given in Tables 1 and 2, the set of PHUIs is shown in Table 3 when the minimum utility threshold is set at ε = 25% and the minimum expected support threshold is set at μ = 15%.

TABLE 3: Derived PHUIs over an uncertain database

Itemset	Actual utility	expSup
(C)	168	4.36
(E)	135	3.91
(AC)	140	3.26
(BE)	117	3.01
(CE)	174	2.35
(ACD)	122	1.51
(ACE)	135	1.75

4 Proposed Potential High-Utility Itemsets Mining Algorithms over Uncertain Databases

In this section, the PHUI-apriori algorithm is first proposed as a baseline algorithm to level-wisely mine PHUIs over uncertain databases. PHUI-list algorithm is further designed to improve the performance of PHUI-apriori algorithm to directly discover PHUIs over uncertain databases based on the designed probability-utility (PU)-list structure and an enumeration tree without candidate generation each time. Two developed algorithms are described below.

4.1 Proposed PHUI-apriori Algorithm

To the best of our knowledge, this is the first paper to discuss the PHUIM over uncertain databases. A PHUI-apriori algorithm is firstly presented here to mine PHUIs in a level-wise way.

4.1.1 Pruning strategy by downward closure property

In the well-known Apriori algorithm, the downward closure (DC) property is kept to reduce the number of candidates for ARM. The DC property is also kept in the designed PHUI-apriori algorithm for mining PHUIs.

Theorem 1. ***(Downward Closure Property of High Probability Itemsets)*** *An itemset is obtained the downward closure (DC) property if it is a potential high-utility itemset over uncertain databases.*

Proof: Let k-itemset be X^k, and its subset be X^{k-1}. Since $p(X^k, T_q) = p(T_q)$, for any transaction T_q in D, it can be found that $\frac{p(X^k, T_q)}{p(X^{k-1}, T_q)} \geq 1$. Since X^{k-1} is subset of X^k, thus,

$$\begin{aligned} expSup(X^k) &= \sum_{X^k \subseteq T_q \wedge T_q \in D} p(X^k, T_q) \\ &\leq \sum_{X^{k-1} \subseteq T_q \wedge T_q \in D} p(X^{k-1}, T_q) \\ &= expSup(X^{k-1}). \end{aligned}$$

Thus, if X^k is a HPI, its expected support count is larger than or equals to the minimum expected support count as *expSup*(X^k) $\geq \mu$, and *expSup*(X^{k-1}) $\geq$ *expSup*(X^k) $\geq \mu$. □

Corollary 1. *If an itemset X^k is a HPI, every subset X^{k-1} of X^k is a HPI.*

Corollary 2. *If an itemset X^k is not a HPI, no superset X^{k+1} of X^k is a HPI.*

In HUIM, the DC property of ARM could not be directly extended to mine HUIs. The TWDC property [8] was proposed to reduce the search space in HUIM.

Definition 11. The transaction-weighted utility (TWU) of an itemset X is the sum of all transaction utilities $tu(T_q)$ containing an itemset X, which is defined as:

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

Definition 12. An itemset X is considered as a high transaction-weighted utilization itemset (HTWUI) if its $TWU(X) \geq TU \times \varepsilon$.

In Table 1, the TWU of an item (E) is calculated as $TWU(E)$ = $tu(T_1)$ + $tu(T_3)$ + $tu(T_5)$ + $tu(T_7)$ + $tu(T_8)$ + $tu(T_{10})$ = (74 + 63 + 30 + 44 +19 + 53) = 283. An item (E) is considered as a HTWUI since $TWU(E)$ = 283 $>$ 134. The TWDC property is also extended to the mining of PHUIs over uncertain databases, and a novel transaction-weighted probabilistic and utilization downward closure (TWPUDC) property is further designed to reduce the search space of PHUI-apriori algorithm for mining PHUIs.

Definition 13. An itemset X is defined as a high transaction-weighted probabilistic and utilization itemset (HTWPUI) if $TWU(X) \geq \varepsilon \times TU$ and *expSup*(X) $\geq \mu \times |D|$.

For example, in Tables 1 and 2, since μ is set at 15%, the minimum expected support count is calculated as (15% $\times$ 10) = 1.5. For example of an item (E), $TWU(E)$ = 283 $>$ 134. In addition, *expSup*(E) = $p(E, T_1)$ + $p(E, T_3)$ + $p(E, T_5)$ + $p(E, T_7)$ + $p(E, T_8)$ + $p(E, T_{10})$ (= 0.9 + 0.85 + 0.75 + 0.45 + 0.36 + 0.6) = 3.91 $>$ 1.5. Thus, an item (E) is considered as a high transaction-weighted probabilistic and utilization itemset (HTWPUI).

Theorem 2. ***(Downward Closure Property of HTWPUI, TWPUDC)*** *Let X^k and X^{k-1} be the HTWPUI from uncertain databases, and $X^{k-1} \subseteq X^k$. The TWU(X^{k-1}) $\geq$ TWU(X^k) and expSup(X^{k-1}) $\geq$ expSup(X^k).*

Proof: Let X^{k-1} be a (k-1)-itemset and its superset k-itemset is denoted as X^k. Since $X^{k-1} \subseteq X^k$, thus,

$$\begin{aligned} TWU(X^k) &= \sum_{X^k \subseteq T_q \wedge T_q \in D} tu(T_q). \\ &\leq \sum_{X^{k-1} \subseteq T_q \wedge T_q \in D} tu(T_q). \\ &= TWU(X^{k-1}). \end{aligned}$$

From **Theorem 1**, it can be found that *expSup*(X^{k-1}) $\geq$ *expSup*(X^k). Therefore, if X^k is a HTWPUI, any subset X^{k-1} is also a HTWPUI. □

Corollary 3. *If an itemset X^k is a HTWPUI, every subset X^{k-1} of X^k is a HTWPUI.*

Corollary 4. *If an itemset X^k is not a HTWPUI, no superset X^{k+1} of X^k is a HTWPUI.*

Theorem 3. ***(PHUIs $\subseteq$ HTWPUIs)*** *The transaction-weighted probability and utilization downward closure (TWPUDC) property ensures that PHUIs $\subseteq$ HTWPUIs, which indicates that if an itemset is not a HTWPUI, then none of its supersets will be PHUIs.*

Proof: $\forall$ $X \subseteq D$, X is an itemset; thus,

$$\begin{aligned} u(X) &= \sum_{X \subseteq T_q \wedge T_q \in D} u(X, T_q). \\ &\leq \sum_{X \subseteq T_q \wedge T_q \in D} tu(T_q). \\ &= TWU(X). \end{aligned}$$

According to **Theorem 2**, we can get *expSup*(X^{k-1}) $\geq$ *expSup*(X^k). Thus, if X is not an HTWPUI, none of its supersets are PHUIs. □

4.1.2 Detail of PHUI-apriori algorithm

The proposed PHUI-apriori algorithm has two phases: in the first phase, the HTWPUIs are found, and in the second phase, the PHUIs are derived with an additional database rescan. The TWPUDC property inherits the TWDC property of the two-phase model to keep the downward closure property, thus reducing the search space for finding PHUIs. Only the remaining HTWPUI^{k-1} will be used to generate the next C_k at each level. Based on the above definitions and theorems, detail of the proposed PHUI-apriori algorithm is described in PHUI-apriori algorithm.

