Knowledge-Based Systems xxx (2016) xxx–xxx

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

Efficient algorithms for mining high-utility itemsets in uncertain databases

Q1 Jerry Chun-Wei Lin[a,*], Wensheng Gan[a], Philippe Fournier-Viger[b], Tzung-Pei Hong[c,d], Vincent S. Tseng[e]

Q2 [a] School of Computer Science and Technology, Harbin Institute of Technology, Shenzhen Graduate School, HIT Campus, Shenzhen University, Town Xili, Shenzhen, China
[b] School of Natural Sciences and Humanities, Harbin Institute of Technology, Shenzhen Graduate School, HIT Campus, Shenzhen University, Town Xili, Shenzhen, China
[c] Department of Computer Science and Information Engineering, National University of Kaohsiung, Kaohsiung, Taiwan
[d] Department of Computer Science and Engineering, National Sun Yat-sen University, Kaohsiung, Taiwan
[e] Department of Computer Science, National Chiao Tung University, Hsinchu, Taiwan

ARTICLE INFO

Article history:
Received 12 June 2015
Revised 26 December 2015
Accepted 27 December 2015
Available online xxx

Keywords:
High-utility itemset
Uncertain database
Probabilistic-based
Upper-bound
PU-list structure

ABSTRACT

High-utility itemset mining (HUIM) is a useful set of techniques for discovering patterns in transaction databases, which considers both quantity and profit of items. However, most algorithms for mining high-utility itemsets (HUIs) assume that the information stored in databases is precise, i.e., that there is no uncertainty. But in many real-life applications, an item or itemset is not only present or absent in transactions but is also associated with an existence probability. This is especially the case for data collected experimentally or using noisy sensors. In the past, many algorithms were respectively proposed to effectively mine frequent itemsets in uncertain databases. But mining HUIs in an uncertain database has not yet been proposed, although uncertainty is commonly seen in real-world applications. In this paper, a novel framework, named potential high-utility itemset mining (PHUIM) in uncertain databases, is proposed to efficiently discover not only the itemsets with high utilities but also the itemsets with high existence probabilities in an uncertain database based on the tuple uncertainty model. The PHUI-UP algorithm (potential high-utility itemsets upper-bound-based mining algorithm) is first presented to mine potential high-utility itemsets (PHUIs) using a level-wise search. Since PHUI-UP adopts a generate-and-test approach to mine PHUIs, it suffers from the problem of repeatedly scanning the database. To address this issue, a second algorithm named PHUI-List (potential high-utility itemsets PU-list-based mining algorithm) is also proposed. This latter directly mines PHUIs without generating candidates, thanks to a novel probability-utility-list (PU-list) structure, thus greatly improving the scalability of PHUI mining. Substantial experiments were conducted on both real-life and synthetic datasets to assess the performance of the two designed algorithms in terms of runtime, number of patterns, memory consumption, and scalability.

1. Introduction

The main purpose of Knowledge Discovery in Database (KDD) is to discover meaningful and useful information from a collection of data [5–7,12,17]. Frequent itemset mining (FIM) and association rule mining (ARM) [6,7] are some of the most important and common tasks of KDD, since they meet the requirements of numerous domains and applications. ARM typically consists of discovering frequent itemsets (FIs) in a level-wise way using a user-specified minimum support threshold, to then derives association rules (ARs) by also considering a minimum confidence threshold. Many algorithms have been proposed to efficiently mine ARs from precise databases. 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, which first determines frequent 1-itemsets to build a compressed tree structure, and then discovers the frequent itemsets from the constructed FP-tree without generating candidates [17]. Traditional ARM only considers whether items or itemsets are present or not in transactions. But in real-life applications, several other factors need to be considered such as profit, quantity, cost, and other measures of users' preferences. Considering such factors allows

Q3 * Corresponding author. Tel.: +8618824678687.
E-mail addresses: jerrylin@ieee.org (J.C.-W. Lin), wsgan001@gmail.com (W. Gan), philfv@hitsz.edu.cn (P. Fournier-Viger), tphong@nuk.edu.tw (T.-P. Hong), vtseng@cs.nctu.edu.tw (V.S. Tseng).

http://dx.doi.org/10.1016/j.knosys.2015.12.019
0950-7051/

discovering more valuable patterns that let retailers and managers adopt more profitable business strategies than using patterns found using traditional ARM [39].

High-utility itemset mining (HUIM) [11,27,38,39] considers both the quantity and the profit of items and itemsets 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. The goal of HUIM is to identify items or itemsets in transactions that bring considerable profit to a retailer, although they may not be the most frequent ones. Chan et al. first introduced the concept of HUIM [11]. Yao et al. defined a strict unified framework for mining high-utility itemsets (HUIs) [39]. Liu et al. designed the Two-Phase model [27] to provide an upper bound on the utility of itemsets named the transaction-weighted utility (TWU), which can greatly reduce the number of candidates to be considered for mining HUIs by performing an additional database scan. Several tree-based approaches for efficiently discovering HUIs have been proposed, such as IHUP [9], HUP-growth [24], UP-growth [34] and UP-growth+ [33]. These pattern-growth approaches are faster than previous approaches but may still suffer from long execution time and large memory consumption because they need to generate and maintain a huge number of candidates in memory for mining HUIs. To address the above limitations of traditional HUIM, Liu et al. proposed the HUI-Miner algorithm to directly produce HUIs without performing multiple database scans and without using a candidate generate-and-test approach, by relying on a novel utility-list structure [26]. HUI-Miner was shown to outperform previous state-of-the-art HUIM algorithms. The design of more efficient algorithms for mining HUIs from precise databases is still an active research topic.

In real-life applications, uncertainty may be introduced when data is collected from noisy data sources such as RFID, GPS, wireless sensors, and WiFi systems [2,4]. When applied to incomplete or inaccurate data, traditional pattern mining algorithms (e.g., FIM, ARM) cannot be applied to discover the information required for decision making. Many algorithms have been developed to discover useful information in uncertain databases. The UApriori algorithm was first proposed to mine frequent itemsets in uncertain databases using a generate-and-test and breadth-first search approach [13]. The uncertain frequent pattern (UFP)-growth algorithm was then designed to mine uncertain frequent itemsets without generating candidates, using a UFP-tree structure [22]. Lin et al. also presented a compressed uncertain frequent pattern (CUFP)-tree [23], and an algorithm to mine uncertain frequent itemsets from the built tree nodes. Development of other algorithm for mining uncertain frequent itemsets in uncertain databases is still in progress [4,25,32,36].

