Knowledge-Based Systems
Manuscript Draft

Manuscript Number: KNOSYS-D-15-00796

Title: Efficient Algorithms for Mining High-Utility Itemsets with Uncertain Databases

Article Type: Full Length Article

Keywords: High-utility itemsets mining; uncertain database; probabilistic-based; upper-bound; PU-list structure.

Efficient Algorithms for Mining High-Utility Itemsets with Uncertain Databases

Jerry Chun-Wei Lin[1,2*], Wensheng Gan[1], Philippe Fournier-Viger[3], Tzung-Pei Hong[4,5], and Vincent S. Tseng[6]
[1]Innovative Information Industry Research Center (IIIRC)
[2]Shenzhen Key Laboratory of Internet Information Collaboration
School of Computer Science and Technology
Harbin Institute of Technology Shenzhen Graduate School
HIT Campus Shenzhen University Town Xili, Shenzhen 518055 P.R. China
[3]Department of Computer Science, University of Moncton, Canada
[4]Department of Computer Science and Information Engineering
National University of Kaohsiung, Kaohsiung, Taiwan, R.O.C.
[5]Department of Computer Science and Engineering
National Sun Yat-sen University, Kaohsiung, Taiwan, R.O.C.
[6]Department of Computer Science
National Chiao Tung University, Hsinchu, Taiwan, R.O.C.
jerrylin@ieee.org, wsgan001@gmail.com, philippe.fournier-viger@umoncton.ca, tphong@nuk.edu.tw, vtseng@cs.nctu.edu.tw

Abstract—High-utility itemsets mining (HUIM) is an extension of frequent itemsets mining that considers the measures of quantity and profit. Most algorithms for HUIM mine high-utility itemsets (HUIs) from a certain database in which each item has certainly shown the bought quantity in the transactions. In real-life applications, an item or itemset is not only present or absent in the transactions but also associated with an existing probability, especially the collected data from experimental measurements or noisy sensors. In the past, many algorithms were respectively proposed to effectively mine frequent itemsets over uncertain databases. The mining of HUIs from an uncertain database has not yet been proposed but is commonly seen in real-world applications. In this paper, a novel framework, namely potential high-utility itemsets mining (PHUIM) over uncertain databases, is proposed to

[*] Corresponding author

efficiently discover not only the itemsets with high utilities but also the itemsets with high probabilities of existence in an uncertain database based on the tuple uncertainty model. The PHUI-UP algorithm (potential high-utility itemsets upper-bound-based mining algorithm) is presented first to level-wisely mine the potential high-utility itemsets (PHUIs). Since PHUI-UP applies the generate-and-test framework to mine PHUIs, the secondly PHUI-List algorithm (potential high-utility itemsets PU-list-based mining algorithm) is then designed to directly mine PHUIs based on the designed *p*robability-utility-list (PU-list) structures without candidate generation, thus greatly improving the scalability for the mining of PHUIs. Substantial experimental results were conducted on both real-life and synthetic datasets to show the performance of the two designed algorithms in terms of runtime, number of patterns, memory consumption, and scalability.

Keywords: High-utility itemsets mining; uncertain database; probabilistic-based; upper-bound; 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 [6, 5, 7, 11, 14, 31]. Depending on different requirements in various domains and applications, association rule mining (ARM) [5, 7] is an important and common issue in KDD. In ARM, it first discovers the frequent itemsets in a level-wise way based on the minimum support threshold and then derives the association rules (ARs) based on the minimum confidence threshold. Many algorithms have been proposed to efficiently mine ARs from precise databases, and they can be generally classified into level-wise and pattern-growth approaches. Agrawal et al. first designed the well-known Apriori algorithm to mine ARs in a level-wise way [7]. Han et al. then presented the FP-growth algorithm to first construct the frequent 1-itemsets into compressed tree structures and second discover the frequent itemsets from the constructed FP-tree without candidate generation [14]. Traditional ARM only considers whether or not the item or itemset is present in a transaction. Other major factors in real-life applications, such as profit, quantity, cost, or other measures according to the user’s preference, which may bring more valuable and profitable strategies for retailers or managers, are not considered in ARM [32].

High-utility itemsets mining (HUIM) [10, 22, 33, 32] considers both the quantity and the profit of an item or itemset to measure how “useful” an item or itemset is. An itemset is called a high-utility itemset (HUI) if its total utility value in a database is no less than a user-specified minimum utility count. Generally speaking, the “utility” of an itemset/pattern can be presented as the user-specified importance, i.e., weight, cost, risk, unit profit or value. The goal of HUIM is to identify the rare items or itemsets in the transactions, but it can bring valuable profits for the retailer. Chan et al. first presented the concepts of HUIM [10]. Yao et al. defined a strict unified framework for mining high-utility itemsets (HUIs) [32]. Liu et al. designed a Two-Phase model [22] to estimate the upper bound of the transaction-weighted downward closure (TWDC) property, which can greatly reduce the number of candidates for mining HUIs with an additional database rescan. Lin et al. also presented the high utility pattern (HUP)-tree structure for efficiently mining HUIs without candidate generation [17]. Vincent et al. then inherited a similar idea to the HUP-tree property for designing an utility pattern (UP)-tree with two mining algorithms for efficiently discovering HUIs [28, 27]. Based on the pattern-growth approach, more computations are still required to generate and keep the huge number of candidates discovered for mining HUIs. To solve the above limitations of traditional HUIM, Liu et al. proposed the HUI-Miner model to directly produce HUIs without multiple database rescans and candidate generate-and-test based on the utility-list structures [21]. The proposed algorithm has great performance compared to the state-of-the-art algorithms in HUIM. The designs of many algorithms for mining HUIs [13, 19] from precise databases are still in progress.

In real-life applications, data may be uncertainly collected from incomplete data sources, RFID, GPS, wireless sensors, or WiFi systems [4, 3]. Due to the incomplete or inaccurate data, traditional frequent pattern (e.g., itemsets, association rule) mining cannot be applied to discover the required information for decision making. Many algorithms are developed to handle uncertain databases for discovering useful information. The UApriori algorithm was first proposed to mine the frequent itemsets over an uncertain database by employing the generate-and-test and breadth-first search mechanisms [12]. The uncertain frequent pattern (UFP)-growth was then designed to mine the uncertain frequent itemsets without candidate generation based on the UFP-tree structure [15]. Lin et al. also presented a compressed uncertain frequent pattern (CUFP)-tree [18] for directly producing the uncertain frequent itemsets from the built tree nodes. The development of a few related works on mining the uncertain frequent itemsets over an uncertain database is still in progress [3, 20, 26, 30].

For HUIM, most algorithms are developed to handle a precise database, which is not practical in real-life applications. In real-life applications, an item or itemset is 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 basket analysis of an uncertain database, each transaction contains several items with their purchase probabilities for the customer. For instance, the transaction {*A*:2, *C*:3, *E*:2, 90%} indicates that an event consists of three items {*A*, *C*, *E*} with quantities {2, 3, 2} can be bought with 90% probability of existence. In the risk prediction field, the risk event also be recorded with an occur probability. For instance, the event {*B*:1, *D*:3, *E*:1, 75%} indicates that an event consists of three items {*B*, *D*, *E*} with occur frequency {1, 3, 1} was happened with 75% probability of existence. Since the "utility" of an itemset/pattern can be presented as the user-specified importance, i.e., weight, cost, risk, unit profit or value. The HUIM can be used as an useful tool in many real-world applications, and most application scenarios are associated with uncertain database. In particular, 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. It may be misleading if the discovered results of HUIs with low existential probability, in fact, people always more interested in finding the high existential probability and high profitable patterns. Mining of HUIs over an uncertain database has not yet been developed. In this paper, a novel potential high-utility itemsets mining (PHUIM) model is designed to mine the potential and meaningful patterns named highly profitable itemsets with high potential probability (w.r.t potential high-utility itemsets, PHUIs). Two mining algorithms called PHUI-UP (potential high-utility itemsets upper-bound-based mining algorithm) and PHUI-List (potential high-utility itemsets PU-list-based mining algorithm) are respectively developed based on the level-wise approach and the designed probability-utility-list (PU-list) structures to mine PHUIs. Major contributions of this paper are summarized as follows:

1. Previous works on HUIM have addressed the mining of HUIs from a precise database. To the best of our knowledge, this is the first paper to address the issue of mining PHUIs from an uncertain database.
2. A new knowledge namely potential high-utility itemsets (PHUIs) is first formulated. Moreover, the potential high-utility itemsets mining framework (PHUIM) is proposed.
3. Two algorithms, PHUI-UP (potential high-utility itemsets upper-bound-based mining algorithm) and PHUI-List (potential high-utility itemsets PU-list-based mining algorithm), are respectively designed to efficiently mine PHUIs over an uncertain database, and they can be used as state-of-the-art algorithms by other researchers in the future.
4. PHUI-UP is proposed as a baseline algorithm for mining PHUIs level-wise over

an uncertain database based on an Apriori-like mechanism and the designed upper-bound model. As an improved algorithm, PHUI-List is proposed for discovering PHUIs directly without candidate generation based on a designed vertical data structure, probability-utility-list (PU-list).

5. Substantial experiments conducted on both real-life and synthetic databases show that the two proposed algorithms can effectively discover the complete PHUIs over an uncertain database. Specifically, the second algorithm has a better performance than the first in terms of runtime, memory consumption, and scalability.

The remainder of this paper is organized as follows. Related works are briefly reviewed in Section 2. The preliminaries and problem statement related to PHUIM over an uncertain database are presented in Section 3. The two proposed algorithms, PHUI-UP and PHUI-List are respectively described in Section 4. An experimental evaluation showing the performances of the proposed approaches is provided in Section 5. Conclusions and future work 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 are briefly reviewed.