Based on the designed TWPUDC property, **Theorem 3** ensures that the proposed PHUI-apriori algorithm can make sure that no supersets of small transaction-weighted probabilistic and utilization itemsets are in the preliminary candidate set (**correctness**) and can extract the complete PHUIs from the candidate HTWPUIs (**completeness**). Therefore, the results of the proposed PHUI-apriori algorithm are correct and complete.

The proposed PHUI-apriori algorithm first scans the database to find the TWU values and the probabilities of all 1-itemsets in the database (Lines 1 to 4). The set of HTWPUI^k (k is initially set as 1) is then produced (Lines 5 to 9) and will be further used to generate the next candidates C_{k+1} for discovering HTWPUI^{k+1} in a level-wise way (Lines 12 to 21). In this process, the original database has to be rescanned

Algorithm 1 PHUI-apriori

Input: D, an uncertain database; *ptable*, profit table; ε, minimum utility threshold; μ, minimum expected support threshold.
Output: the set of potential high-utility itemsets (PHUIs).

1: **for** each T_q in $D \wedge i_j$ in T_q **do**
2: calculate $TWU(i_j)$;
3: calculate $expSup(i_j)$.
4: **end for**
5: **for** each i_j in D **do**
6: **if** $TWU(i_j) \geq TU \times \varepsilon \wedge expSup(i_j) \geq |D| \times \mu$ **then**
7: $\text{HTWPUI}^1 \leftarrow i_j$.
8: **end if**
9: **end for**
10: set $k \leftarrow 2$.
11: set X as (k)-itemset.
12: **while** $HTWPUI^{k-1} \neq$ ***null*** **do**
13: $C_k \leftarrow$ Apriori_gen(HTWPUI^{k-1}).
14: **for** each k-itemset X in C_k **do**
15: scan D to find $TWU(X)$ and $expSup(X)$.
16: **if** $TWU(X) \geq TU \times \varepsilon \wedge expsup(X) \geq |D| \times \mu$ **then**
17: $\text{HTWPUI}^k \leftarrow X$.
18: **end if**
19: **end for**
20: $k \leftarrow k+1$.
21: **end while**
22: $\text{HTWPUIs} \leftarrow \text{HTWPUI}^k$
23: **for** each k-itemset X in HTWPUIs **do**
24: scan D to find $u(X)$.
25: **if** $u(X) \geq TU \times \varepsilon$ **then**
26: $\text{PHUI}^k \leftarrow X$.
27: **end if**
28: **end for**
29: $\text{PHUIs} \leftarrow \text{PHUI}^k$.
30: ***return*** PHUIs.

to find the HTWPUI^{k+1} (Line 15). The first phase of PHUI-apriori algorithm is terminated when no candidate is generated. An additional database rescan is required in the second phase to find the final PHUIs from the HTWPUIs (Lines 23 to 30).

4.1.3 An illustrated example of PHUI-apriori algorithm

In order to keep consistency, Tables 1 and 2 are used to illustrate the proposed PHUI-apriori algorithm as the example step-by-step. Assume the minimum utility threshold is also set at 25% and the minimum expected support threshold is also set at 15%. The minimum support count and the minimum expected support can be respectively calculated as (442 × 25%) = 110.5 and (10 × 15%) = 1.5. The PHUI-apriori algorithm first scans the database to find the TWU values and the probabilities of all 1-itemsets in the databases. The results are (*A*:303, 3.71; *B*:222, 3.71; *C*:336, 4.36; *D*:209, 3.41; *E*:283, 3.91) in which (*A*:303, 3.71) indicates that the $TWU(A)$ = 303 and *expSup*(*A*) = 3.71. In this example, all itemsets satisfies the above conditions and then put into the set of HTWPUI^1. Based on the designed PHUI-apriori algorithm, C_2 are then generated from HTWPUI^1. A database scan is required to find the TWU values and the *expSup* values of C_2 as (*AB*:107, 1.3; *AC*:259, 3.26; *AD*:166, 1.96; *AE*:181, 2.2; *BC*:116, 1.45; *BD*:87, 1.9; *BE*:209, 3.01; *CD*:122, 1.51; *CE*:190, 2.35; *DE*:74, 1.2). Among them, only the itemsets (*AC*, *AD*, *AE*, *BE*, *CD*, *CE*) satisfy the condition as $TWU(X) > 110.5$ and $expSup(X) > 1.5$; they are then put into the set of HTWPUI^2. The variable k is then set to 3. This process is repeated until no candidates are generated. The results are shown in Table 4.

TABLE 4: Derived HTWPUIs over an uncertain database

Itemset	TWU	expSup
(*A*)	303	3.71
(*B*)	222	3.71
(*C*)	336	4.36
(*D*)	209	3.41
(*E*)	283	3.91
(*AC*)	259	3.26
(*AD*)	166	1.96
(*AE*)	181	2.2
(*BE*)	209	3.01
(*CD*)	122	1.51
(*CE*)	190	2.35
(*ACD*)	122	1.51
(*ACE*)	137	1.75

After the first phase, the second phase is executed with an additional database scan to find the actual utility value of each remaining candidate. The results of PHUIs were shown in Table 3.

4.2 Proposed PHUI-list Algorithm

In the past, HUI-Miner [7] was proposed to efficiently mine HUIs. Experiments on HUI-Miner indicate that it has the best performance compared to the state-of-the-art algorithms of HUIM. In this section, an efficient PHUI-list algorithm is proposed to improve the performance of the PHUI-apriori algorithm for efficiently mining PHUIs. A probability-utility (PU)-list is designed to directly mine PHUIs without an additional database rescan compared to the PHUI-apriori algorithm. Detail of the presented PU-list structure, the enumeration tree for the search space, and the proposed PHUI-list algorithm are described below.

4.2.1 PU-list structure

The PU-list structure inherits the utility-list property to keep more related information from transactions for directly mining PHUIs. Each entry for a 1-itemset X of the PU-list consists of the corresponding TID number of X (***TID***), the probability of X in T_q (***Prob***), the utility of X in T_q (***Iutility***), and the remaining utility of X in T_q (***Rutility***).

Definition 14. An entry of X consists of four fields, including the *TIDs* of X in T_q ($X \subseteq T_q \in D$), the probabilities of X in T_q (***Prob***), the utilities of X in T_q (***Iutility***), and the remaining utilities of X in T_q (***Rutility***), in which ***Rutility*** is defined as $\sum_{i \in T_q \wedge i \notin X} u(i, T_q)$.

The construction procedure of the PU-list is recursively processed if it is necessary to determine the k-itemsets in the search space. Detail is shown below.