In ref. [16], it had been thoroughly discussed that utility and probability are two different measures used for describing objects (e.g., useful patterns) in real-life applications. The utility is a semantic measure (how the "utility" of a pattern is measured according to a user's a priori knowledge and goals), while probability is an objective measure (the probability of a pattern's existence). Objective interestingness measures (e.g., probability) indicate the support and degree of correlation of a pattern in a given database. However, they do not consider the knowledge of the user who uses the data to discover patterns. Subjective and semantics-based measures (e.g., utility) incorporate the user's background knowledge and goals, and are suitable both for more experienced users and interactive data mining [16]. In HUIM, most algorithms are developed to handle precise databases, which mainly focus on the semantics-based utility measures and do not consider objective probability measures. Thus, traditional HUIM is insufficient to process uncertain data in real-life applications. In real-life applications, an item or itemset is not only present or absent in transactions but also often associated with an existential probability, especially when data is collected from experimental measurements or noisy sensors. Generally speaking, the "utility" of an itemset/pattern represents its importance to the user, i.e., weight, cost, risk, unit profit or value. Most realistic application scenarios are uncertain databases where the "utility" measure can also be considered. For example, in market basket analysis, an uncertain database contains customer transactions, where each transaction record obtained by RFID may be imprecise, i.e., it may contain several items, and be associated with an existence probability [2,4]. For instance, the transaction {A:2, C:3, E:2, 90%} indicates that an event consisting of three items {A, C, E} bought with quantities {2, 3, 2} has occurred with an existence probability of 90%. In the field of risk prediction, the risk associated with an event can also be viewed as an occurrence probability. For instance, the event {B:1, D:3, E:1, 75%} indicates that an event consists of three items {B, D, E} with occurrence frequencies of {1, 3, 1} and that it occurred with a 75% existence probability. Since the "utility" can be viewed as the user-specified importance, i.e., weight, cost, risk, unit profit or value, HUIM is a useful tool for many real-world applications. And most application scenarios are associated with uncertain databases, e.g., to discover the potential high utility itemsets in market basket analysis; find the potential high risk events in risk prediction system, and mine the potential high risk diseases to make predictions. But numerous discovered HUIs may not be the patterns required by a manager or retailer to take efficient decisions, since traditional HUIM algorithms do not consider existence probabilities. Discovered patterns may be misleading if they have low existential probabilities. In fact, people are always more interested in finding patterns with high existential probabilities than with low existential probabilities. But no algorithm has yet been proposed for mining HUIs in an uncertain database.

In this paper, a novel potential high-utility itemset mining (PHUIM) model is designed to mine potential and meaningful patterns, named highly profitable itemsets with high potential probability (abbreviated as 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 a level-wise approach and a designed probability-utility-list (PU-list) structure, to mine PHUIs. Major contributions of this paper are summarized as follows:

1. Previous work on HUIM has addressed the issue of mining HUIs efficiently in a precise database. To the best of our knowledge, this is the first paper to address the issue of mining PHUIs in an uncertain database that take both the semantics-based utility measure and the objective probability measure into account.
2. A novel type of patterns named potential high-utility itemset (PHUI) is designed. Moreover, the potential high-utility itemset 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 in an uncertain database. They can be used as state-of-the-art algorithms by researchers in future work.
4. PHUI-UP is proposed as a baseline algorithm for mining PHUIs using a level-wise approach in an uncertain database based on an Apriori-like approach and a designed upper-bound model. An improved algorithm named PHUI-List is proposed for discovering PHUIs directly without generating candidates based on a designed vertical data structure, named probability-utility-list (PU-list).
5. Substantial experiments have been conducted on both real-life and synthetic databases. Results show that the two proposed

Please cite this article as: J.C.-W. Lin et al., Efficient algorithms for mining high-utility itemsets in uncertain databases, Knowledge-Based Systems (2016), http://dx.doi.org/10.1016/j.knosys.2015.12.019

algorithms can effectively discover the complete set of PHUIs in an uncertain database. Specifically, the second algorithm outperforms the first one in terms of runtime, memory consumption, and scalability.

The remainder of this paper is organized as follows. Related work is briefly reviewed in Section 2. Preliminaries and problem statement related to PHUIM in uncertain databases are presented in Section 3. The two proposed algorithms, PHUI-UP and PHUI-List are respectively described in Section 4. An experimental evaluation assessing the performance of the proposed algorithms is presented in Section 5. Finally, a conclusion is drawn and future work is discussed in Section 6.

2. Related work

In this section, related work about mining frequent itemsets in uncertain databases and high-utility itemset mining are briefly reviewed.

2.1. Mining frequent itemsets in uncertain databases

Most approaches for mining frequent itemsets or association rules are designed to handle a binary database, where an item or itemset is either present or absent in each transaction. In real-life applications, a huge amount of collected data is inaccurate or incomplete (e.g. data from wireless sensor networks) [2,4]. Discovering frequent itemsets in an uncertain database has emerged as an important issue in recent years [4,13,22,25,32,36]. Depending on the specific applications, approaches to mining uncertain frequent itemsets in uncertain databases can be generally classified into two categories: those using the expected support-based model [3,13,22] and those based on the probabilistic frequency model [10,31].