2.1 Mining Frequent Itemsets over Uncertain Databases

Most approaches for mining frequent itemsets of ARs are processed to handle a binary database, 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 [4, 3]. Discovering the frequent itemsets over an uncertain database has emerged as an important issue in recent years [3, 12, 15, 20, 26, 30]. Depending on the specific applications, approaches to mining uncertain frequent itemsets over an uncertain database can be generally classified into two classes: the expected support-based model [2, 12, 15] and the probabilistic frequency model [9, 25]. The expected support-based model considers the expectation of support as the frequency metric, while the probabilistic frequency model utilizes the frequentness probability as a measurement for mining the probabilistic frequent itemsets [26]. In the expected support-based model, an itemset is considered as a frequent itemset over an uncertain database if its expected support is no less than a specified minimum expected support. In the probabilistic frequent model, an itemset is defined as a frequent itemset if its frequentness probability is no less than a specified minimum probability.

Chui et al. first designed the UApriori algorithm [12] to mine the frequent itemsets over uncertain databases level-wise based on the well-known Apriori algorithm. The expected support threshold was thus defined for mining frequent itemsets over an uncertain database. 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 structures and divide-and-conquer and depth-first search strategies, and it has better performance compared to the UApriori algorithm. UH-mine was proposed by Aggarwal et al. [2]; it was extended from the H-Mine algorithm and also adopts the divide-and-conquer and depth-first search strategies to carry out recursive processing for mining uncertain frequent itemsets from the UH-Struct structure. Wang et al. then presented the MBP algorithm [29] by reducing the number of candidates compared to the number considered by the UApriori algorithm over an uncertain database. Lin et al. then designed a compressed uncertain frequent pattern (CUFP)-tree structure to efficiently mine the uncertain frequent itemsets [18]. Most algorithms, such as UApriori [12], UFP-growth [15], UH-mine [2], and CUFP-Miner [18], belong to the expected support-based model.

Bernecker et al. first proposed a new probabilistic formulation for mining frequent itemsets based on *possible world semantics* model; it is called the probabilistic frequent model and is quite different from the previous expected support-based model [9]. Sun et al. also developed p-FP structures and proposed two efficient algorithms which discover frequent patterns in bottom-up (p-Apriori) and top-down (TODIS) manners [25]. The p-Apriori and TODIS algorithms were compared with regard to mining of the probabilistic frequent itemsets by adopting dynamic programming or the divide-and-conquer strategy based on the probabilistic frequent model. Besides frequent itemsets mining over an uncertain database, approaches for mining the uncertain frequent itemsets over data streams have been proposed, such as UF-streaming [16] and SUF-growth [16]. Recently, Tong et al. combined the expected support-based model and probabilistic frequent model to present a new representation for mining frequent itemsets over uncertain database with uniform measures [26]. The development of many algorithms for mining frequent itemsets over uncertain databases is still in progress.

2.2 High-Utility Itemsets Mining

High-utility itemsets mining (HUIM) is based on the measurement of local transaction utility and external utility 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 [10]. A new strategy was also developed for pruning the low utility itemsets, thus speeding up the computations for the mining of HUIs. Yao et al. defined the problem of utility mining by analyzing the utility relationships of the itemsets [32]. 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 [22] to efficiently discover HUIs based on the designed TWDC property. It first uses the transaction-weighted utility (TWU) as an effective upper bound to keep the high transaction-weighted utilization itemsets (HTWUIs) in order to avoid the combinational problem of HUIM in a level-wise way. The original database is then rescanned to determine the remaining HTWUIs for deriving HUIs. Lin et al. proposed the HUP-tree algorithm [17] to find HUIs without candidate generation. It adopted the Two-Phase model and the designed tree structure to keep 1-HTWUIs. An array-based structure was attached to each node for keeping the quantities of its prefix path in the tree, thus speeding up the computations for the mining of HUIs. Vincent et al. proposed the UP-tree structure and developed two mining algorithms, UP-growth [27] and UP-growth+ [28], to efficiently derive HUIs. Liu et al. proposed the HUI-Miner algorithm [21] to build utility-list structures 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. Based on the presented experiments, HUI-Miner became the state-of-the-art algorithm in HUIM in terms of execution time and memory consumption. Fournier-Viger et al. further designed an FHM algorithm by enhancing the property of HUI-Miner for analyzing the co-occurrences among 2-itemsets [13]. The number of operations can be greatly reduced based on the FHM algorithm, but a little more memory is required to keep the co-occurrence relationships of the 2-itemsets.

The development of other algorithms for HUIM is still in progress, but most of them are processed to handle a precise database. However, to the best of our knowledge, mining of high-utility itemsets over an uncertain database has not yet been developed, this is the first work that studies the problem of mining potential high-utility itemsets (PHUIs) over an uncertain database.

3. Preliminaries and Problem Statement

In this section, we first discuss which uncertain database model and which uncertainty calculated model are suitable for the address problem. Then, the preliminaries and problem statement related to potential high-utility itemsets mining (PHUIM) over an uncertain database are given below.

3.1 The uncertain database model and probability measure

The probabilistic frequent itemsets (PFI) model [9, 25] proposed by Bernecker et al. aims at calculating the (exact) probability that an itemset X is frequent: $P_{\geq minSup}(X)$ (w.r.t. frequentness probability), in which $minSup$ is the user-specified minimum support. It finds only those itemsets that have a high probability of being frequent (confidence in results). In real-life applications, the derived high utility itemsets are usually, however, rare but high utility. It ignores the number of possible worlds in which an itemsets is frequent, while PFI model does. For example, when the number of possible worlds where an itemset having a high exiting probability is infrequent but high-utility, we still interest in this high probability itemset. Hence, the probabilistic frequent itemsets model is not suitable to the addressed PHUIM. In contrast, an itemset is considered frequent if its expected support (the summation probabilities) in the uncertain database exceeds *minSup* in the expected-support model [2, 12, 15], it is similar to the potential probability in our PHUIM framework somewhat.

In our PHUIM model, we aim at discovering those itemsets with high utility which will be happened with high probability, and the frequentness is not suitable to our PHUIM framework. The uncertainty model used in our approach is very close to the model used for probabilistic databases. A tuple uncertainty probabilistic database, called "tuple uncertainty" [24], where each tuple is associated with an existing probability is adopt in the PHUIM model. The probability measure which is similar to the expected support-based frequent pattern mining model [2, 12, 15] is adopted in the proposed PHUIM framework, and we named this potential probability measure. The derived patterns have high potential probability being a HUI in the uncertain database. Hence, in this paper, the tuple uncertainty database model [24] and the uncertainty calculated model which is somewhat similar to the expected support-based model [2, 12, 15] are adopted in the proposed two algorithms.

3.2 Preliminaries

Let $I = \{i_1, i_2, \ldots, i_m\}$ be a finite set of m distinct items in an uncertain quantitative database $D = \{T_1, T_2, \ldots, T_n\}$, 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 uncertainty 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 potential probability threshold, are respectively defined as ε and μ.

An example of an uncertain quantitative database 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 ε and the minimum potential probability threshold μ are respectively set at ε (= 25%) and μ (= 15%).

Table 1: An uncertain database

TID	Transaction (item, quantity)	probability
1	(*A*, 2); (*C*, 3); (*E*, 2)	0.9

2	(*B*, 1); (*D*, 2)	0.7
3	(*A*, 1); (*B*, 2); (*C*, 1); (*E*, 3)	0.85
4	(*C*, 2)	0.5
5	(*B*, 3); (*D*, 2); (*E*, 1)	0.75
6	(*A*, 2); (*C*, 2); (*D*, 5)	0.7
7	(*A*, 1); (*B*, 1); (*D*, 4); (*E*, 1)	0.45
8	(*B*, 4); (*E*, 1)	0.36
9	(*A*, 3); (*C*, 3); (*D*, 2)	0.81
10	(*B*, 2); (*C*, 3); (*E*, 1)	0.6

Table 2: The utility table

Item	Profit ($)
A	4
B	1
C	12
D	6
E	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, in Table 1, the utility of item {*C*} in T_1 is $u(C, T_1) = q(C, T_1) \times pr(C)$ (= 3×12) (= 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, consider an item {*C*} and an itemset {*A*C} in T_1; $p(C, T_1) = p(T_1)$ (= 0.9), and $p(AC, T_1) = p(T_1)$ (= 0.9).

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 × 4 + 3 × 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, in Table 1, $u(A) = u(A, T_1) + u(A, T_3) + u(A, T_6) + u(A, T_7) + u(A, T_9)$ (= 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).

In the expected support-based model in uncertain data mining, the expected support of an itemset X in D can be computed by summing the support of X in possible world W_j (while taking into account the probability of W_j being the true world) over all possible worlds as $expSup(X) = \sum_{i=1}^{|D|} (\prod_{x \in X} P(x, T_q))$ [12, 26]. The probability measure of the concept of PHUI in an uncertain database is defined as follows.

Definition 5. *The potential probability of an itemset X in D is denoted as $Pro(X)$, which can be defined as* $Pro(X) = \sum_{X \subseteq T_q \wedge T_{q \in} D} p(X, T_q)$.

For example, in Table 1, $Pro(A) = p(A, T_1) + p(A, T_3) + p(A, T_6) + p(A, T_7) + p(A, T_9)$ (= 0.9 + 0.85 + 0.7 + 0.45 + 0.81) (= 3.71), and $Pro(ABE) = p(ABE, T_3) +$

$p(ABE, T_7)$ (= 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, in Table 1, $tu(T_1) = u(A, T_1) + u(C, T_1) + u(E, T_1)$ (= 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, in Table 1, 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 in a database D is defined as a high-utility itemset (HUI) if its utility value u(X) is larger than or equal to the minimum utility count as $\sum_{X \subseteq T_q \wedge T_q \in D} u(X, T_q) \geq \varepsilon \times TU$. Otherwise, it is called a low-utility itemset.*

For example, in Table 1, 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 × 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 potential itemset* (*HPI*) *if the potential probability of an itemset X is larger than or equal to the minimum potential probability, which is defined as* $\sum_{X \subseteq T_q \wedge T_q \in D} p(X, T_q) = Pro(X) \geq \mu \times |D|$.