Algorithm 2 PU-list construction procedure

Input: X, an itemset; $X.PUL$ is the PU-list of X; $X_{ab}.PUL$, $X_a.PUL$, $X_b.PUL$, $X_a \subseteq X$ and $X_b \subseteq X$, $X_a \neq X_b$.
Output: $X_{ab}.PUL$.

set $X_{ab}.PUL \leftarrow$ ***null***.
for each element $E_a \in X_a$ **do**
    if $\exists E_a \in X_b.PUL \wedge E_a.TID := E_b.TID$ **then**
        search $E \in X.PUL \wedge E.TID := E_a.TID$.
        $E_{ab} \leftarrow <E_a.TID, E_a.Prob, E_a.Iutility + E_b.Iutility - E.Iutility, E_b.Rutility>$.
    else
        $E_{ab} \leftarrow <E_a.TID, E_a.Prob, E_a.Iutility + E_b.Iutility, E_b.Rutility>$.
    end if
    $X_{ab}.PUL \leftarrow E_{ab}$.
end for
return $X_{ab}.PUL$.

Note that it is necessary to initially construct the PU-list of the HTWPUI[1] as the input for the later recursive process. Since the 1-items of HTWPUI[1] are (A:303, 3.71; B:222, 3.71; C:336, 4.36; D:209, 3.41; E:283, 3.91) in which (A:303, 3.71) indicates that the item (A) has its $TWU(A) = 303$ and $expSup(A)$ = 3.71. The PU-list is constructed in ascending order of their TWU values as ($D < B < E < A <$ C), which is shown in Fig. 1.

{D}			
2	0.7	12	1
5	0.75	12	18
6	0.7	30	32
7	0.45	24	20
9	0.81	12	48

{B}			
2	0.7	1	0
3	0.85	2	61
5	0.75	3	15
7	0.45	1	19
8	0.36	4	15
10	0.6	2	51

{E}			
1	0.9	30	44
3	0.85	45	16
5	0.75	15	0
7	0.45	15	4
8	0.36	15	0
10	0.6	15	36
TID	*Prob*	*Iutility*	*Rutility*

{A}			
1	0.9	8	36
3	0.85	4	12
6	0.7	8	24
7	0.45	4	0
9	0.81	12	36

{C}			
1	0.9	36	0
3	0.85	12	0
4	0.5	24	0
6	0.7	24	0
9	0.81	36	0
10	0.6	36	0

Fig. 1: Constructed PU-list structure of HTWPUI[1].

Definition 15. The $X.Iutility.SUM$ is the sum of the utilities of an itemset X in D, which can be defined as:

$$X.Iutility.SUM = \sum_{X \subseteq T_q \wedge T_q \in D} (X.Iutility).$$

Definition 16. The $X.Rutility.SUM$ is the sum of the remaining utilities of an itemset X in D, which can be defined as:

$$X.Rutility.SUM = \sum_{X \subseteq T_q \wedge T_q \in D} (X.Rutility).$$

Take an item (A) from Fig. 1 as an example to illustrate the process. The item (A) exists in TID {1, 3, 6, 7, 9}, its $A.Iutility.SUM$ is calculated as (8 + 4 + 8 + 4 + 12) = 36, and $A.Rutility.SUM$ is calculated as (36 + 12 + 24 + 0 + 36) = 108. Take an item (AE) as an example to illustrate the process. The item (AE) exists in TID = {1, 3, 7}, its $AE.Iutility.SUM$ = $A.Iutility.SUM$ + $E.Iutility.SUM$ = (8 + 4 + 8) + (30 + 45 + 15) = 110, and $AE.Rutility.SUM$ = $A.Rutility.SUM$ + $E.Rutility.SUM$ = (36 + 12 + 0) + (44 + 16 + 4) = 112.

4.2.2 An enumeration tree

Based on the PU-list structure, the search space of the proposed PHUI-list algorithm can be represented as the enumeration tree by the TWU values of the 1-items in the set of HTWPUI[1] in ascending order. The process is constructed as follows. Each node in the tree is represented as an itemset, which is the extension (superset) of its parent node. The illustrated enumeration tree of the given example is shown in Fig. 2.

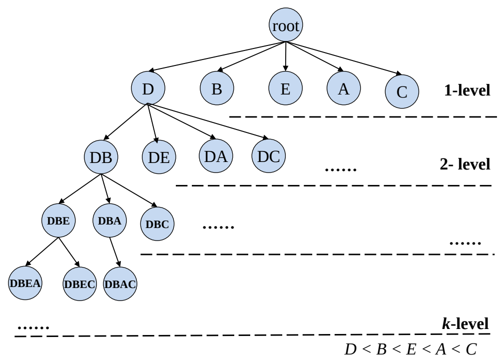

Fig. 2: An enumeration tree.

The PHUI-list algorithm is based on depth-first search strategy in which the enumeration tree is travelled to search for the possible itemsets. Each node is determined by the summation of ***Prob***, ***Iutility***, and ***Rutility*** as the early pruning strategy to decide whether the supersets of the processed node need to be determined. If the processed node satisfies the following two conditions: (1) the summation of ***Iutility*** and ***Rutility*** of the current processed node is larger than or equals to the minimum utility count ($\varepsilon \times TU$), and (2) the *expSup* (***Prob***) of the processed node is larger than or equal to the minimum expected support count ($\mu \times |D|$), the supersets of the processed node will be generated and determined (the pruning strategy will be described later in more details). The actual utility of the processed node can be determined by its ***TID***, ***Iutility***, and the ***Prob*** for discovering PHUIs without an additional database scan. Based on the constructed enumeration tree, the following lemmas can be obtained.

Lemma 1. The search space of the proposed PHUI-list algorithm can be represented as the enumeration tree by the TWU values of 1-items in the set of HTWPUI[1] in ascending order.

Proof: From the constructed enumeration tree in Fig. 2, the top-down and breath-search mechanisms are performed to traverse the tree nodes from 1-level to k-level, which is performed in the similar way as Apriori-like mechanism. Thus, the enumeration tree is represented as a complete search space of the proposed PHUI-list algorithm. □

4.2.3 Early pruning strategies

Based on the above definitions, two early pruning strategies are used to find the compressed search space according to the TWPUDC property. Thus, a great number of unpromising candidates can be efficiently pruned, which can significantly reduce the search space of the proposed PHUI-list algorithm. Based on the constructed enumeration tree, the lemmas can be obtained as follows.

Lemma 2. For each node in the enumeration tree, the sum of its probability values in the PU-list structure is no less than the sum of probability values of its any child node.

Proof: Assume a node in the enumeration tree is X^{k-1}, any child node (superset) can be denoted as X^k. According to **Theorem 1**, **2**, and **3**, this lemma can be correctly obtained. □

Pruning Strategy 1. *For any node X in the enumeration tree, if the the sum of probabilities of a node X in the constructed PU-list is less than the minimum expected support, any of its child node is not a PHUI.*

Rationale 1. According to **Lemma 2**, it can be obtained that if the probability of a node is less than the minimum expected support count, it can be regarded as an unpromising itemset of PHUI. Thus, any of its child node (superset) is also an unpromising itemset, which can be directly pruned in the search space.