Chui et al. first designed the UApriori algorithm [13] to mine frequent itemsets in uncertain databases using a level-wise approach based on the well-known Apriori algorithm. The expected support threshold was thus defined for mining frequent itemsets in an uncertain database. Leung et al. designed the UFP-growth algorithm [22] to mine uncertain frequent itemsets (UFIs) without candidate generation based on the extended frequent pattern (FP)-tree structure, and a divide-and-conquer and depth-first search approach. It was shown to outperform the UApriori algorithm. UH-mine was proposed by Aggarwal et al. [3]. It extends the H-Mine algorithm and also adopts a divide-and-conquer and depth-first search strategy to carry out recursive processing for mining UFIs using the UH-Struct structure. Wang et al. then presented the MBP algorithm [35], which generally explores a smaller number of candidates compared to the UApriori algorithm. Lin et al. then designed a compressed uncertain frequent pattern (CUFP)-tree structure to efficiently mine UFIs [23]. Most algorithms, such as UApriori [13], UFP-growth [22], UH-mine [3], and CUFP-Miner [23], 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 [10]. Sun et al. also developed a p-FP structure and proposed two efficient algorithms which discover frequent patterns in bottom-up (p-Apriori) and top-down (TODIS) manners [31]. The p-Apriori and TODIS algorithms were compared for mining probabilistic frequent itemsets by adopting dynamic programming or the divide-and-conquer strategy based on the probabilistic frequent model. Besides frequent itemset mining in uncertain databases, approaches for mining uncertain frequent itemsets in data streams have been proposed, such as UF-streaming [21] and SUF-growth [21]. Recently, Tong et al. combined the expected support-based model and probabilistic frequent model to present a new representation for mining frequent itemsets in uncertain databases with uniform measures [32]. The development of algorithms for mining frequent itemsets in uncertain databases is still in progress.

2.2. High-utility itemset mining

High-utility itemset mining (HUIM) is based on the measurement of local transaction utility and external utility to find itemsets with high profits. Chan et al. designed top-k high-utility closed patterns for deriving patterns by considering both positive and negative utilities [11]. A new strategy was also developed for pruning the low utility itemsets, thus speeding up the discovery of HUIs. Yao et al. defined the problem of utility mining by analyzing the utility relationships of itemsets [39]. The utility bound and support bound properties were defined, forming the mathematical model for mining HUIs. Since the downward closure property is no longer kept for HUIM, Liu et al. presented the Two-Phase model [27] to efficiently discover HUIs based on the designed TWU upper-bound and the TWDC property. Ahmed et al. proposed a tree-based IHUP model to build an effective HUP-tree structure for mining HUIs in incremental databases [9]. Lin et al. proposed the HUP-growth algorithm [24] to find HUIs without candidate generation. It adopts the Two-Phase model and the designed tree structure to maintain 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 mining process. Tseng et al. proposed the UP-tree structure and developed two mining algorithms, UP-growth [34] and UP-growth+ [33], to efficiently derive HUIs. Liu et al. proposed the HUI-Miner algorithm [26] to build utility-list structures and to develop a Set-enumeration tree to both prune the search space and directly extract HUIs without candidate generation, and without performing an additional database scan. Fournier-Viger et al. further designed the FHM algorithm by enhancing the pruning property of HUI-Miner by considering co-occurrences among 2-itemsets [14].

In recent years, many interesting issues in HUIM have been extensively studied, such as up-to-date HUI mining [20], mining HUIs in dynamic environment [19], mining HUIs in stream data [40], mining on-shelf HUIs [15], mining top-k HUIs [37,40] and mining HUIs with multiple minimum utility thresholds [18]. The development of other algorithms for HUIM is still in progress, but most of them are designed to handle precise databases. However, to the best of our knowledge, mining high-utility itemsets in uncertain databases has not yet been considered. This is the first work that studies the problem of mining potential high-utility itemsets (PHUIs) in uncertain databases.

3. Preliminaries and problem statement

This section first discusses which uncertain database model and which uncertainty calculation model is suitable for the addressed problem. Then, preliminaries and problem statement related to potential high-utility itemset mining (PHUIM) in an uncertain database are given.

3.1. The uncertain database model and probability measure

The probabilistic frequent itemsets (PFI) model [10,31] 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), where *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, rare rather than frequent. For example, an itemset having a high existing probability may be

Please cite this article as: J.C.-W. Lin et al., Efficient algorithms for mining high-utility itemsets in uncertain databases, Knowledge-Based Systems (2016), http://dx.doi.org/10.1016/j.knosys.2015.12.019

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 profit table.

Item	Profit ($)
A	4
B	1
C	12
D	6
E	15

infrequent in an uncertain database but it can yield a high profit, thus one may still be interested in this high probability itemset. Hence, the probabilistic frequent itemsets model is not suitable for PHUIM. In contrast, an itemset is considered frequent if its expected support (the sum of its probabilities) in an uncertain database exceeds *minSup* in the expected-support model [3,13,22], which is similar to the potential probability in our PHUIM framework.

As mentioned, utility and probability are two different measures in real-life applications. The proposed PHUIM model takes the two measures into account to respectively consider users' background knowledge and the objective measure of existence probability. PHUIM aims at discovering itemsets with high utility that occur with a high existence probability, and the probabilistic frequentness is not suitable for the PHUIM framework. The uncertainty model used in the PHUIM model is very close to the model used for probabilistic databases. A tuple uncertainty probabilistic database considers "tuple uncertainty" [29], where each tuple is associated with an existing probability, as in the proposed PHUIM model. The probability measure which is similar to the expected support-based frequent itemset mining model [3,13,22] is adopted in the proposed PHUIM framework, and we named this the potential probability measure. The patterns found by algorithms for the proposed PHUIM model have high potential probability of being HUIs in the uncertain database. Hence, in this paper, the tuple uncertainty database model [29] and the uncertainty calculation model which is somewhat similar to the expected support-based model [3,13,22] are adopted in the two proposed algorithms.

3.2. Preliminaries

Let $I = \{i_1, i_2, \ldots, i_n\}$ be a finite set of n distinct items in an uncertain quantitative database $D = \{T_1, T_2, \ldots, T_m\}$, 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 a unique identifier called its *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 [29]. A corresponding profit table, $ptable = \{pr_1, pr_2, \ldots, pr_n\}$, 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 profit table is shown in Table 2. In this example, the minimum utility threshold ε and the minimum potential probability threshold μ are respectively set to ε (= 25%) and μ (= 15%).

Definition 1. The utility of an item i_j in a transaction 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 $u(C, T_1) = q(C, T_1) \times pr(C) (= 3 \times 12) (= 36)$.