For example, in Table 1, since μ is set at 15%, the minimum potential probability is calculated as (15% × 10) (= 1.5). Thus, an itemset {*A*} is considered as a high potential itemset since *Pro*(*A*) (= 3.71 > 1.5). An itemset (*ABE*) is, however, not considered as a high potential itemset since its potential probability is calculated as *Pro*(*ABE*) (= 1.3), which is smaller than the minimum potential probability (= 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. It means that X which having high potential probability is regarded as a HUI due to its high utility value*.

Based on the above definitions, the problem statement of mining PHUIs over an uncertain database can be formulated as below.

3.3 Problem Statement

Given an uncertain database *D*, its total utility is *TU*, the minimum utility threshold is set at ε, the minimum potential probability threshold is set at μ, the problem of potential high-utility itemsets mining (PHUIM) over the uncertain database is to mine the potential high-utility itemsets (PHUIs) whose utilities are larger than or equal to (ε

$\times$ *TU*), and its potential probability is larger than or equal to ($\mu \times |D|$).

From the running example given in Tables 1 and 2, the set of PHUIs is shown in Table 3 when the minimum utility threshold is set at 25% and the minimum potential probability threshold is set at 15%.

Table 3: The derived PHUIs for the example uncertain database

Itemsets	Utility	*Pro*
{*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. The Proposed Potential High-Utility Itemsets Mining Algorithms over Uncertain Databases

In this section, two efficient algorithms for mining PHUIs over an uncertain database, PHUI-UP and PHUI-List, are respectively described as follows. The PHUI-UP algorithm is first proposed as a baseline algorithm to mine PHUIs level-wise over an uncertain database based on Apriori-like [7] approach and the developed upper-bound. PHUI-List is further designed to improve the performance of the PHUI-UP algorithm to discover PHUIs directly over an uncertain database based on the designed probability-utility (PU)-list structure and a Set-enumeration tree without candidate generation each time.

5.1 Proposed PHUI-UP Algorithm

To the best of our knowledge, this is the first paper to discuss the PHUIM over an uncertain database. A naive PHUI-UP algorithm is presented here to mine PHUIs in a level-wise way based on the well-known Apriori-like mechanism [5, 7].

5.1.1 The pruning strategy by the 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-UP algorithms for mining PHUIs.

Theorem 1 (Downward closure property of high probability itemsets). *If an itemset is a high probability itemset in an uncertain database, then this itemset contains the downward closure property in this database.*

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

$$Pro(X^k) = \sum_{X^k \subseteq T_q \wedge T_q \in D} p(X^k, T_q) \leq \sum_{X^{k-1} \subseteq T_q \wedge T_q \in D} p(X^{k-1}, T_q) = Pro(X^{k-1}).$$

If X^k is a HPI, its potential probability is larger than or equal to the minimum potential probability count as $Pro(X^k) \geq |D| \times \mu$, and thus $Pro(X^{k-1}) \geq Pro(X^k) \geq |D| \times \mu$.

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 the HUIs. The TWDC property [22] 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*) *iff its $TWU(X) \geq TU \times \varepsilon$.*

For example, 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 an uncertain database, and a novel potential transaction-weighted utilization downward closure (PTWUDC) property is further designed to reduce the search space of PHUI-UP algorithm for mining PHUIs.

Definition 13. *An itemset X is defined as a potential high transaction-weighted utilization itemset* (*PHTWUI*) *iff $TWU(X) \geq \varepsilon \times TU$ and $Pro(X) \geq \mu \times |D|$.*

For example, in Tables 1 and 2, since μ is set at 15%, the minimum potential probability is calculated as (15% × 10) (= 1.5). Take an item (E) as an example, according to definition 12, $TWU(E)$ (= 283) (> 134). In addition, $Pro(E) = p(E, T_1) + p(E, T_3) + p(E, T_5) + p(E, T_7) + p(E, T_8) + p(E, T_{10})$ (= 0.9 + 0.85 + 0.75 + 0.45 + 0.36 + 0.6) (= 3.91) (> 1.5). Thus, the itemset {E} is considered as a potential high transaction-weighted utilization itemset (PHTWUI).

Theorem 2. (Downward closure property of PHTWUI). *Let X^k and X^{k-1} be the PHTWUI from the uncertain database, and $X^{k-1} \subseteq X^k$. Then the downward closure property o f PHTWUI indicates that $TWU(X^{k-1}) \geq TWU(X^k)$ and $Pro(X^{k-1}) \geq Pro(X^k)$.*

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

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

From Theorem 1, it can be found that $Pro(X^{k-1}) \geq Pro(X^k)$; therefore, if X^k is a PHTWUI, any subset X^{k-1} is also a PHTWUI.

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

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

Theorem 3 (PHUIs ⊆ PHTWUIs). *The potential transaction-weighted utilization downward closure (PTWUDC) property ensures that PHUIs ⊆ PHTWUIs, which indicates that if an itemset is not a PHTWUI, then none of its supersets will be*

PHUIs.

Proof. For ∀ $X \in D$, X is an itemset; thus

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

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

5.1.2 The PHUI-UP algorithm

The proposed PHUI-UP algorithm has two phases: in the first phase, the PHTWUIs and are found, and in the second phase, the PHUIs are derived with an additional database rescan. The PTWUDC 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 PHTWUI^{k-1} will be used to generate the next C_k at each level.

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

Based on the above definitions and theorems, details of the proposed PHUI-UP algorithm are described below.

Algorithm 1: PHUI-UP

INPUT: *D*, the uncertain transactional database; *ptable*, profit table; ε, minimum utility threshold; μ, minimum potential probability threshold;
OUTPUT: The set of potential high-utility itemsets (PHUIs).

```
1.  scan D to calculate $TWU(i_j)$ and $Pro(i_j)$ for each $i_j$.
2.  for each $i_j \in D$ do
3.      if $TWU(i_j) \geq TU \times \varepsilon \wedge Pro(i_j) \geq |D| \times \mu$ then
4.          $PHTWUI^1 \leftarrow PHTWUI^1 \cup i_j$.
5.      end if
6.  end for
7.  set $k \leftarrow 2$, then $X$ as the ($k$)-itemset.
8.  while $PHTWUI^{k-1} \neq \emptyset$ do
9.      $C_k = Apriori\_gen(PHTWUI^{k-1})$;
10.     for each $k$-itemset $X \in C_k$ do
11.         scan D to calculate $TWU(X)$ and $Pro(X)$.
12.         if $TWU(X) \geq TU \times \varepsilon \wedge Pro(X) \geq |D| \times \mu$ then
13.             $PHTWUI^k \leftarrow PHTWUI^k \cup X$.
14.         end if
15.     end for
16.     $k \leftarrow k+1$.
17. end while
18. $PHTWUIs \leftarrow \bigcup PHTWUI^k$ .
19. for each $k$-itemset $X \in PHTWUIs$ do
20.     scan D to calculate $u(X)$.
21.     if $u(X) \geq TU \times \varepsilon$ then
22.         $PHUIs \leftarrow PHUIs \cup X$.
23.     end if
24. end for .
25. return PHUIs
```

As shown, the proposed PHUI-UP algorithm first scans the database to find the

TWU values and the probabilities of all 1-itemsets in the database (Line 1). According to the designed conditions of PHTWUI^k (Line 3), the PHTWUI^k (k is initially set as 1) are then produced (Lines 2 to 6) and will be further used to generate the next candidates C_{k+1} for discovering PHTWUI^{k+1} in a level-wise way (Lines 7 to 18). In this process, the original database has to be rescanned to find the PHTWUI^{k+1} (Line 11). The first phase of PHUI-UP is terminated when no candidate is generated. Then the algorithm performs the second phase, in this phase, an additional database rescan is required to find the final PHUIs from the PHTWUIs (Lines 19 to 24).

5.1.3 An example for PHUI-UP algorithm

In order to keep consistency, consider the example which given in Tables 1 and 2, also assume the minimum utility threshold is set at 25% and the minimum potential probability threshold is set at 15%. Thus, the minimum support count and the minimum potential probability can be calculated as (442 × 25%) (= 110.5) and (10 × 15%) (= 1.5), respectively. The detailed proceeds of the proposed PHUI-UP algorithm as follows.

The PHUI-UP algorithm first scans the database to find the TWU values and the probabilities of all 1-itemsets in the database, the results is {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 itemset {A} has its TWU values as 303 and probability value as 3.71. Since each of

1-itemsets satisfies *TWU*(*X*) > 110.5 *and Pro*(*X*) > 1.5, all of them are put into the set of potential high transaction-weighted utilization itemsets (PHTWUIs). Then *k* is set to 2, the PHUI-UP algorithm generates the candidates C_2 by applied *Apriori_gen*(*PHTWUI*^{1}) in alphabetical order, the set of C_2 is {*AB*; *AC*; *AD*; *AE*; *BC*; *BD*; *BE*; *CD*; *CE*; *DE*}. The uncertain database is required to rescan to calculate the *TWU* and *Pro* values for each itemset in C_2, the results is {*AB*: 107, 1.3; *AC*: 259, 3.26; *AD*: 122, 1.51; *AE*: 181, 2.2; *BC*: 116, 1.45; *BD*: 87, 1.9; *BE*: 190, 2.05; *CD*: 122, 1.51; *CE*: 190, 2.35; *DE*: 74, 1.2}. Among them, only the itemset {*AC*: 259, 3.26; *AD*: 122, 1.51; *AE*: 181, 2.2; *BE*: 190, 2.05; *CD*: 122, 1.51; *CE*: 190, 2.35} satisfy *TWU*(*X*) > 110.5 and *Pro*(*X*) > 1.5, and put into the set of PHTWUIs. Then *k* is set to 3 and process in the same way, the first phase terminates when no candidate is generated. After the first phase, the complete set of PHTWUIs is shown at Table 4.