Lemma 3. For each node X^{k-1} in the enumeration tree, the sum of its $X^{k-1}.Iutility.SUM$ and $X^{k-1}.Rutility.SUM$ in the PU-list structure is no less than the sum of utilities of its any child node.

Proof: For a node X^{k-1} in the enumeration tree, any of its child node is denoted as X^k. The (X^k - X^{k-1})⇒(X^k/X^{k-1}) indicates that X^k is an extension (child node) of X^{k-1}, thus,

∀ transaction $T_q \supseteq X^k$:
$\because X^{k-1} \subset X^k \subseteq T_q \Rightarrow (X^k/X^{k-1}) \subseteq (T_q/X^{k-1})$.
$\therefore$ in T_q,

$$\begin{aligned}X^k.Iutility &= X^{k-1}.Iutility + (X^k/X^{k-1}).Iutility\\ &= X^{k-1}.Iutility + \sum_{i\in(X^k/X^{k-1})} i.Iutility\\ &\leq X^{k-1}.Iutility + \sum_{i\in(T_q/X^{k-1})} i.Iutility\\ &= X^{k-1}.Iutility + X^{k-1}.Rutility\end{aligned}$$

$\therefore$ in each T_q,
$X^k.Iutility \leq X^{k-1}.Iutility + X^{k-1}.Rutility$.
$\because X^{k-1} \subset Xk \Rightarrow TIDs(X^k) \subseteq TIDs(X^{k-1})$.
$\therefore$ in D,

$$\begin{aligned}X^k.Iutility.SUM &= \sum_{T_q\in TIDs(X^k)} X^k.Iutility\\ &\leq \sum_{T_q\in TIDs(X^k)} (X^{k-1}.Iutility + X^{k-1}.Rutility)\\ &\leq \sum_{T_q\in TIDs(X^{k-1})} (X^{k-1}.Iutility + X^{k-1}.Rutility)\\ &= \sum_{T_q\in TIDs(X^{k-1})} X^{k-1}.Iutility + \sum_{T_q\in TIDs(X^{k-1})} X^{k-1}.Rutility\\ &= X^{k-1}.Iutility.SUM + X^{k-1}.Rutility.SUM.\end{aligned}$$

Thus, the sum of utilities of X^k is less than or equals to the sum of $X.Iutility.SUM$ and $X.Rutility.SUM$ of X^{k-1}. □

Pruning Strategy 2. *For any node X in the enumeration tree, if the sum of $X.Iutility.SUM$ and $X.Rutility.SUM$ in the constructed PU-list is less than the minimum utility count, any of its child node is not a PHUI.*

Rationale 2. According to **Lemma 3**, it can be obtained that if the sum of total utilities and remaining utilities of a node is less than the minimum utility count, it can be regarded as an unpromising itemset of PHUI. Thus, any of its child node (superset) is also unpromising itemset, which can be directly pruned in the search space. Thus, when the sum of all the utilities itemsets are being estimated, those utilities of unpromising itemsets and its supersets can be regarded as irrelevant and be pruned directly.

Lemma 4. The number of traversal nodes in the enumeration tree by the PHUI-list algorithm with two pruning strategies is smaller than the number of candidates generated by the PHUI-apriori algorithm based on its designed TWPUDC property, which indicates that the search space of the former algorithm can be compressed compared to the later one.

Proof: According to **Lemma 2** and **3**, it can be found that the remaining utilities of PU-list structure is smaller than the TU value. Thus, the sum of $X.Iutility.SUM$ and $X.Rutility.SUM$ is smaller than its TWU value. Based on the PU-list structure and two pruning strategies, a tighter upper bound can be obtained to efficiently prune the unpromising itemsets and reduce the search space compared to the TWPUDC property. □

Theorem 4. *The discovered PHUIs by the PHUI-list algorithm is correct and complete.*

Proof: According to **Lemma 1**, **2** and **3**, the designed PHUI-list algorithm can ensure that any unpromising itemset will be discarded (**completeness**) and the related information can be exactly obtained from the PU-list structure (**correctness**). □

4.2.4 Detail of PHUI-list algorithm

Based on the two proposed pruning strategies, the designed PHUI-list algorithm can prune the itemsets with lower expected support count and utility count early, without constructing their PU-list structures of supersets, which can effectively reduce both the computations of join operations and the search space from the enumeration tree. Based on the above definitions and properties, the pseudo-code of the proposed PHUI-list algorithm is described in Algorithm 3.

Algorithm 3 PHUI-list

Input: D, uncertain databases; *ptable*, a profit table; ε, minimum utility threshold; μ, minimum expected support threshold.
Output: The set of potential high-utility itemsets (PHUIs).

```
1: scan D to find HTWPUI^1.
2: for each T_q in HTWPUI^1 do
3:     for each X in T_q do
4:         X.PUL ←{T_q, Prob, Iutility, Rutility}.
5:     end for
6: end for
7: D.PUL ← ⋃ X.PUL.
8: for each X in D.PUL do
9:     if X.Iutility.SUM ≥ TU × ε ∧ expSup(X) ≥ |D| × μ
   then
10:        PHUIs← X.
11:    end if
12:    if X.Iutility.SUM + X.Rutility.SUM ≥ TU × ε ∧
   expSup(X) ≥ |D| × μ then
13:        extendPULs ← null.
14:        for each z after y in D.PUL do
15:            extendPULs ← extendPULs +
               Construct(X.PUL,y,z).
16:        end for
17:        call PHUI-list(X,extendPULs,ε,μ).
18:    end if
19: end for
20: return PHUIs.
```

The proposed PHUI-list algorithm first scans the original uncertain databases to find the potential high-transaction-weight utilization 1-itemsets (HTWPUI[1]) (Line 1) and also to construct the PU-list of each 1-itemset in HTWPUI[1] (Lines 2 to 6). The probability-utility list ($D.PUL$) for all 1-extensions of X is recursively processed (Lines 8 to 19) by using a depth-first search procedure. Each 1-itemset X is determined to directly produce the PHUIs (Lines 9 to 11). Two pruning strategies are then applied to further determine whether its supersets satisfy the designed conditions for executing the later depth-first search (Lines 12 to 18). The construction process **Construct**$(X.PUL, y, z)$ is then executed to construct $extendPULs$ for recursively processing the designed algorithm to mine PHUIs (Lines 8 to 19). Based on the designed PU-list structure, the PHUI-list algorithm can thus directly mine the complete PHUIs from uncertain databases without candidate generation.