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, consider an item {*C*} and an itemset {*AC*} 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 a transaction T_q is denoted as $u(X, T_q)$, and defined as $u(X, T_q) = \sum_{i_j \in X \wedge X \subseteq T_q} u(i_j, T_q)$.

For example, the utility of itemset {*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 $u(X)$, and is 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 a possible world W_j (while taking into account the probability of W_j being the true world) over all possible worlds as $expSup(X) = \sum_{i=1}^{|D|} (\prod_{x_i \in X} p(x_i, T_q))$ [13,32]. The probability measure of the addressed PHUIM problem in an uncertain database is defined as follows.

Definition 5. The potential probability of an itemset X in D is denoted as $Pro(X)$, and 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 $tu(T_q)$, and defined as $tu(T_q) = \sum_{j=1}^{m} u(i_j, T_q)$, where 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 $tu(T_{10}) (= 53)$. Thus, the total utility in the running example is calculated as: $TU (= 74 + 13 + 63 + 24 + 30 + 62 + 44 + 19 + 60 + 53) (= 442)$.

Please cite this article as: J.C.-W. Lin et al., Efficient algorithms for mining high-utility itemsets in uncertain databases, Knowledge-Based Systems (2016), http://dx.doi.org/10.1016/j.knosys.2015.12.019

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

Definition 8. An itemset X in a database D is defined as a high-utility itemset (HUI) if its utility value $u(X)$ is no less than 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 to 25%. The item {*A*} is not considered as a HUI since $u(A)$ (= 36), which is smaller than (0.25 × 442) (= 110.5). The 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 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 to 15%, the minimum potential probability is calculated as (15% × 10) (= 1.5). Thus, the item {*A*} is a HPI since $Pro(A)$ (= 3.71 > 1.5). The itemset {*ABE*} is, however, not a HPI 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 following two conditions: (1) X is a HUI; (2) X is a HPI. It means that X has a high potential probability and is regarded as a HUI due to its high utility value.

Based on the above definitions, the problem statement of mining PHUIs in uncertain databases is formulated as follows.

3.3. Problem statement

Given an uncertain database D, its total utility TU, a user-specified minimum utility threshold ε, a user-specified minimum potential probability threshold μ, the problem of potential high-utility itemset mining (PHUIM) in uncertain databases is to mine the potential high-utility itemsets (PHUIs) whose utilities are larger than or equal to ($\varepsilon \times TU$), 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 to 25% and the minimum potential probability threshold is set to 15%.

4. The proposed algorithms for mining potential high-utility itemsets in uncertain databases

In this section, two efficient algorithms for mining PHUIs in uncertain databases are presented, named PHUI-UP and PHUI-List. The PHUI-UP algorithm is first proposed as a baseline algorithm to mine PHUIs using a level-wise approach in uncertain databases based on an Apriori-like [7] approach and the developed upper-bound model. PHUI-List is further designed as a more efficient algorithm to discover PHUIs directly in uncertain databases (without candidate generation) based on the designed probability-utility (PU)-list structure and a Set-enumeration tree.

4.1. Proposed PHUI-UP algorithm

To the best of our knowledge, this is the first paper to discuss PHUIM in 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 approach [6,7].

4.1.1. The pruning strategy by the downward closure property

In the well-known Apriori algorithm, the downward closure property of the support measure is employed to reduce the number of candidates for ARM. Similarly, the downward closure property of the proposed probability measure is used in the designed PHUI-UP algorithms for mining PHUIs. Details are given as follows.

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

Proof. Let X^k be a k-itemset, and X^{k-1} be one of its subsets. 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 a subset of X^k, the set of *TIDs* of X^k is a subset of the *TIDs* of X^{k-1}. Thus,

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

If X^k is a HPI, its potential probability is larger than or equal to the minimum potential probability count, that is $Pro(X^k) \geq |D| \times \mu$, 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 downward closure property of ARM cannot be directly extended to mine HUIs. The TWDC property [27] was proposed to reduce the search space in HUIM.

Definition 11. The transaction-weighted utility (*TWU*) of an itemset X in D is the sum of all transaction utilities $tu(T_q)$ containing 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 in D is considered as a high transaction-weighted utilization itemset (*HTWUI*) if its $TWU(X) \geq TU \times \varepsilon$.

For example, in Table 1, the TWU of 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). Item {*E*} is a *HTWUI* since $TWU(E)$ (= 283) (> 134).

The TWDC property is also extended to mine PHUIs in uncertain databases, and a novel transaction-weighted probabilistic and utilization downward closure (TWPUDC) property is further designed to reduce the search space of the PHUI-UP algorithm for mining PHUIs.

Definition 13. An itemset X in D is defined as a high transaction-weighted probabilistic and utilization itemset (*HTWPUI*) if (1) $TWU(X) \geq \varepsilon \times TU$, and (2) $Pro(X) \geq \mu \times |D|$.

For example, in Tables 1 and 2, since μ is set to 15%, the minimum potential probability is calculated as (15% × 10) (= 1.5). Consider item {E} as 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 item {E} is a HTWPUI.

Theorem 2 (**Downward closure property of HTWPUI**). *Let X^k and X^{k-1} be HTWPUIs in an uncertain database, such that $X^{k-1} \subseteq X^k$. Then, the downward closure property of HTWPUIs 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 X^k be one of its superset. Since $X^{k-1} \subseteq X^k$,

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

From **Theorem 1**, it can be found that $Pro(X^{k-1}) \geq Pro(X^k)$; 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 ⊆ HTWPUIs**). *The transaction-weighted probabilistic and utilization downward closure (TWPUDC) property ensures that PHUIs ⊆ HTWPUIs. It indicates that if an itemset is not a HTWPUI, then none of its supersets are PHUIs.*

Proof. For $\forall X^{k-1} \in D$, note that X^{k-1} is a (k-1)-itemset.

(1) According to **Theorem 2**, it can be found that $TWU(X^{k-1}) \geq TWU(X^k)$ and $Pro(X^{k-1}) \geq Pro(X^k)$, if X^{k-1} is not a HTWPUI, none of X^{k-1} its supersets X^k will be HTWPUIs.