Table 4: The derived PHTWUIs for the example uncertain database

Itemsets	*TWU*	*Pro*
{*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

Then the PHUI-UP algorithm performs the second phase, an additional database rescan is required to find the final PHUIs from the PHTWUIs by calculating the actual utility value for each of PHTWUIs. Taking itemsets {A} and {AC} in Table 4 as an example, according definition 4, $u(A) = 36 < 110.5$ and $u(AC) = 140 > 110.5$, so the itemset {A} is not a PHUI, while itemset {AC} is regarded as a PHUI. Finally, the complete PHUIs are discovered by the PHUI-UP algorithm, the results are shown at Table 3.

5.2 Proposed PHUI-List Algorithm

In the past, HUI-Miner [21] was proposed to efficiently mine HUIs from the precise database. Experiments on HUI-Miner indicate that it has the best performance compared to the state-of-the-art HUIM algorithm. In this section, an efficient PHUI-List algorithm is proposed to improve the performance of the PHUI-UP algorithm for efficiently mining PHUIs. A new vertical data structure, called probability-utility-list (PU-list), is designed to store the necessary information for PHUI-List. Thus, PHUI-List can directly mine PHUIs without multiple database rescan compared to the PHUI-UP algorithm. Details of the presented PU-list structure, the enumeration tree for the search space, and the proposed PHUI-List algorithm are described below.

5.2.1 Probability-Utility-List (PU-List) Structure

The PU-list structure inherits the utility-list property to keep more related information from each of transactions to directly mining PHUIs. Each entry for an 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 (**iu**), and the remaining utility of *X* in T_q (**ru**). For two item/itemset *X* and *Y*, note that the order $X \prec Y$ means that *X* is before *Y.*

Definition 14. *An entry for an itemset X of the PU-list 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 (**iu**)*, and the remaining utilities of X in* T_q (**ru**)*, in which* **ru** *is defined as* $X.ru = \sum_{i \notin X \wedge i \in T_q \wedge X \prec i} u(i, T_q)$.

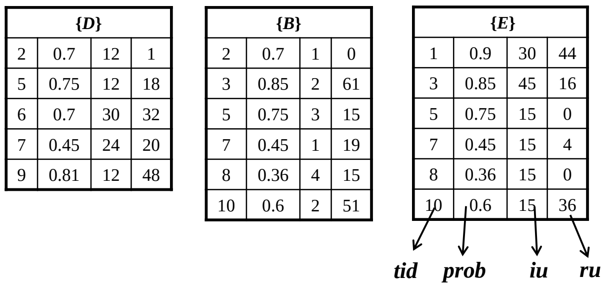
The construction procedure of the PU-list is recursively processed if it is necessary to determine the *k*-itemsets in the search space. The construction procedure is shown in Algorithm 2.

Algorithm 2 PU-list construction procedure

```
INPUT: $X.PUL$, the PU-list of an itemset $X$;
            $X_a.PUL$, the PU-list of an itemset $X_a$;
            $X_b.PUL$, the PU-list of an itemset $X_b$ ($X_a \neq X_b$).
OUTPUT: $X_{ab}.PUL$.
1.  $X_{ab}.PUL \leftarrow \emptyset$.
2.  for each element $E_a \in X_a$ do
```

```
3.      if $\exists E_b \in X_b.PUL$ and $E_a.tid = E_b.tid$ then
4.          if $X.PUL \neq \emptyset$ then
5.              Search for element $E \in X.PUL$ such that $E.tid = E_a.tid$;
6.              $E_{ab} \leftarrow <E_a.tid, E_a.prob, E_a.iu + E_b.iu - E.iu, E_b.ru>$.
7.          else
8.              $E_{ab} \leftarrow <E_a.tid, E_a.prob, E_a.iu + E_b.iu, E_b.ru>$.
9.          end if
10.         $X_{ab}.PUL \leftarrow E_{ab}$.
11.     end if
12. end for
13. return $X_{ab}.PUL$.
```

Note that it is necessary to initially construct the PU-list of the PHTWUI[1] as the input for the later recursive process. Since the 1-itemsets of PHTWUI[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 itemset {*A*} has its TWU values as 303 and probability value as 3.71, and the PU-list is constructed in ascending order of their TWU values as {*D* < *B* < *E* < *A* < *C*}, as shown in Figure 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*	*iu*	*ru*

{*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

Figure 1: The constructed PU-list structure of PHTWUI[1]

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

$$X.IU = \sum\nolimits_{X \subseteq T_q \wedge T_q \in D} (X.iu).$$

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

$$X.RU = \sum\nolimits_{X \subseteq T_q \wedge T_q \in D} (X.ru).$$

Taking an itemset {*A*} from Figure 1 as an example to illustrate the process. The itemset {*A*} exists in TID {1, 3, 6, 7, 9}, its *A.IU* is calculated as (8 + 4 + 8 + 4 + 12) (= 36), and *A.RU* is calculated as (36 + 12 + 24 + 0 + 36) (= 108). Besides, for an itemset {*AE*} which exists in TID = {1, 3, 7}, its *AE.IU* (= *A.IU* + *E.IU*) {= (8 + 4 + 8) + (30 + 45 + 15)} (= 110), and *AE.RU* = *A.RU* + *E.RU* {= (36 + 12 + 0) + (44 + 16 + 4)} (= 112).

5.2.2 An Enumeration Tree

Based on the PU-list structures, the search space of the proposed PHUI-List algorithm can be represented as the enumeration tree by the TWU values of 1-PHTWUIs 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 Figure 2.

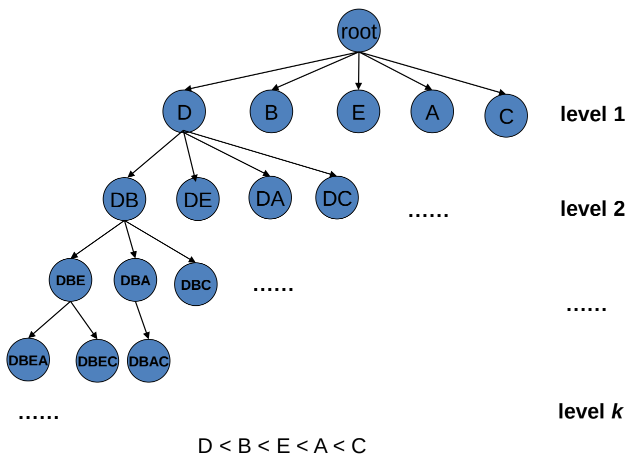


Figure 2: An enumeration tree

The PHUI-List algorithm is based on a 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***, ***iu***, and ***ru*** 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 ***iu*** and ***ru*** of the current processed node is larger than or equal to the minimum utility count ($\varepsilon \times TU$), and (2) the *Pro* (also the ***prob***) of the processed node is larger than or equal to the minimum potential probability ($\mu \times |D|$), the supersets of the processed node will be generated and determined (the pruning strategy will be described later in more detail). Finally, the actual utility of the processed node can be directly determined by its ***tid***, ***iu***, and the ***prob*** of discovering PHUIs without database scan. Throughout the construction of the enumeration tree, we can observe the following lemmas.

Lemma 1. *The search space of the proposed PHUI-List algorithm can be represented as the enumeration tree by the TWU values of 1-PHTWUIs in ascending order*.

Proof. Based on the construction of the enumeration tree, as shwon in Figure 2, when traversing the enumeration tree based on a bread-first search strategy, from top-to-down, the nodes in level 1 are first processed, then processed the nodes in level 2, until level *k*. It is similar to the PHUI-UP algorithm which applied the Apriori-like mechanism to level-wisely generate candidate and test. Hence, the enumeration tree can represent the complete search space of the proposed PHUI-List algorithm. In addition, the search space of the enumeration tree is equal no matter adopt a bread-first search strategy or a depth-first search strategy.

5.2.3 Early Pruning Strategies

Based on the above definitions, two early pruning strategies are used to find a more compressed search space according to the PTWUDC property, a great number of unpromising candidates were efficiently pruned, thus significantly reducing the search space of the proposed PHUI-List algorithm. Throughout the construction of the enumeration tree, we can observe the following lemmas.

Lemma 2. *The sum of all the probabilities of any node in the enumeration tree* *are greater than or equal to the sum of all the probabilities of any one of its children.*

Proof. Assume that a node in the enumeration tree is X^{k-1}, then any one of X^{k-1} its children (supersets) can be denoted as X^k. According to Theorems 2 and 3, we can get $TWU(X^k) = \sum_{X^k \subseteq T_q \wedge T_q \in D} tu(T_q) \leq \sum_{X^{k-1} \subseteq T_q \wedge T_q \in D} tu(T_q) = TWU(X^{k-1})$ and $Pro(X^{k-1}) \geq Pro(X^k)$. Thus, we thus have this property.

Pruning strategy 1. *When traversing the enumeration tree based on a depth-first search strategy, if the sum of all the probabilities of a tree node X in its constructed PU-list is less than the minimum potential probability, then none of the children of node X is a PHUI.*

Rationale. According to **Lemma 2**, we can realize that if the sum of all the probabilities of a node is less than the minimum potential probability, it can be regarded as an unpromising itemset to PHUI, then any it's child node (supersets) is also an unpromising itemset. Hence, when the sum of all the probabilities of itemsets are being estimated, those probabilities of unpromising itemsets and their supersets can be regarded as irrelevant and be pruned directly.

Lemma 3. *For any node X in the enumeration tree, the sum of X.IU and X.RU are greater than or equal to the sum of all the utilities of any one of its children.*

Proof. Assume that a node X in the enumeration tree is X, then any one of X its children (supersets) can be denoted as X'. For $\forall$ transaction $T_q \supseteq X$:

$\because$ *X*’ is an extension of *X* ⇒ (*X*'−*X*) = (*X*'/*X*)

$$X \subset X' \subseteq T_q \Rightarrow (X'/X) \subseteq (T_q/X)$$

$\therefore$ in T_q, $X'.iu = X.iu + (X'/X).iu$

$$= X.iu + \sum\nolimits_{z \in (X'/X)} z.iu$$

$$\le X.iu + \sum\nolimits_{z \in (T_q/X)} z.iu$$

$$= X.iu + X.ru;$$

$\therefore$ in T_q, $X'.iu \le X.iu + X.ru$.