4.2.5 An illustrated example of PHUI-list algorithm

In order to keep consistency, the parameters used to illustrate the PHUI-list algorithm are the same as Section 4.1.3. The PHUI-list algorithm first scans uncertain databases to extract the necessary information of ***TID***, ***Pro***, ***Iutility*** and ***Rutility*** for constructing the PU-list structures of all 1-items. The satisfied HTWPUI[1] are first discovered shown in Fig. 1. The item (D) is first processed in the PU-list structure. In this example, an item (D) does not satisfy one of the conditions as $TWU(D) = 90 < 110.5$; an item (D) is regarded as the unpromising item. The $D.Iutility.SUM$ + $D.Rutility.SUM$ = (90 + 1 + 18 + 32 + 20 + 48) = 209, which is larger than the minimum utility count. The child nodes (supersets) of (D) may be a PHUI; the depth-first search is still performed for its child nodes. The PU-list structure for 2-itemsets with its prefix item (D) is shown in Fig. 3.

{DB}			
2	0.7	13	0
5	0.75	15	15
7	0.45	25	19

{DE}			
5	0.75	27	0
7	0.45	39	4

{DA}			
6	0.7	38	24
7	0.45	28	0
9	0.81	24	36

{DC}			
6	0.7	54	0
9	0.81	48	0

TID ***Prob*** ***Iutility*** ***Rutility***

Fig. 3: Constructed PU-list structure with the prefix item (D).

The first child node (DB) in Fig. 3 is first determined. In this example, $DB.Iutility.SUM$ = (13 + 15 + 25) = 53 < 110.5, and $expSup(DB)$ = (0.7 + 0.75 + 0.45) = 1.9 > 1.5. Thus, an itemset (DB) is not a PHUI. Since $DB.Iutility.SUM$ + $DB.Rutility.SUM$ is calculated as (53) + (0 + 15 + 19) = 87 < 110.5, the depth-first search of (DB) is terminated. The next node of (DE) is then processed in the same way. After all nodes are determined and visited, the complete PHUIs can be directly discovered by the PHUI-list algorithm without an additional database scan. The results were shown in Table 3.

5 Experimental Results

Experiments for mining PHUIs over uncertain datasets were conducted to evaluate the performance of two proposed algorithms in terms of runtime, memory consumption, the number of discovered patterns, and scalability. The algorithms in the experiments were implemented in Java language and performed on a personal computer with an Intel Core i5-3460 dual-core processor and 4 GB of RAM and running the 32-bit Microsoft Windows 7 operating system. Experiments under varied minimum utility thresholds (MUs) and varied minimum expected support thresholds (MEs) are discussed below.

5.1 Tested Datasets

Both real-life and synthetic datasets were used in the experiments. Three real-life datasets, namely foodmart [26], accidents [27], and retail [27] datasets, as well as one synthetic dataset, T10I4D100K [4], were used in the experiments to evaluate the performance of two proposed algorithms. Both the quantity (internal) and profit (external) values are assigned to the items in the accident, retail, and T10I4D100K datasets by Liu's simulation model [8] but not to those in the foodmart dataset. In addition, due to the tuple uncertainty property, each transaction in these datasets is assigned a unique probability value in the range of 0.5 to 1.0. The characteristics of used datasets are shown in Table 5.

TABLE 5: Characteristics of used datasets

Database	#|D|	#|I|	AvgLen	MaxLen	Type
foodmart	21,556	1,559	4	11	sparse
retail	88,162	16,470	10.3	76	sparse
accidents	340,183	468	33.8	51	dense
T10I4D100K	100,000	870	10.1	29	sparse

5.2 Runtime

The runtime of the two proposed algorithms under varied MUs with a fixed ME value are compared and shown in Fig. 4. In addition, the runtime of the two proposed algorithms under varied MEs with a fixed MU value are then compared and shown in Fig. 5.

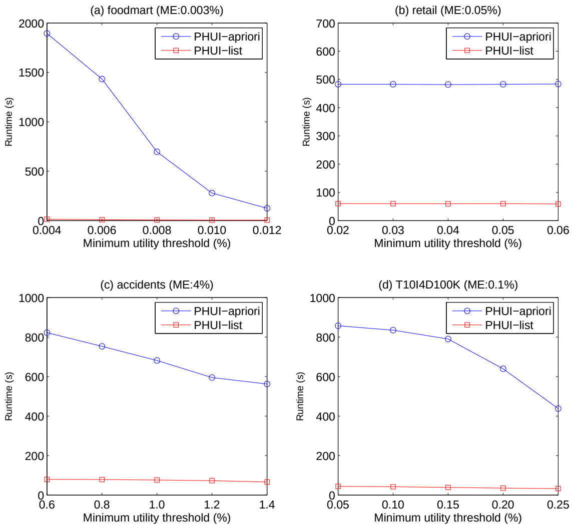

Fig. 4: Runtime of two algorithms under varied MUs with a fixed ME.

From Fig. 4, we can observe that the proposed PHUI-list algorithm has better performance than the naive PHUI-apriori algorithm, which indicated that the generate-and-test approach has worse performance than the PU-list structure. For example, for the accidents dataset in Fig. 4(c), the ME was set at 4% and the varied MUs were set from 0.6 to 1.4%, with 0.2% increment each time. The runtime of PHUI-apriori algorithm is dramatically decreased from 822 to 562 seconds, while that of PHUI-list algorithm changed steadily from 80 to 66 seconds. The reason is that when the MUs are set quite low, longer patterns of HTWPUIs are discovered first before the PHUIs are found by the designed PHUI-apriori algorithm, and thus more computations are needed to process both the generate-and-test mechanism and the two-phase model, especially in a dense dataset. The PHUI-list algorithm directly determines the PHUIs from

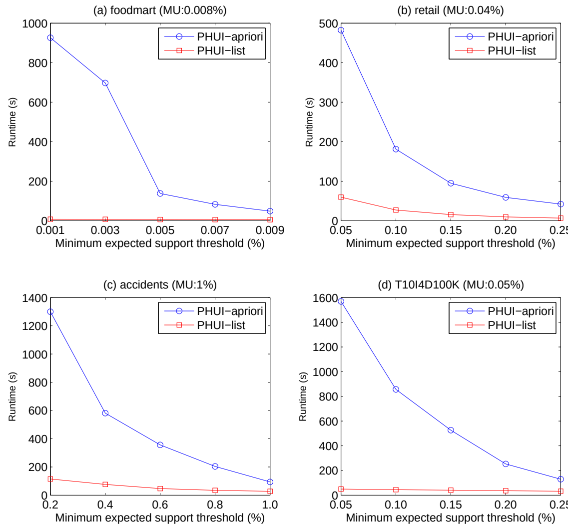

Fig. 5: Runtime of two algorithms under varied MEs with a fixed MU.

the enumeration tree without candidate generation, it can effectively avoid the time-consuming process of database scan. Moreover, the PHUI-list algorithm applied two pruning strategies to effectively prune the unpromising items early. Based on the designed PU-list structure, the runtime to discover PHUIs can be greatly reduced.