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

Based on definitions 10 and 13, if X^{k-1} is not a HTWPUI, it means that X^{k-1} does not satisfy the following two conditions $TWU(X^{k-1}) \geq \varepsilon \times TU$ and $Pro(X^{k-1}) \geq \mu \times |D|$; X^{k-1} would not be a PHUI. Thus, X^{k-1} and any of its supersets X^k are HTWPUIs, and they also would not be PHUIs. Thus, this theorem holds.

4.1.2. The PHUI-UP algorithm

The proposed PHUI-UP 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 scan. The TWPUDC property inherits the TWDC property of the Two-Phase model to preserve the downward closure property, thus reducing the search space for finding PHUIs. Only the remaining HTWPUI^{k-1} are used to generate the next C_k at each level.

Based on the designed TWPUDC property, **Theorem 3** ensures that the proposed PHUI-UP algorithm can make sure that no supersets of non-HTWPUIs are in the preliminary candidate set (correctness) and that it can extract the complete set of PHUIs from the derived candidate HTWPUIs (completeness). Therefore, the proposed PHUI-UP algorithm is correct and complete. Based on the above definitions and theorems, details of the proposed PHUI-UP algorithm are given below.

As shown in Algorithm 1, the proposed PHUI-UP algorithm first scans the database to find the *TU*, TWU values and the probabilities of all 1-itemsets in the database (Line 1). According to the designed conditions of HTWPUI^k (Line 3), the HTWPUI^k (where k is initially set to 1) are then produced (Lines 2 to 6) and will be further used to generate the next candidates C_{k+1} for discovering HTWPUI^{k+1} in a level-wise way (Lines 8 to 17). In this process, the original database has to be rescanned to find the set of HTWPUI^{k+1} in each level C_{k+1} (Line 10). The first phase of PHUI-UP terminates when no candidate is generated. Then the algorithm performs the second phase. In this phase, an additional database scan is required to find the final PHUIs from the HTWPUIs (Lines 19 to 24).

Algorithm 1 PHUI-UP

INPUT: D, an 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 *TU* and obtain $TWU(i_j)$ and $Pro(i_j)$ for each $i_j \in D$.
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. $HTWPUI^1 \leftarrow HTWPUI^1 \cup i_j$;
5. **end if**
6. **end for**
7. set $k \leftarrow 2$;
8. **while** $HTWPUI^{k-1} \neq \emptyset$ **do**
9. $C_k = Apriori\_gen(HTWPUI^{k-1})$;
10. **for** each k-itemset $X \in C_k$ **do** /* scan D once for all k-itemset */
11. calculate $TWU(X)$ and $Pro(X)$;
12. **if** $TWU(X) \geq TU \times \varepsilon \wedge Pro(X) \geq |D| \times \mu$ **then**
13. $HTWPUI^k \leftarrow HTWPUI^k \cup X$;
14. **end if**
15. **end for**
16. $k \leftarrow k+1$;
17. **end while**
18. $HTWPUIs \leftarrow HTWPUIs \cup HTWPUI^k$;
19. **for** each k-itemset $X \in HTWPUIs$ **do** /* scan D for all candidates */
20. calculate the actual utility for $X := u(X)$;
21. **if** $u(X) \geq TU \times \varepsilon$ **then**
22. $PHUIs \leftarrow PHUIs \cup X$;
23. **end if**
24. **end for**
25. **return** *PHUIs*;

Complexity analysis. Let m be the number of transactions in a database D, and n be the number of items in the largest transaction in D, and L_k be the set of HTWPUI^k. The first step of the PHUI-UP algorithm is to calculate the *TU*, and the $TWU(i)$ and $Pro(i)$ value of each item i in D. This requires a single database scan, and thus $O(m \times n)$ time. Then, for each level C_k, candidates are generated by applying the *Apriori_gen* subprocedure [7]. This procedure combines pairs of itemsets in L_{k-1} to generate the itemsets in C_k in $O(|L_k| \times |L_k|)$ time. Then, the algorithm scans the whole database once to calculate the information of each k-itemset in C_k. This can be done by reading the database once and comparing each itemset in C_k with each transaction, which requires $O(m \times n \times |C_k|)$ time. The algorithm finishes the first phase when a level is empty, and in the worst case, there are n levels. Finally, the database is scanned once more to calculate the exact utility of all *HTWPUIs*, in $O(m \times n \times |HTWPUIs|)$ time. Thus, in the worst case, the overall time complexity of the PHUI-UP algorithm is $O(m \times n + \sum_{k \geq n} (|L_k| \times |L_k| + m \times n \times |C_k|) + m \times n \times |HTWPUIs|)$time. Since this approach uses an Apriori-based approach for generating candidates at each level, it may suffer from a high space and time complexity.

4.1.3. An example of the PHUI-UP algorithm

To preserve consistency, consider the running example of Tables 1 and 2. Moreover, assume that the minimum utility threshold is set to 25% and that the minimum potential probability threshold is set to 15%. Thus, the minimum support count and the minimum potential probability are respectively calculated as (442

Table 4
The derived HTWPUIs 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

× 25%) (= 110.5) and (10 × 15%) (= 1.5). The proposed PHUI-UP algorithm is applied 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 result is {*A*: 303, 3.71; *B*: 222, 3.71; *C*: 336, 4.36; *D*: 209, 3.41; *E*: 283, 3.91}, where {*A*: 303, 3.71} indicates that the TWU value of itemset {*A*} is 303 and its probability value is 3.71. Since all 1-itemsets satisfies $TWU(X) \geq 110.5$ and $Pro(X) \geq 1.5$, they are all put into the set of potential high transaction-weighted utilization itemsets (HTWPUIs). Then k is set to 2, and the PHUI-UP algorithm generates the candidates C_2 by applying *Apriori_gen*(*HTWPUI*[1]) using the alphabetical order of items. The set C_2 is {*AB*; *AC*; *AD*; *AE*; *BC*; *BD*; *BE*; *CD*; *CE*; *DE*}. The uncertain database is then rescanned to calculate the *TWU* and *Pro* values of each itemset in C_2. The result 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 these, only the itemsets {*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 $TWU(X) \geq 110.5$ and $Pro(X) \geq 1.5$. These itemsets are thus added to the set of HTWPUIs. Then k is set to 3 and the procedure is applied in the same way. The first phase terminates when no candidate is generated. After the first phase, the complete set of HTWPUIs is shown at Table 4.