$\because$ $X \subset X' \Rightarrow X'.tids \subseteq X.tids$

$\therefore$ in *D*, $X'.IU = \sum_{T_q \in X'.tids} X'.iu$

$$\le \sum_{T_q \in X'.tids} (X.iu + X.ru)$$

$$\le \sum_{T_q \in X.tids} (X.iu + X.ru)$$

$$= X.IU + X.RU.$$

$\therefore$ in *D*, $X'.IU \le X.IU + X.RU$.

Pruning strategy 2. *When traversing the enumeration tree based on a depth-first search strategy, if the sum of X.IU and X.RU in the constructed PU-list is less than the minimum utility count, then none the children of node X is a PHUI.*

Rationale. According to **Lemma 2**, we can realize that if a node is less than the minimum utility count ($\varepsilon \times TU$), it can be regarded as an unpromising itemset to PHUI, then any it’s child node (supersets) is also unpromising itemset. 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 actual number of nodes be traveled in the enumeration tree* (*denoted as* N_1), *is always smaller than or equal to the number of candidate itemsets* (*denoted as* N_2) *generated by the PHUI-UP algorithm, it indicates that the two early pruning strategies which used in PHUI-List performs a more compressed search space then the PTWUDC property which used in PHUI-UP.*

Proof. By **Lemmas 2** and **3**, it can be see that the PU-list keep the remaining utilities is smaller than the TU value, so for any itemset X, the sum of $X.IU$ and $X.RU$ is much smaller than its TWU value. As a more tighter upper-bound, PU-list thus efficiently pruning the more unpromising itemsets and performs a more compressed search space then the PTWUDC property. Thus, $N_1 \leq N_2$.

Lemma 5. *The final results of PHUIs discovered by the PHUI-List algorithm is correct and complete.*

Proof. With the PHUI-List algorithm, every transaction from an uncertain database is scanned firstly and those promising items their knowledge are extracted according to the ***tid***, ***prob***, ***iu***, and ***ru***. After the knowledge are extracted, those items are continously construted their PU-list structures. When travering the enumeration tree to find the PHUIs, each node can be evaluated exactly from the PU-lists. By **Lemmas 1**, **2** and **3**, the PHUI-List algorithm can ensure that no any promising itemsets will be discarded (completeness) and the related information can be obtain exactly from the PU-list structures (correctness). Therefore, it can be said that based on hte PU-list structures and enumeration tree, the results which obtained by the proposed PHUI-List algorithm are correct and complete.

Based on the two proposed pruning strategies, the designed PHUI-List algorithm can prune the itemsets with lower potential probability count and utility count early, without constructing their supersets' PU-lists, which can effectively reduce both the computations of join operations and the search space from the enumeration tree.

5.2.4 The PHUI-List Algorithm

Based on the above definitions and properties, the pseudo-code of the proposed PHUI-List algorithm is described below.

Algorithm 3: PHUI-List Algorithm

```
INPUT: $D$, the uncertain transactional database; ptable, profit table; $\varepsilon$, minimum
utility threshold; $\mu$, minimum potential probability threshold;
OUTPUT: The set of potential high-utility itemsets (PHUIs).
/* $X.PNUL$, the PU-list of an itemset $X$ */;
/* $D.PNUL$, the PU-list of $D$ */;
1.  $D.PUL \leftarrow \emptyset$, $X.PUL \leftarrow \emptyset$;
2.  scan $D$ to calculate the $TWU(i)$ and $Pro(i)$ value of each item $i \in I$.
3.  find $I^* \leftarrow \{i \mid TWU(i) \geq TU \times \varepsilon \wedge Pro(i) \geq |D| \times \mu, i \in I\}$.
4.  sort $I^*$ in $TWU$ ascending order;
5.  scan $D$ to construct the $X.PUL$ structure of each 1-item $i \in I^*$.
6.  $D.PUL \leftarrow \cup X.PUL$;
7.  call PHUI-Search($\emptyset$, $I^*$, $\varepsilon$, $\mu$);
8.  return PHUIs
```

As shown in the main procedure of PHUI-List (Algorithm 3), the proposed PHUI-List algorithm first scans the uncertain database to calculate the *TWU*(*i*) and *Pro*(*i*) value of each item *i*∈*I*, and then find the potential high transaction-weight utilization 1-itemsets (I^* w.r.t. $PHTWUI^1$) (Line 3). After sort I^* in *TWU* ascending order, the PHUI-List algorithm scan *D* again to construct the PU-list of each 1-item in $PHTWUI^1$ w.r.t $i \in I^*$ (Lines 5 to 6). The probability-utility list (*D.PUL*) for all 1-extensions of $i \in I^*$ is recursively processed by using a depth-first search procedure **PHUI-Search** (Line 7). As shown in **PHUI-Search** (Algorithm 4), each itemset X_a is determined to directly produce the PHUIs (Lines 2 to 5). Two pruning strategies (Line 6) are then applied to further determine whether its supersets satisfy the designed conditions for executing the later depth-first search (Lines 6 to 14). The construction process **Construct(*X*, X_a, X_b)** is then executed to construct a set of PU-list of all 1-extensions of itemset X_a (Lines 7 to 12), then recursively processing the designed **PHUI-Search** procedure to mine PHUIs (Line 13). Based on the designed PU-list structure, the PHUI-List algorithm can thus directly mine the complete PHUIs from the uncertain database without candidate generation.

Algorithm 4: PHUI-Search Procedure

INPUT: *X*, an itemset; *extendOfX*, a set of PU-list of all 1-extensions of itemset *X*; ε, the user-specified minimum utility threshold; μ, the user-specified minimum potential probability threshold.

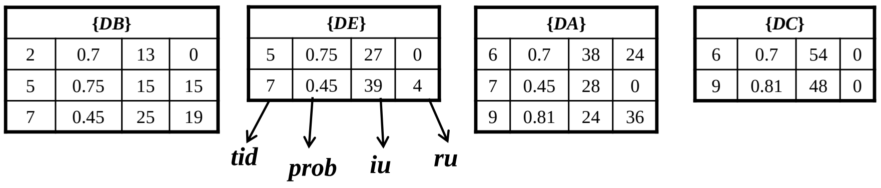
OUTPUT: The set of potential high-utility itemsets (PHUIs).

```
1. for each pattern $X_a \in extendOfX$ do
2.     obtain the $X_a.IU$ and $Pro(X_a)$ values from the constructed $X_a.PUL$;
3.     if $X_a.IU \geq TU \times \varepsilon \wedge Pro(X_a) \geq |D| \times \mu$ then
4.         $PHUIs \leftarrow PHUIs \cup X_a$.
5.     end if
```

```
6.     if ($X_a.IU + X_a.RU \geq TU \times \varepsilon$) ∧ ($Pro(X_a) \geq |D| \times \mu$) then
7.         $extendOfX_a \leftarrow \emptyset$ .
8.         for each itemset $X_b \in extendOfX$ such that $b$ after $a$ do
9.             $X_{ab} \leftarrow X_a \cup X_b$;
10.            $X_{ab}.PUL \leftarrow$ Construct($X$, $X_a$, $X_b$);
11.            $extendOfX_a \leftarrow extendOfX_a \cup X_{ab}.PUL$.
12.        end for
13.        call PHUI-Search($X_a$, $extendOfX_a$, $\varepsilon$, $\mu$);
14.    end if
15. end for
16. return PHUIs
```

5.2.5 An example for PHUI-List Algorithm

In order to keep consistency, consider the same example and thresholds which described in section 4.1.3. The PHUI-List algorithm first scans the original uncertain database to extract the necessary information (i.e. ***tid***, ***prob***, ***iu*** and ***ru***) for each 1-items from each of transactions, then the PU-list structures of all 1-items have been constructed. If any 1-items X satisfies $TWU(X) \geq \varepsilon \times TU$ *and* $Pro(X) \geq \mu \times |D|$, put into the set of potential high transaction-weighted utilization itemsets (PHTWUIs), the final set of 1-PHTWUIs are shown in Figure 1, sorted by the TWU values in ascending order. The itemset {D} is first processed, since {D} does not satisfies the PHUI conditions: $D.IU$ = (12 + 12 + 30 + 24 + 12) (= 90) < (442 × 25%) (= 110.5), and $Pro(D)$ = (0.7 + 0.75 + 0.7 + 0.45 + 0.81) (= 3.41) > (10 × 15%) (= 1.5). Thus, itemset {D} is not regarded as a PHUI, but ($D.IU$ + $D.RU$) = (90 + (1 + 18 + 32 + 20 + 12)) (= 119), larger than the minimum utility count, its children nodes may be a PHUI, then still perform the depth-first search. Based on the PU-list structures of 1-PHTWUIs, executes the PU-list construction procedure to construct PU-list structures for 2-itemsets with prefix itemset {*D*}, as shown in Figure 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

Figure 3: The constructed PU-list of PHTWUI^2 with prefix itemset {*D*}

Then the first child node (*DB*) is determined, by the PU-list structures of (*DB*), the sum of utilities and probabilities of (*DB*) can be calculated as *DB.IU* = (13 + 15 + 25) = 53 < 110.5, and *Pro(DB*) = (0.7 + 0.75 + 0.45) = 1.9 > 1.5, so (*DB*) and all its children are not regarded as a PHUI and pruned, stop the depth-first search of (*DB*). The next child node {*DE*} is then processed in the same way, as well as other nodes. After traveling over the enumeration tree, the complete PHUIs can be directly discovered by the PHUI-List algorithm without database scan, the results are shown at Table 3.

5. Experimental Results

In this section, substantial experiments on the two proposed algorithms, PHUI-UP and

PHUI-List, for mining PHUIs over an uncertain dataset were conducted to evaluate the performance in terms of runtime, memory consumption, the number of discovered patterns, and scalability. Notice that this is the first work focuses on potential high-utility ittemset mining, we further compare the derived PHUIs with the traditional HUIs which derived by the well-known traditional HUIM algorithm, the patterns analysis between PHUI and HUIs can evaluate the acceptable the proposed HPUIM framework.

All 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. The experimental results are shown and discussed below.