From Fig. 5, it can also be observed that PHUI-list algorithm outperforms PHUI-apriori algorithm under the varied MEs in four datasets. Specifically, the runtime of PHUI-apriori algorithm is sharply decreased along with the increasing of MEs, while the runtime of the PHUI-list algorithm is steadily decreased. The result is reasonable since when ME is set higher, fewer candidates are generated for later processing to mine the PHUIs. Although the PHUI-apriori algorithm uses the TWPUDC property to reduce the search space, it is still performed in a level-wise way to generate and test candidates for mining PHUIs. In this situation, huge numbers of candidates are generated but PHUIs are produced only rarely. When ME is set higher, many redundant unpromising candidates are pruned early, and thus the search space and runtime of the PHUI-apriori algorithm are sharply decreased.

5.3 Pattern Analysis

When the probability of each transaction is set at 1.0 over uncertain datasets, it indicates that each transaction has 100% existential probability; the uncertain dataset turns into a precise quantitative to mine HUIs which is the same as traditional way to mine HUIs. The state-of-the-art HUI-miner algorithm [7] is thus used to mine HUIs in the precise dataset without probability values compared to the designed PHUI-apriori and PHUI-list algorithms. The results of pattern analysis for HUIs and PHUIs under varied MUs with a fixed ME and under varied MEs with a fixed MU are respectively shown in Fig. 6 and Fig. 7.

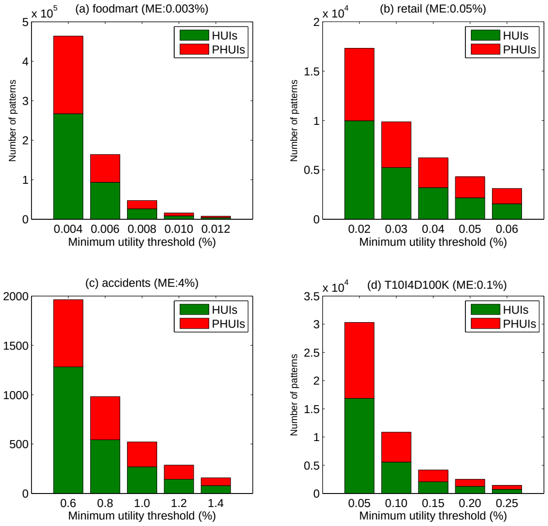

Fig. 6: Number of HUIs and PHUIs under varied MUs with a fixed ME.

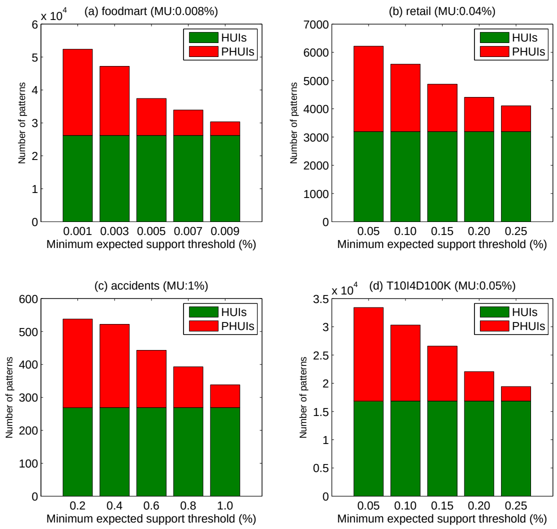

Fig. 7: Number of HUIs and PHUIs under varied MEs with a fixed MU.

From Fig. 6, it can be found that the number of PHUIs is always smaller than the number of HUIs under varied MUs in both sparse and dense datasets, which indicates that numerous HUIs are discovered but few PHUIs are produced by considering the probability value of each transaction in uncertain datasets. In real-world applications, numerous discovered HUIs may not be the patterns of interest for helping the manager or retailer to make efficient decisions without considering the probability factor. This situation happens regularly when MU is set lower. When the MU is set higher, fewer PHUIs are produced by the designed approaches compared to the number of HUIs discovered by HUI-Miner. This is because the proposed algorithms are used to discover PHUIs based on both the high-utility and the high-probability constraint, while HUI-Miner is used to discover the HUIs based on only the high-utility constraint. In particular, the patterns of PHUIs are fewer and more valuable compared to those general HUIs since the variety of item probability is concerned. From the experimental results, it can also be seen that both the number of HUIs and the number of PHUIs are decreased as along with the increasing of MUs.

From Fig. 7, we can see that fewer PHUIs are discovered by the proposed algorithms compared to the number of discovered HUIs by HUI-Miner under varied MEs in four uncertain datasets. The number of discovered HUIs remains steady as the MEs increase. This is reasonable since HUI-Miner only considers the high-utility constraint for discovering HUIs. The number of PHUIs decreases dramatically as the increasing of MEs. The reason is the same as the one described in Fig. 6, since the proposed algorithms are based on two constraints for mining PHUIs. In particular, the discovered PHUIs can be considered as valuable patterns compared to the traditional approaches for discovering HUIs, since the probability factor generally occurs in a real-life situations.

5.4 Memory Consumption

The JAVA API is used to evaluate the memory consumption of two proposed algorithms. The results under varied MUs with a fixed ME and under varied MEs with a fixed MU are respectively shown in Fig. 8 and Fig. 9.

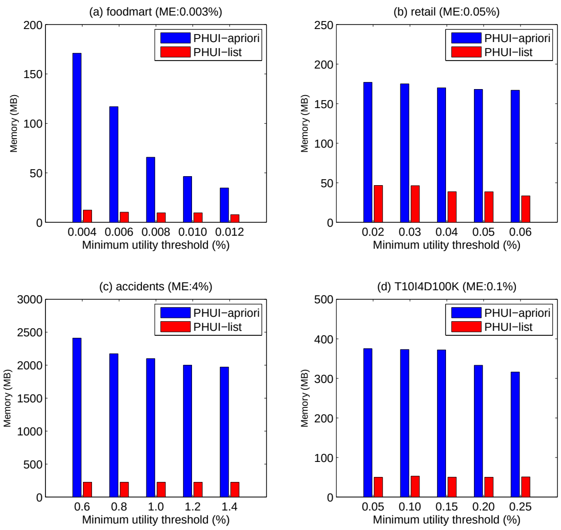

Fig. 8: Memory consumption under varied MUs with a fixed ME.

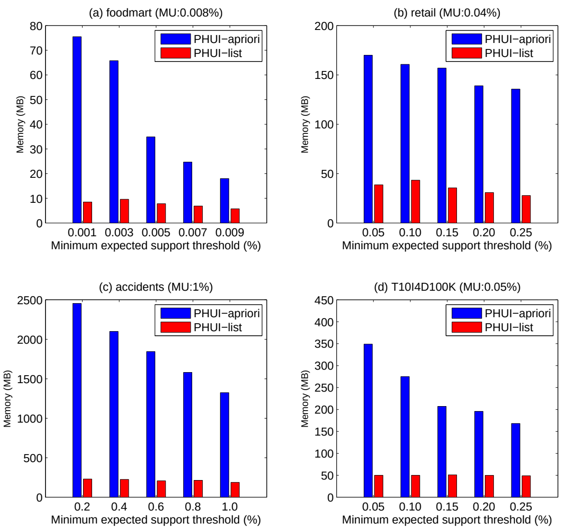

Fig. 9: Memory consumption under varied MEs with a fixed MU.