Then the PHUI-UP algorithm performs the second phase. An additional database scan is done to find the final PHUIs in the set of HTWPUIs by calculating their actual utility values. Consider itemsets {*A*} and {*AC*} in Table 4 as an example. According to definition 4, $u(A) = 36 < 110.5$ and $u(AC) = 140 > 110.5$, so the itemset {*A*} is not a PHUI, while the itemset {*AC*} is regarded as a PHUI. Finally, the complete set of PHUIs is discovered by the PHUI-UP algorithm. The result is shown at Table 3.

4.2. Proposed PHUI-List algorithm

In the past, HUI-Miner [26] was proposed to efficiently mine HUIs in precise databases. Experiments using HUI-Miner have shown that it outperforms the other state-of-the-art HUIM algorithms. In this section, an efficient PHUI-List algorithm is proposed to more efficiently mine PHUIs. A new vertical data structure, called probability-utility-list (PU-list), is designed to store information required by PHUI-List during the mining process. Thus, using this data structure, PHUI-List can directly mine PHUIs without repeatedly scanning the database, in contrast with the PHUI-UP algorithm. Details of the presented PU-list structure, the Set-enumeration tree for the search space, and the proposed PHUI-list algorithm are given below.

Algorithm 2 PU-list construction procedure

INPUT: X, the PU-list of an itemset X; X_a, the extension of X with an item a; X_b, the extension of X with an item b ($a \neq b$, $a \prec b$).
OUTPUT: $X_{ab}.PUL$, the PU-list of X_{ab}.
1. set $X_{ab}.PUL \leftarrow \emptyset$;
2. **for** each element $E_a \in X_a$ **do**
3. **if** $\exists E_b \in X_b.PUL \wedge E_a.tid = E_b.tid$ **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. $X_{ab}.PUL \leftarrow X_{ab}.PUL \cup E_{ab}$;
11. **end if**
12. **end for**
13. **return** $X_{ab}.PUL$;

4.2.1. Probability-Utility-list (PU-list) structure

The PU-list structure is inspired by the utility-list structure. It keeps information from transactions in memory to directly mine PHUIs. Each entry in the PU-list of an itemset X represents a transaction T_q containing X and consists of the corresponding *TID* (***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***). In the following, for two items/itemsets X and Y, the notation $X \prec Y$ means that X precedes Y according to the total order on items used by the mining process (e.g. alphabetical order or TWU ascending order). The extension of an itemset X is that $\{X \cup i\}$ where i is a 1-item.

Definition 14. Give an itemset X and a transaction (or itemset) T such that $X \subseteq T$, the set of all items from T that are not in X is denoted as $T \backslash X$, and the set of all the items appearing after X in T is denoted as T/X. Thus, $T/X \subseteq T \backslash X$.

For example, consider $X = \{BC\}$ and the transaction T_3 in Table 1, $T_3 \backslash X = \{AE\}$, and $T_3/X = \{E\}$.

Definition 15. An entry in the PU-list of an itemset X representing a transaction T_q consists of four fields: (1) the *TID* of X in T_q ($X \subseteq T_q \in D$), (2) the probabilities of X in T_q (***prob***), (3) the utilities of X in T_q (***iu***), and (4) the remaining utilities of X in T_q (***ru***), where ***iu*** is defined as $X.iu = \sum_{i \in X \wedge X \subseteq T_q} u(i, T_q)$ for each transaction T_q, and ***ru*** is defined as $X.ru = \sum_{i \notin X \wedge i \in T_q \wedge X \prec i} u(i, T_q)$, for each transaction T_q.

The construction of PU-lists is performed by a specific procedure shown in Algorithm 2. This construction procedure is applied for each k-itemset encountered in the search space.

The PU-list of a k-itemset ($k > 1$) is constructed using the PU-lists of some of its subsets of length $k-1$. Note that it is necessary to initially construct the PU-lists of itemsets in HTWPUI[1] to recursively obtain the PU-lists of larger itemsets. In the running example, the 1-itemsets in HTWPUI[1] are {*A*: 303, 3.71; *B*: 222, 3.71; *C*: 336, 4.36; *D*: 209, 3.41; *E*: 283, 3.91}, where {*A*: 303, 3.71} indicates that the itemset {*A*} has a TWU value of 303 and a probability value of 3.71. PU-lists are constructed in ascending order of their TWU values ($D \prec B \prec E \prec A \prec C$), as shown in Fig. 1.

Definition 16. Based on the PU-list structure proposed in definition 15, let *X.IU* denotes the sum of the utilities of an itemset X in D, which can be defined as:

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

Definition 17. Based on the designed PU-list structure proposed in definition 15, let *X.RU* be the sum of the remaining utilities of an itemset X in D, which can be defined as:

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

{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

Fig. 1. The constructed PU-lists of HTWPUI[1].

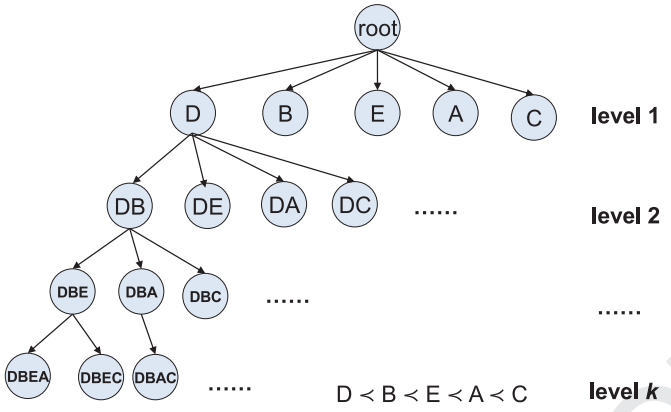

Fig. 2. A Set-enumeration tree.