5.1 Datasets

Both real-life and synthetic datasets were used in the experiments. Three real-life datasets, namely foodmart [23], accident [1], and retail [1], as well as one synthetic dataset, T10I4D100K [8], were used in the experiments to evaluate the performance of the two proposed algorithms. The foodmart dataset was collected from an anonymous chain store. It is a quantitative dataset of the products sold by the chain store. The accident dataset contains anonymous traffic accident data for a public road in Belgium. The retail dataset contains anonymous retail market basket data from a retail store in Belgian. The synthetic dataset T10I4D100K was generated by IBM Quest Synthetic Data Generation Code [8]. 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 [22] 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. Parameters and characteristics of these datasets are respectively shown in Tables 5 and 6.

Table 5: Parameters of used datasets

#|*D*|	Total number of transactions
#|*I*|	Number of distinct items
AvgLen	Average length of transactions
MaxLen	Maximal length of transactions
Type	Dataset type: sparse or dense

Table 6: Characteristics of used datasets

Datasets	**#|*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 performance was used to compare the performances of the two proposed algorithms in four uncertain datasets under varied minimum utility thresholds (MUs) and varied minimum potential probability thresholds (MPs). The runtime of the two proposed algorithms under varied MUs with a fixed MP value are compared and shown in Figure 4. In addition, the runtime of the two proposed algorithms under varied MPs with a fixed MU value are then compared and shown in Figure 5.

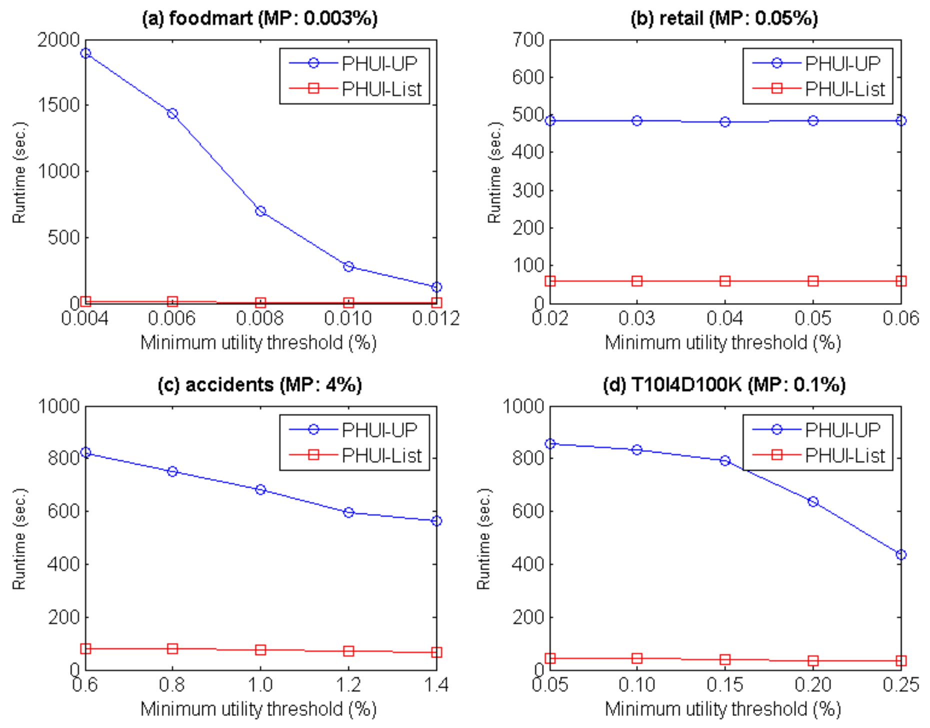


Figure 4: Runtime of the two proposed algorithms under varied MUs with a fixed MP.

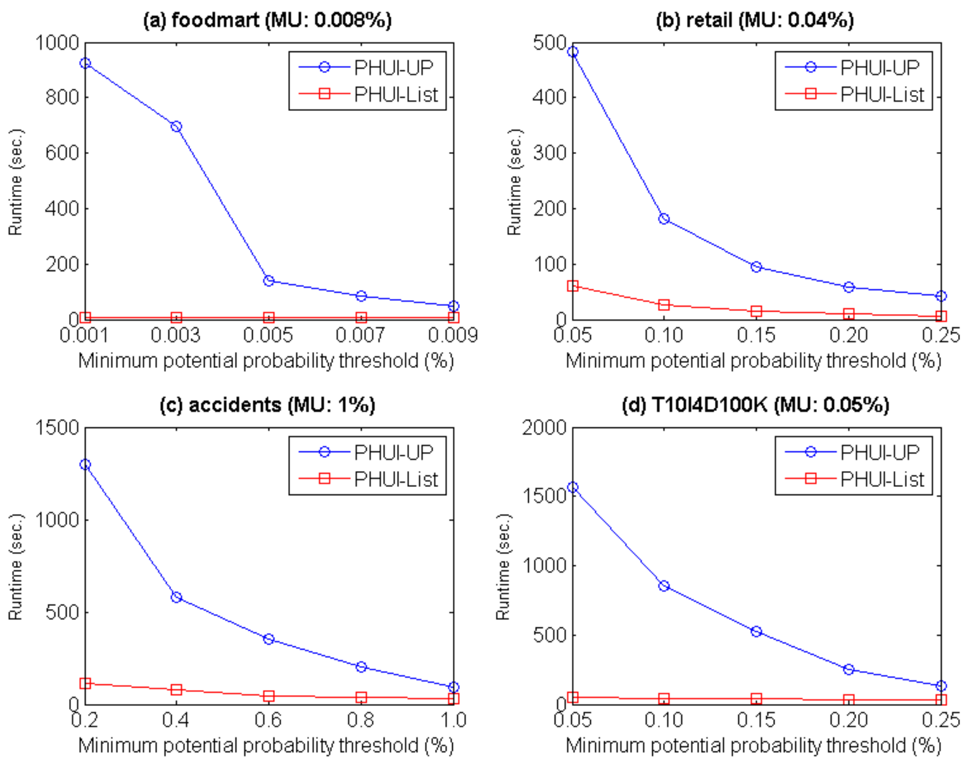


Figure 5: Runtime of the two proposed algorithms under varied MPs with a fixed MU.

From Figure 4, it can be observed that the proposed PHUI-List algorithm has better performance than the naive PHUI-UP algorithm. The proposed PHUI-List algorithm is generally up to almost one or two orders of magnitude faster than the PHUI-UP algorithm, which can indicate that the generate-and-test approach has worse performance than the PU-list structure. For example, for the accidents dataset in Figure 4(c), the MP was set at 1% and the varied MUs were set from 0.2% to 1.0%, with a 0.2% increment each time. The runtime of PHUI-UP decreased dramatically from 822 to 562 seconds, while that of PHUI-List changed steadily from 80 seconds to 66 seconds. The reason is that when the MUs are set quite low, longer patterns of

PHTWUIs are discovered first before the PHUIs are found by the designed PHUI-UP 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. While the PHUI-List algorithm directly determine the PHUIs from the enumeration tree without candidate generate-and-test, it effectively avoiding the time-consuming dataset scan. Moreover, the PHUI-List algorithm applied two pruning strategies to effectively prune the unpromising items early on, based on the designed PU-list structures, thus greatly reducing the runtime needed to discover PHUIs.

As shown in Figure 5, it can also be observed that the PHUI-List algorithm outperforms the PHUI-UP algorithm under the varied MPs in the four datasets. Specifically, the runtime of PHUI-UP decreased sharply as the MPs increased, while the runtime of the PHUI-List algorithm decreased steadily. The result is reasonable since when MP is set higher, fewer candidates are generated for later processing to mine the PHUIs. Although the PHUI-UP algorithm uses the PTWUDC property to reduce the search space, it is still performed in a level-wise way to generate and test enormous candidates for mining PHUIs. In this situation, huge numbers of candidates are generated but PHUIs are produced only rarely. Therefore, when MP is set higher, many redundant unpromising candidates are pruned early on, and thus the search space and runtime of the PHUI-UP algorithm decrease sharply.

5.3 Patterns Analysis

In this subsection, the PHUIs patterns which discovered from the uncertain dataset by the PHUI-UP and PHUI-List algorithms are evaluated. Notice that an algorithm for discovering PHUIs over an uncertain dataset has not been developed in the past. In fact, when the probability of each transaction is set at 1.0 over the uncertain dataset, it means each of transactions has the 100% existential probabilities, then the uncertain dataset turns into a precise quantitative dataset, it equals to the traditional utility mining algorithms to mine HUIs. The state-of-the-art HUI-miner algorithm is thus used to mine HUIs in the precise dataset without probability values compared to the designed PHUI-UP and PHUI-List algorithms for mining PHUIs. The results of pattern analysis for HUIs and PHUIs under varied MUs with a fixed MP and under varied MPs with a fixed MU are respectively shown in Figures 6 and 7.