From Fig. 8 and Fig. 9, it can be clearly seen that the proposed PHUI-list algorithm requires less memory compared to the PHUI-apriori algorithm under varied MUs with a fixed ME and under varied MEs with a fixed ME for four datasets. Specially, the PHUI-list algorithm requires nearly constant memory under varied parameters in four datasets. The PHUI-apriori algorithm requires more memory when the MU or ME is set lower compared to the PHUI-list algorithm. For example, PHUI-apriori algorithm requires 2100 MB of memory on average, but PHUI-list algorithm only requires 225 MB of memory on average, as can be observed in Fig. 8(c). It is also easy to find that the performance gap between PHUI-apriori and PHUI-list algorithms gets smaller with the increasing of MU and a fixed ME or with the increasing of ME and a fixed MU. For instance, when ME was set at 0.001% and MU was set at 0.008%, the PHUI-apriori and the PHUI-list algorithms required 76.5 and 9 MB of memory, respectively, which can be observed from the foodmart dataset in Fig. 9(a). When ME was set at 0.009% and MU at 0.008%, the PHUI-apriori and PHUI-list algorithms required 20 and 5.5 MB of memory, respectively, as shown in Fig. 9(a). Since the unpromising candidates can be early pruned, less memory is required to keep the remaining candidates for the later mining process. Generally, the PHUI-list algorithm has better performance compared to the PHUI-apriori algorithm under varied parameters for four datasets.

5.5 Scalability

The scalability of the two proposed algorithms is compared on synthetic dataset T10I4N4KD$|X|K$ for increases in the dataset size, which is set from 100K to 500K, with increments of 100K. The results under varied MEs and MUs are shown in Fig. 10.

From Fig. 10, it shows that the two proposed algorithms have good scalability under varied dataset sizes in terms of runtime, memory consumption, and number of patterns. From the results of Fig. 10(a), Fig. 10(b), Fig. 10(d), and Fig. 10(e), it can be observed that the two designed algorithms required more computations and memory to find the PHUIs as the dataset size increased. The proposed PHUI-list algorithm always has better results than the PHUI-apriori algorithm. An interesting observation is that the runtime and memory consumption of the proposed PHUI-list algorithm is steadily increased but those of PHUI-apriori algorithm is sharply increased along with the increasing of dataset size. For example, PHUI-list algorithm was almost two orders of magnitude faster than the PHUI-apriori algorithm when ME was set at 0.05% and ME was set at 0.1% as the dataset size is increased from 100K to 500K, which can be observed from Fig. 10(a). Moreover, fewer PHUIs are produced by the proposed algorithms compared to the number of HUIs found by the HUI-Miner algorithm, as can be observed in Fig. 10(c) and Fig. 10(f). From the observed results of the scalability experiments, it can be concluded that the two proposed algorithms have good scalability and the proposed PHUI-list algorithm has better performance in terms of runtime, memory consumption, and scalability.

6 Conclusions and Future Works

In this paper, a novel framework is proposed for mining potential high-utility itemsets (PHUIs) from uncertain databases. The apriori-based potential high-utility itemsets (PHUI-apriori) algorithm and the list-based potential high-utility itemsets (PHUI-list) algorithm are respectively proposed to consider the mining of not only high-utility itemsets but also high probability itemsets from uncertain databases. The designed PHUI-apriori algorithm is based on the level-wise approach to generate and test candidates for mining PHUIs. The second PHUI-list algorithm is then developed to improve the performance compared to the PHUI-apriori algorithm based on the designed PU-list structure and the enumeration tree for directly mining PHUIs without candidate generation. Substantial experiments were conducted to show the performance of two proposed algorithms in terms of runtime, patterns analysis, memory consumption, and scalability.

Since this is the first work for mining PHUIs from uncertain databases, further research issues including dynamic data mining, stream mining, and top-k patterns mining can also be studied. Besides, designing more efficient and condensed structure based on different uncertain models for mining the desired information is also another critical issue in the nearly future.

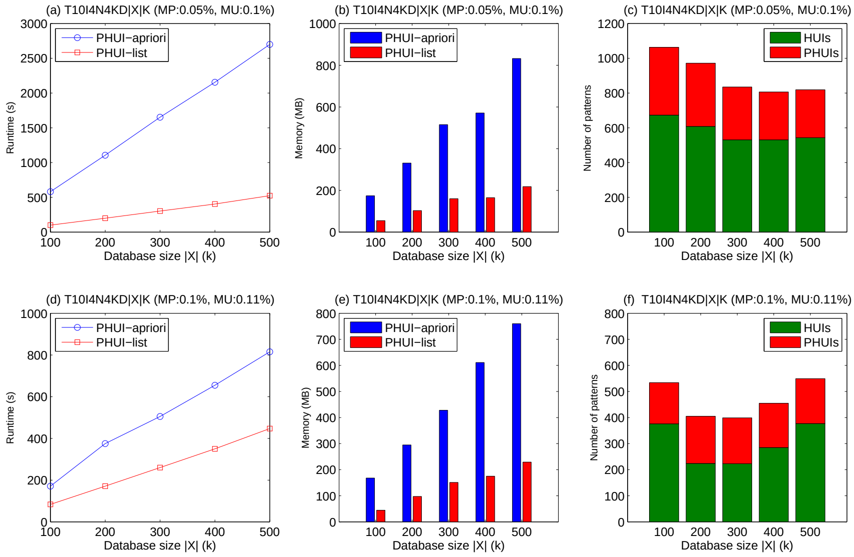

Fig. 10: Scalability results of two proposed algorithms.

REFERENCES

[1] R. Agrawal, T. Imielinski, and S. Arun, "Mining association rules between sets of items in large database," *ACM SIGMOD International Conference on Management of Data*, pp. 207–216, 1993.

[2] M. S. Chen, J. Han, and P. S. Yu, "Data mining: an overview from a database perspective," *IEEE Transactions on Knowledge and Data Engineering*, vol. 8, no. 6, pp. 866–883, 1996.

[3] X. Wu, V. Kumar, J. R. Quinlan, J. Ghosh, Q. Yang, H. Motoda, G. J. McLachlan, A. Ng, and B. Liu, "Top 10 algorithms in data mining," *Knowledge and Information Systems*, vol. 14, pp. 1–37, 2006.

[4] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules in large databases," pp. 487–499, 1994.

[5] J. Han, J. Pei, Y. Yin, and R. Mao, "Mining frequent patterns without candidate generation: A frequent-pattern tree approach," *Data Mining and Knowledge Discovery*, vol. 8, pp. 53–87, 2004.