Consider itemset {*A*} as example to illustrate these definitions. The itemset {*A*} exists in transactions having *TIDs* {1, 3, 6, 7, 9}. *A.IU* is calculated as $(8 + 4 + 8 + 4 + 12) = 36$, and *A.RU* is calculated as $(36 + 12 + 24 + 0 + 36) = 108$. Consider itemset {*AE*} which exists in transactions with *TIDs* = {1, 3, 7}. $AE.IU = A.IU + E.IU = (8 + 4 + 8) + (30 + 45 + 15) = 110$, and $AE.RU = A.RU + E.RU = (36 + 12 + 0) + (44 + 16 + 4) = 112$.

4.2.2. A set-enumeration tree

Based on the PU-list structure, the search space of the proposed PHUI-List algorithm can be represented as a Set-enumeration tree, where the ascending order of TWU values of HTWPUIs[1] is used by the mining process ($D \prec B \prec E \prec A \prec C$). The set-enumeration tree is defined as follows. Each node in the tree represents an itemset, which is the extension of its parent node. The illustrated Set-enumeration tree of the given example is shown in Fig. 2.

The PHUI-List algorithm explores the Set-enumeration tree using a depth-first search. For each considered node in the tree, the sum of ***iu*** and ***ru,*** *and* ***prob***, are calculated and an early pruning strategy is applied to decide whether the extensions of the processed node need to be explored. If the processed node satisfies the following two conditions: (1) the sum 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 extensions (descendant nodes) of the processed node will be explored recursively (more details about the pruning strategy will be given later). Finally, the actual utility of the processed node can be directly determined by its ***tid***, ***iu***, and the ***prob*** in its PU-List structure, for discovering PHUIs directly, that is without performing multiple database scans. For the construction of the Set-enumeration tree, the following lemmas hold.

Remark 1. *The search space of the proposed PHUI-List algorithm can be represented as a Set-enumeration tree, using the ascending order of TWU values of HTWPUIs[1] .*

Proof. Based on the construction of the Set-enumeration tree, as shown in Fig. 2, when traversing the enumeration tree based on a breadth-first search strategy, from top-to-down, the nodes in level 1 are first processed, then nodes in level 2 are processed, until level *k*. This is similar to the PHUI-UP algorithm which applies an Apriori-like level wise and generate-and-test approach. Hence, the Set-enumeration tree can represent the complete search space of the proposed PHUI-List algorithm. In addition, the search space of the Set-enumeration tree is the same no matter if a breadth-first search or depth-first search strategy is used.

4.2.3. Early pruning strategies

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

Remark 2. *The sum of all the probabilities of any node in the Set-enumeration tree is greater than or equal to the sum of all the probabilities of any one of its descendant nodes.*

Proof. Let X^{k-1} be a node in the Set-enumeration tree, and X^k be a children (extension) of X^{k-1}. According to **Theorem 2**, we can get $Pro(X^{k-1}) \geq Pro(X^k)$. Thus, the property holds.

Pruning strategy 1. *When traversing the Set-enumeration tree using 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 descendant nodes of node X is a PHUI.*

Remark 3. According to **Remark 2**, we can observe that if the sum of all the probabilities of a node is less than the minimum potential probability, this node is not a PHUI, as well as all its descendant nodes (extensions). Hence, after the sum of all the probabilities of a node have been calculated, the probabilities of unpromising itemsets and their extensions can be regarded as irrelevant and be pruned directly.

Lemma 1. *For any node X in the Set-enumeration tree, the sum of X.IU and X.RU is greater than or equal to the sum of all the utilities of any one of its descendant nodes (extensions).*

Proof. Let be a node X in the Set-enumeration tree, and X' denotes any descendant node (extensions) of X. For all transaction $T_q \supseteq X$:

$$\because X' \text{ is an extension of } X \Rightarrow (X' - X) = (X'/X)$$

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

$$\therefore \text{in } T_q,\ X'.iu = X.iu + (X'/X).iu$$
$$= X.iu + \sum_{z \in (X'/X)} z.iu$$
$$\leq X.iu + \sum_{z \in (T_q/X)} z.iu$$
$$= X.iu + X.ru;$$
$$\therefore \text{in } T_q, X'.iu \leq X.iu + X.ru$$
$$\because X \subset X' \Rightarrow X'.tids \subseteq X.tids$$
$$\therefore in\ D,\ X'.IU = \sum_{T_q \in X'.tids} X'.iu$$
$$\leq \sum_{T_q \in X'.tids} (X.iu + X.ru)$$
$$\leq \sum_{T_q \in X.tids} (X.iu + X.ru)$$
$$= X.IU + X.RU.$$
$$\therefore \text{in } D, X'.IU \leq X.IU + X.RU.$$

Pruning strategy 2. *When traversing the Set-enumeration tree using a depth-first search, if the sum of X.IU and X.RU in the constructed PU-list is less than the minimum utility count, then none of the descendant nodes (extensions) of node X is a PHUI.*

Remark 4. According to **Remark 2**, we can observe that if the sum of the *X.IU* and *X.RU* of a node is less than the minimum utility count ($\varepsilon \times TU$), the node can be regarded as an unpromising PHUI, as well as all its descendant nodes (extensions). Thus, after the sum of all the utilities of a node has been calculated, those utilities of unpromising itemsets and their extensions can be regarded as irrelevant and be pruned directly.

Remark 5. *With the two proposed pruning strategies, the actual number of nodes explored in the Set-enumeration tree (denoted as N_1) is always smaller than or equal to the number of candidate itemsets (w.r.t HTWPUIs, denoted as N_2) generated by the PHUI-UP algorithm. This indicates that the two early pruning strategies used in PHUI-List allows obtaining a more compressed search space than using the TWPUDC property which is applied in PHUI-UP.*

Proof. By **Lemma 1** and **Remark 2**, it can be seen that the remaining utility of an itemset in each transaction T_q, which is kept in the PU-list, is smaller than the *TU* value of T_q. Therefore, for any itemset *X*, the sum of *X.IU* and *X.RU* is generally much smaller than its TWU value. By providing tighter upper-bounds, the two early pruning strategies used in PHUI-List can efficiently prune more unpromising itemsets than PHUI-UP, which results in a more compressed search space than using the TWPUDC property. Thus, $N_1 \leq N_2$.

Remark 6. *The PHUI-List algorithm is correct and complete,* that is it discovers all PHUIs and only the PHUIs.