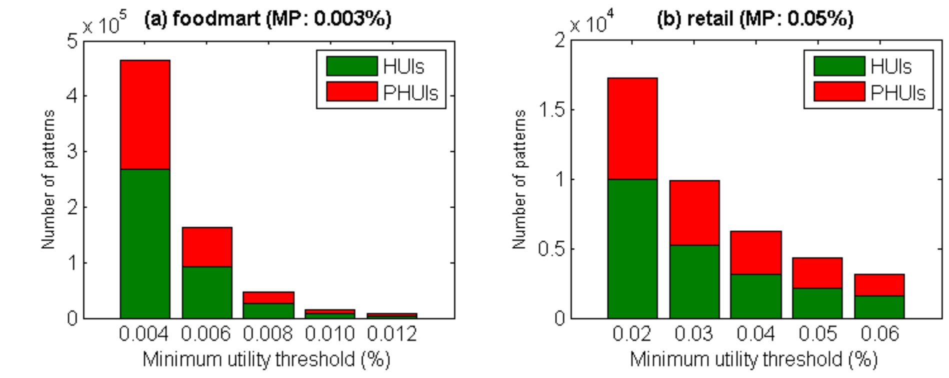
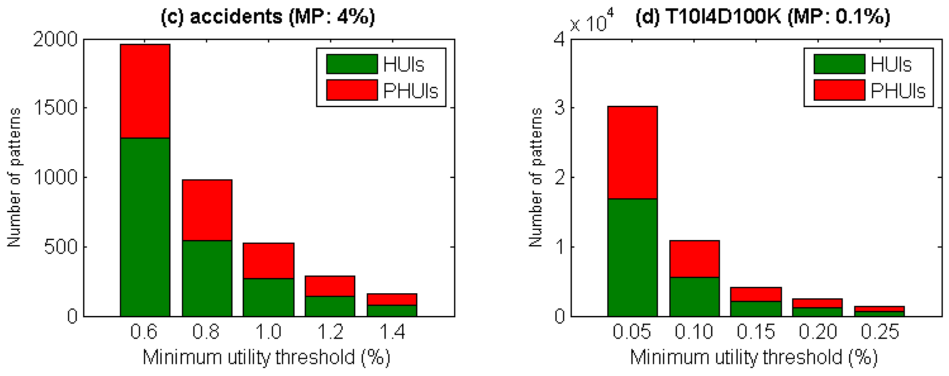



Figure 6: Number of HUIs and PHUIs under varied MUs with a fixed MP.

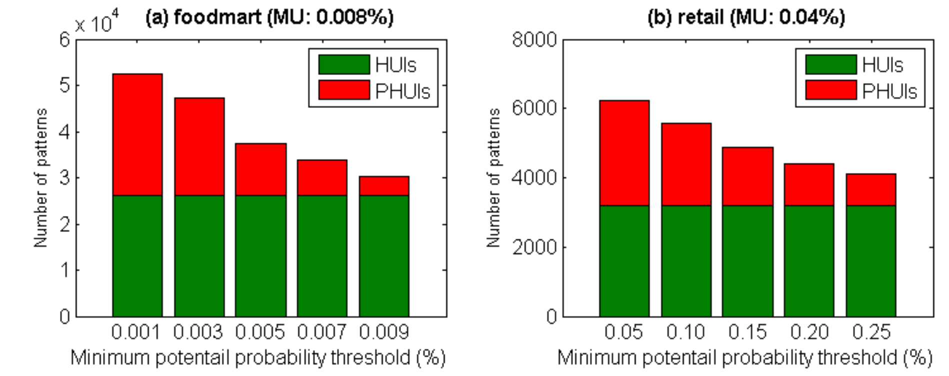
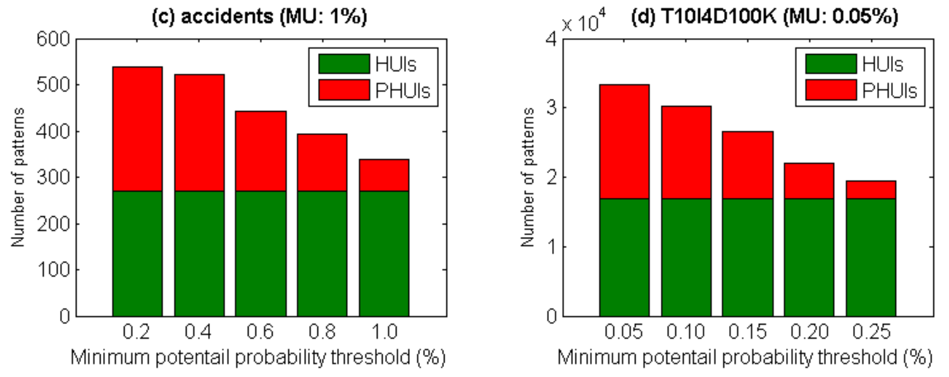



Figure 7: Number of HUIs and PHUIs under varied MPs with a fixed MU.

From Figure 6, it can be observed 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 an uncertain dataset. 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 PHUIs patterns are fewer and more valuable compared to those general HUIs patterns because PHUIs have distinct probability values that consider a variety of item probability information. From the experimental results, it can also be seen that both the number of HUIs and the number of PHUIs decrease as the MUs increase.

From Figure 7, it can be observed that fewer PHUIs are discovered by the proposed algorithms compared to the number of HUIs discovered by HUI-Miner under varied MPs in the four uncertain datasets. The number of discovered HUIs remains steady as the MPs increase. This is reasonable since HUI-Miner only considers the high-utility constraint for discovering HUIs, the number of HUIs are no effected when the MPs increase higher. The number of PHUIs decreases dramatically as the MPs increase. The reason is the same as the one described in Figure 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 of discovering HUIs, since the probability factor generally occurs in a real-life situation.

5.4 Memory Consumption

The two designed algorithms were evaluated to show the performance of memory consumption by using the Java API. The results under varied MUs with a fixed MP and under varied MPs with a fixed MU are respectively shown in Figures 8 and 9. Both Figures 8 and 9 show the peak memory consumption of the two proposed algorithms on all datasets.

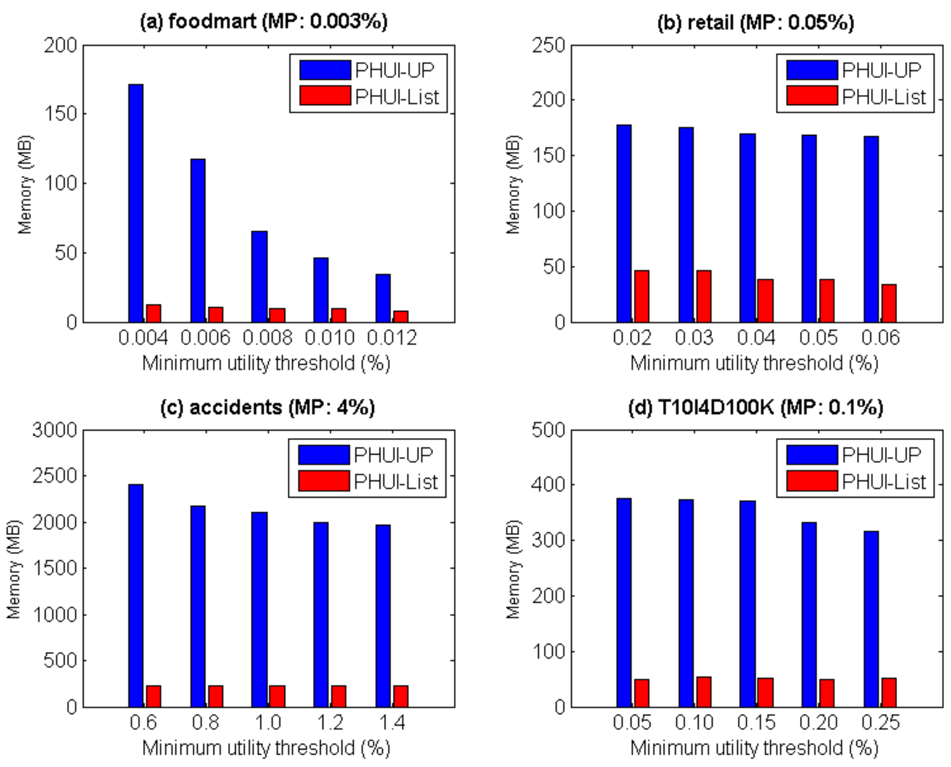


Figure 8: Memory consumption under varied MUs with a fixed MP.

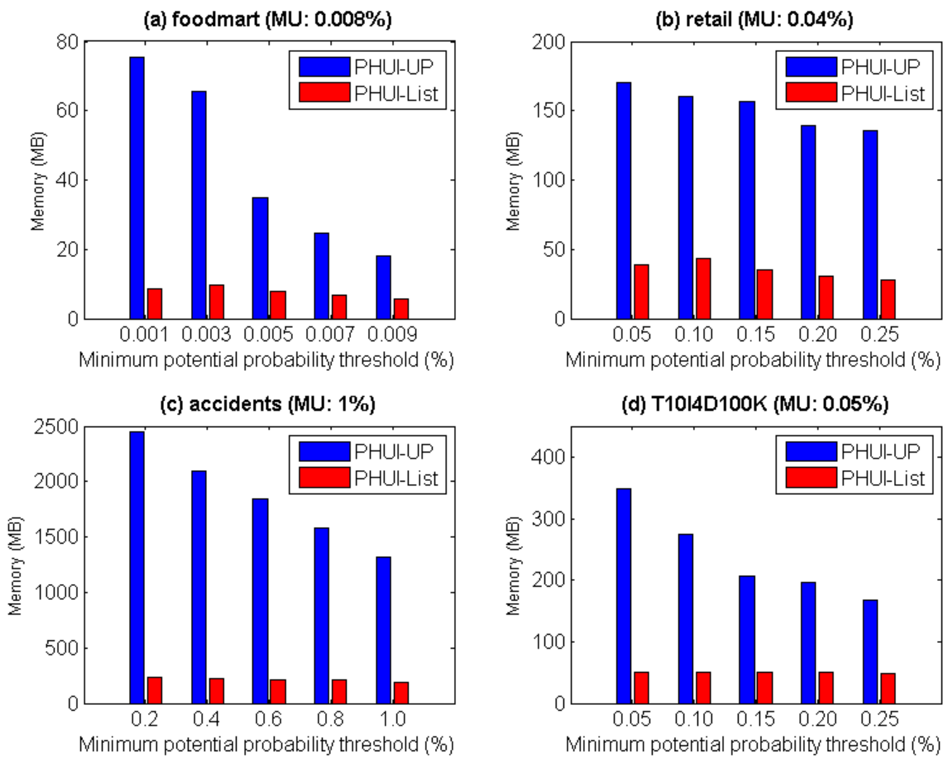


Figure 9: Memory consumption under varied MPs with a fixed MU.