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

[7] M. Liu and J. Qu, "Mining high utility itemsets without candidate generation," *ACM International Conference on Information and Knowledge Management*, pp. 55–64, 2012.

[8] Y. Liu, W. K. Liao, and A. Choudhary, "A two-phase algorithm for fast discovery of high utility itemsets," *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp. 689–695, 2005.

[9] F. V. Philippe, C. W. Wu, S. Zida, and V. S. Tseng, "Fhm: Faster high-utility itemset mining using estimated utility co-occurrence pruning," *Foundations of Intelligent Systems*, vol. 8502, pp. 83–92, 2014.

[10] V. S. Tseng, B. E. Shie, C. W. Wu, and P. S. Yu, "Efficient algorithms for mining high utility itemsets from transactional databases," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, pp. 1772–1786, 2013.

[11] H. Yao, H. J. Hamilton, and C. J. Butz, "A foundational approach to mining itemset utilities from databases," *SIAM International Conference on Data Mining*, pp. 211–225, 2004.

[12] C. C. Aggarwal, Y. Li, J. Wang, and J. Wang, "Frequent pattern mining with uncertain data," *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 29–38, 2009.

[13] C. C. Aggarwal and P. S. Yu, "A survey of uncertain data algorithms and applications," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 5, pp. 609–623, 2009.

[14] C. K. Chui, B. Kao, and E. Hung, "Mining frequent itemsets from uncertain data," *Advances in Knowledge Discovery and Data Mining*, pp. 47–58, 2007.

[15] C. K. S. Leung and B. Hao, "Mining of frequent itemsets from streams of uncertain data," *IEEE International Conference on Data Engineering*, pp. 1663–1670, 2009.

[16] C. W. Lin and T. P. Hong, "A new mining approach for uncertain databases using cufp trees," *Expert Systems with Applications*, vol. 39, no. 4, pp. 4084–4093, 2012.

[17] C. Liu, L. Chen, and C. Zhang, "Summarizing probabilistic frequent patterns: A fast approach," *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 527–535, 2013.

[18] L. Wang, D. L. Cheung, R. Cheng, S. D. Lee, and

X. S. Yang, “Efficient mining of frequent item sets on large uncertain databases,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 24(12), pp. 2170–2183, 2012.

[19] C. C. Aggarwal, “Managing and mining uncertain data,” *Managing and Mining Uncertain Data*, vol. 35, 2010.

[20] T. Bernecker, H. P. Kriegel, M. Renz, F. Verhein, and A. Zuefl, “Probabilistic frequent itemset mining in uncertain databases,” *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 119–128, 2009.

[21] L. Sun, R. Cheng, D. W. Cheung, and J. Cheng, “Mining uncertain data with probabilistic guarantees,” *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 273–282, 2010.

[22] C. K. S. Leung, M. A. F. Mateo, and D. A. Brajczuk, “A tree-based approach for frequent pattern mining from uncertain data,” *Advances in Knowledge Discovery and Data Mining*, pp. 653–661, 2008.

[23] Y. Tong, L. Chen, Y. Cheng, and P. S. Yu, “Mining frequent itemsets over uncertain databases,” *Proceedings of the VLDB Endowment*, vol. 5, no. 11, pp. 1650–1661, 2012.

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

[25] V. S. Tseng, C. W. Wu, B. E. Shie, and P. S. Yu, “Up-growth: an efficient algorithm for high utility itemset mining,” *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 253–262, 2010.

[26] Microsoft, “Example database foodmart of microsoft analysis services.” [Online]. Available: http://msdn.microsoft.com/en-us/library/aa217032(SQL.80).aspx

[27] “Frequent itemset mining dataset repository,” 2012. [Online]. Available: http://fimi.ua.ac.be/data/

Jerry Chun-Wei Lin received Ph.D. degree in Dept. of Computer Science and Information Engineering in 2010 from National Cheng Kung University, Tainan, Taiwan. He is currently working as an assistant professor at School of Computer Science and Technology, Harbin Institute of Technology Shenzhen Graduate School, China. He has published around 100 research papers in referred journals and international conferences. His interests include data mining, soft computing, privacy preserving data mining and security, social network and cloud computing.

Wensheng Gan is s a master student studying at School of Computer Science and Technology, Harbin Institute of Technology Shenzhen Graduate School, China. His research interests include data mining and cloud computing

Tzung-Pei Hong received his B.S. degree in chemical engineering from National Taiwan University in 1985, and his Ph.D. degree in computer science and information engineering from National Chiao-Tung University in 1992. He was in charge of the whole computerization and library planning for National University of Kaohsiung in Preparation from 1997 to 2000 and served as the first director of the library and computer center in National University of Kaohsiung from 2000 to 2001, as the Dean of Academic Affairs from 2003 to 2006, as the Administrative Vice President from 2007 to 2008, and as the Academic Vice President in 2010. He is currently a distinguished professor at the Department of Computer Science and Information Engineering and at the Department of Electrical Engineering. He has published more than 400 research papers in international/national journals and conferences and has planned more than fifty information systems. He is also the board member of more than forty journals and the program committee member of more than three hundred conferences. His current research interests include knowledge engineering, data mining, soft computing, management information systems, and www applications. Dr. Hong is a member of the Association for Computing Machinery, the IEEE, the Chinese Fuzzy Systems Association, the Taiwanese Association for Artificial Intelligence, and the Institute of Information and Computing Machinery.

Vincent S. Tseng holds the position of Distinguished Professor at Department of Computer Science and Information Engineering at National Cheng Kung University (NCKU), Taiwan. Currently he also serves as the chair for IEEE CIS Tainan Chapter. Before Dr. Tseng joined NCKU in 1999, he was a postdoctoral research fellow in Computer Science Division of University of California at Berkeley, U.S.A. He served as the president of Taiwanese Association for Artificial Intelligence during 2011-2012 and acted as the director for Institute of Medical Informatics of NCKU during August 2008 and July 2011. During February 2004 and July 2007, he had also served as the director for Informatics Center in National Cheng Kung University Hospital. Dr. Tseng received his Ph.D. degree with major in computer science from National Chiao Tung University, Taiwan, in 1997. He has a wide variety of research interests covering data mining, biomedical informatics, multimedia databases, mobile and Web technologies. He has published around 300 research papers in referred journals and international conferences as well as 15 patents held. He has been on the editorial board of a number of journals including IEEE Transactions on Knowledge and Data Engineering, ACM Transactions on Knowledge Discovery from Data, IEEE Journal of Biomedical and Health Informatics, International Journal of Data Mining and Bioinformatics, Journal of Information Science and Engineering, and Taiwanese Journal of Medical Informatics (associate editor-in-chief). He has also served as chairs/program committee members for a number of premier international conferences related to data mining and biomedical informatics, including KDD, ICDM, SDM, BIBM, BCB, PAKDD, CIKM, IJCAI, etc. He was also the program committee chair for PAKDD2014.