From Figures 8 and 9, it can be clearly seen that the proposed PHUI-List algorithm requires less memory compared to the PHUI-UP algorithm both under varied MUs with a fixed MP and under varied MPs with a fixed MP for the four datasets. Specially, the PHUI-List algorithm requires nearly constant memory under varied parameters for the four datasets. The PHUI-UP algorithm requires more memory when the MU or MP is set lower compared to the PHUI-List algorithm. For example, PHUI-UP requires 2100 MB of memory on average, but PHUI-List only requires 225 MB of memory on average, as can be observed in Figure 8(c). The reason is that the proposed algorithms are based on two strategies for pruning unpromising candidates for the later process of mining PHUIs. It is also easy to find that the performance gap between PHUI-UP and PHUI-List gets smaller with increasing MU with a fixed MP or increasing MP with a fixed MU. For instance, when MP was set at 0.001 and MU at 0.008%, the PHUI-UP and the PHUI-List algorithms required 76.5 MB and 9 MB of memory, respectively, which can be observed from the foodmart datasets in Figure 9(a). When MP was set at 0.009% and MU at 0.008%, the PHUI-UP and PHUI-List algorithms required 20 MB and 5.5 MB of memory, respectively, as shown in Figure 9(a). Since the unpromising candidates can be pruned early on, 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-UP algorithm under varied parameters for the four datasets. When MU or MP is set lower, the “combinational explosion” problem always occurs at each iteration based on the generate-and-test mechanism, producing numerous redundant patterns in a level-wise way compared to the PHUI-List algorithm.

5.5 Scalability

The scalability of the two proposed algorithms is compared on the 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 different MP and MU are shown in Figure 10.

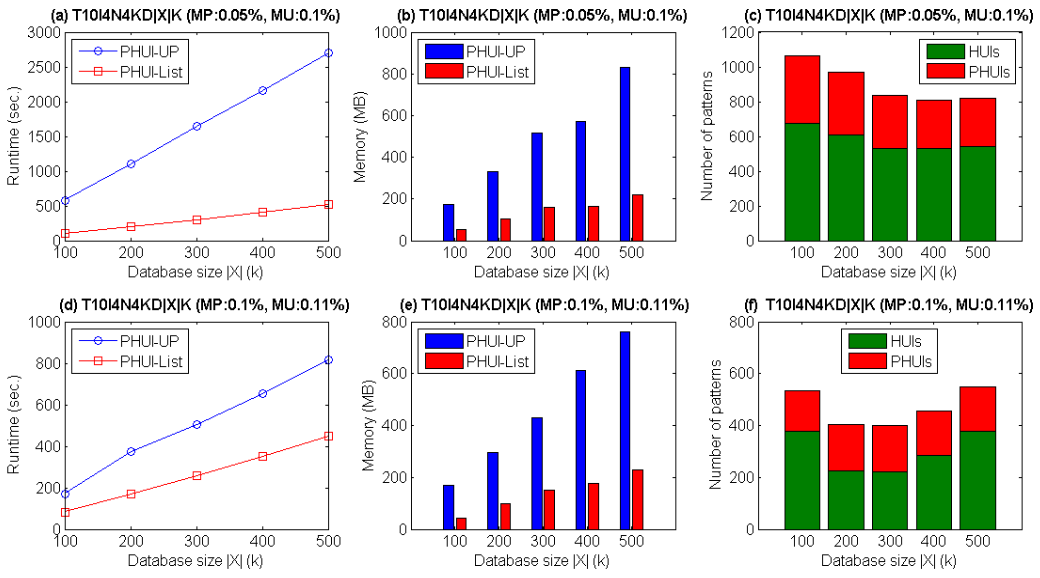


Figure 10: Scalability results of the two proposed algorithms.

From Figure 10, it can be observed 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 Figures 10(a) to 10(d), 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-UP algorithm. An interesting observation is that the runtime and memory consumption of the proposed PHUI-List algorithm increased steadily but those of PHUI-UP increased sharply as the dataset size increased. For example, PHUI-List was almost two orders of magnitude faster than the PHUI-UP algorithm when MP was set at 0.05% and MP was set at 0.1% as the dataset size increased from 100K to 500K, which can be observed from Figure 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 Figures 10(c) and 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, number of patterns, and scalability.

6. Conclusions and Future Work

In the past, many approaches have been proposed to respectively mine frequent itemsets from an uncertain database or to mine high-utility itemsets from a precise database. However, little research has discussed the mining of high-utility itemsets from an uncertain database so far. In this paper, a novel framework is considered to mine the *potential high-utility itemsets* (*PHUIs*) from an uncertain database. Two algorithms, *PHUI-UP* (*potential high-utility itemsets upper-bound-based mining algorithm*) and *PHUI-List* (*potential high-utility itemsets PU-list-based mining algorithm*), are respectively proposed to consider the mining of not only high-utility itemsets but also high probability itemsets from an uncertain database. The designed PHUI-UP algorithm is based on the level-wise approach to generate and test candidates for mining PHUIs. Although it keeps the designed downward closure property to reduce the search space, the combinational explosion problem still occurs in the naive PHUI-UP algorithm. The second PHUI-List algorithm is then developed to improve the performance compared to the PHUI-UP algorithm based on the designed PU-list structure and the enumeration tree for mining PHUIs directly without candidate generation. Substantial experiments were conducted on both real-life and synthetic databases to show the performance of the two proposed algorithms in terms of runtime, patterns analysis, memory consumption, and scalability.

Since this is the first work to address the novel framework for mining PHUIs from an uncertain database, further research issues including dynamic data mining, stream mining, and top-*k* patterns mining can also be studied for further improvements. Besides, designing more efficient and condensed structures based on a different uncertain model for mining the desired information is also another critical issue.

Acknowledgement

This research was partially supported by the Tencent Project under grant CCF-TencentRAGR20140114, by the Shenzhen Peacock Project, China, under grant KQC201109020055A, by the Natural Scientific Research Innovation Foundation in Harbin Institute of Technology under grant HIT.NSRIF.2014100, and by the Shenzhen Strategic Emerging Industries Program under grant ZDSY20120613125016389.

Reference

[1] *Frequent itemset mining dataset repository*. Available: http://fimi.ua.ac.be/data/ (2012)

[2] C. C. Aggarwal, Y. Li, J. Wang, and J. Wang, "Frequent pattern mining with uncertain data," *The 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 29-38, 2009.

[3] C. C. Aggarwal and P. S. Yu, "A survey of uncertain data algorithms and applications," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21(5), pp. 609-623, 2009.

[4] C. C. Aggarwal, "Managing and mining uncertain data," *Managing and Mining Uncertain Data*, 2010.

[5] R. Agrawal, T. Imielinski, and A. Swami, "Mining association rules between sets of items in large database," *The ACM SIGMOD International Conference on Management of Data*, pp. 207-216, 1993.

[6] R. Agrawal, T. Imielinski, and A. Swami, "Database mining: A performance perspective," *IEEE Transactions on Knowledge and Data Engineering*, vol. 5(6), pp. 914-925, 1993.

[7] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules in large databases," *International Conference on Very Large Data Bases*, pp. 487-499, 1994.

[8] R. Agrawal and R. Srikant. *Quest synthetic data generator*. Available: http://www.Almaden.ibm.com/cs/quest/syndata.html (1994)

[9] T. Bernecker, H. P. Kriegel, M. Renz, F. Verhein, and A. Zuefl, "Probabilistic frequent itemset mining in uncertain databases," *The 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 119-128, 2009.

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

[11] 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(6), pp. 866-883, 1996.

[12] 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.

[13] P. Fournier-Viger, 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.

[14] 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(1), pp. 53-87, 2004.

[15] 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.

[16] C. K. S. Leung and B. Hao, "Mining of frequent itemsets from streams of uncertain data," *IEEE 25th International Conference on Data Engineering*, pp. 1663-1670, 2009.

[17] 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(6), pp. 7419-7424, 2011.

[18] C. W. Lin and T. P. Hong, "A new mining approach for uncertain databases using cufp trees," *Expert Systems with Applications*, vol. 39(4), pp. 4084-4093, 2012.

[19] C. W. Lin, G. C. Lan, and T. P. Hong, "An incremental mining algorithm for high utility itemsets," *Expert Systems with Applications*, vol. 39(8), pp. 7173-7180, 2012.

[20] C. Liu, L. Chen, and C. Zhang, "Summarizing probabilistic frequent patterns: A fast approach," *The 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 527-535, 2013.

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

[22] Y. Liu, W. K. Liao, and A. Choudhary, "A two-phase algorithm for fast

discovery of high utility itemsets," *Lecture Notes in Computer Science*, pp. 689-695, 2005.

[23] Microsoft. *Example database foodmart of microsoft analysis services*. Available: http://msdn.microsoft.com/en-us/library/aa217032(SQL.80).aspx

[24] D. Nilesh and S. Dan, "Efficient query evaluation on probabilistic databases," *The VLDB Journal*, vol. 16(4), pp. 523-544, 2007.

[25] L. Sun, R. Cheng, D. W. Cheung, and J. Cheng, "Mining uncertain data with probabilistic guarantees," *The 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 273-282, 2010.

[26] Y. Tong, L. Chen, Y. Cheng, and P. S. Yu, "Mining frequent itemsets over uncertain databases," *The VLDB Endowment*, vol. 5(11), pp. 1650-1661, 2012.

[27] V. S. Tseng, C. W. Wu, B. E. Shie, and P. S. Yu, "UP-growth: An efficient algorithm for high utility itemset mining," *The 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 253-262, 2010.

[28] 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.

[29] L. Wang, R. Cheng, S. D. Lee, and D. Cheung, "Accelerating probabilistic frequent itemset mining: A model-based approach," *The 19th ACM International Conference on Information and Knowledge Managemen*, pp. 429-438, 2010.

[30] 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.

[31] X. Wu, V. Kumar, J. R. Quinlan, J. Ghosh, Q. Yang, H. Motoda, *et al.*, "Top 10

algorithms in data mining," *Knowledge and Information Systems*, vol. 14, pp. 1-37, 2006.

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

[33] H. Yao and H. J. Hamilton, "Mining itemset utilities from transaction databases," *Data & Knowledge Engineering*, vol. 59(3), pp. 603-626, 2006.