Proof. The PHUI-List algorithm first scans every transaction in the uncertain database given as input. For each promising item, its information ***tid***, ***prob***, ***iu***, and ***ru*** is extracted to construct its PU-list structure. When traversing the Set-enumeration tree to find PHUIs, each node can be evaluated using its PU-list. By **Lemma 1**, and **Remarks 1** and **2**, the PHUI-List algorithm ensures that no promising itemset is discarded (completeness) and all required information about an itemset can be obtained exactly from its PU-list structure (correctness). Therefore, it can be said that based on the PU-list structure and Set-enumeration tree, the proposed PHUI-List algorithm is correct and complete.

Based on the two proposed pruning strategies, the designed PHUI-List algorithm can prune itemsets with low potential probability count or utility count early, without constructing their PU-lists as well as the PU-lists of their extensions. This can effectively reduce both the cost of utility-list construction and the search space of the Set-enumeration tree.

Algorithm 3 PHUI-List algorithm

INPUT: *D*, an uncertain transactional database; *ptable*, profit table; ε, a user-specified minimum utility threshold; μ, a user-specified minimum potential probability threshold.
OUTPUT: The set of potential high-utility itemsets (PHUIs).
1. scan *D* to calculate *TU* and the *TWU*(i) and *Pro*(i) value of each item $i \in I$;
2. find $I^* \leftarrow \{i \in I \mid TWU(i) \geq TU \times \varepsilon \wedge Pro(i) \geq |D| \times \mu\}$;
3. sort I^* in *TWU* ascending order;
4. scan *D* once to build the PU-list of each 1-item $i \in I^*$;
5. call **PHUI-Search(*ø, I*, ε, μ*)**;
6. **return** *PHUIs*;

Algorithm 4 PHUI-Search Procedure

INPUT: *X*, an itemset; *extendOfX*, a set of PU-lists of all 1-extensions of itemset *X*; ε, minimum utility threshold; μ, minimum potential probability threshold.
OUTPUT: The set of potential high-utility itemsets (PHUIs).
/* *X.PUL*, the PU-list of an itemset *X* */
1. **for** each itemset $X_a \in extendOfX$ **do**
2. obtain the $X_a.IU$, $X_a.RU$ and $Pro(X_a)$ values from the built $X_a.PUL$;
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** $(X_a.IU + X_a.RU \geq TU \times \varepsilon) \wedge (Pro(X_a) \geq |D| \times \mu)$ **then**
7. $extendOfX_a \leftarrow \emptyset$;
8. **for** each itemset $X_b \in extendOfX$ such that $a \prec b$ **do**
9. $X_{ab} \leftarrow X_a \cup X_b$;
10. $X_{ab}.PUL \leftarrow$ **Construct(*X, Xa, Xb*)**;
11. $extendOfX_a \leftarrow extendOfX_a \cup X_{ab}$;
12. **end for**
13. call **PHUI-Search(*Xa, extendOfXa, ε, μ*)**;
14. **end if**
15. **end for**
16. **return** *PHUIs*;

4.2.4. The PHUI-List algorithm

Based on the above definitions, remarks, lemmas, and the pseudo-code of the proposed PHUI-List algorithm are given below.

As shown in the main procedure of PHUI-List (Algorithm 3), the proposed PHUI-List algorithm first scans the uncertain database to calculate the *TU*, and the *TWU*(i) and *Pro*(i) value of each item $i \in I$ (Line 1), and then find the potential high transaction-weight utilization 1-itemsets (I^* w.r.t. *HTWPUI*[1]) (Line 2). After items in I^* are sorted by the *TWU* ascending order, the PHUI-List algorithm scans *D* again to construct the PU-list of each 1-item in *HTWPUI*[1] w.r.t. $i \in I^*$ (Lines 2 to 4). Each 1-item $i \in I^*$ and its associated PU-list is recursively processed by using the depth-first search procedure **PHUI-Search** (Line 5). Details of the **PHUI-Search** procedure are given below.

The **PHUI-Search** procedure shown in Algorithm 4 considers each itemset X_a to directly produce PHUIs (Lines 2 to 5). Two pruning strategies (Line 6) are applied to further determine whether extensions of X_a satisfy the designed conditions for executing the later depth-first search (Lines 6 to 14). The construction process **Construct(*X, Xa, Xb*)** is executed to construct the PU-lists of all 1-extensions of itemset X_a (w.r.t. $extendOfX_a$) (Lines 7 to 12). Note that each constructed itemset X_{ab} is a 1-extension of itemset X_a. Thus, it should be put into the set $extendOfX_a$ for executing the later depth-first search (Line 11). The designed **PHUI-Search** procedure is then recursively performed to mine PHUIs that are extensions of X_a (Line 13). Based on the designed PU-list structure,

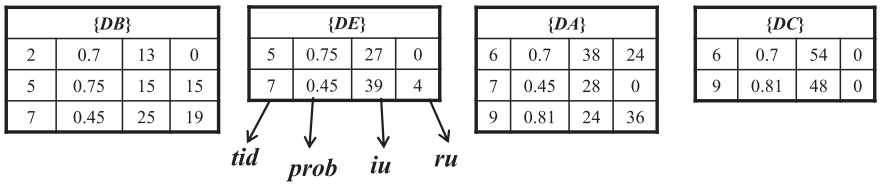

Fig. 3. The constructed PU-lists of HTWPUI[2] having {*D*} as prefix.

the PHUI-List algorithm can thus directly mine the complete set of PHUIs in the uncertain database without candidate generation.

Complexity analysis. The complexity of the algorithm is analyzed as follows. The first step of the PHUI-List algorithm consists of calculating the *TU*, and the *TWU*(i) and *Pro*(i) values of each item $i \in I$. This requires a single database scan, and thus takes $O(m \times n)$ time in the worst case. Then, the items in I^* (the *HTWPUI*[1]) are sorted in ascending order of TWU values in $O(nlogn)$ time. Then, PU-lists of items in I^* are constructed. There is $|I^*|$ PU-lists, each having in the worst case an entry with four fields for each transaction, thus requiring $O(|I^*| \times m)$ space. Building these initial PU-lists requires a database scan, that is $O(m \times n)$ time. Constructing PU-lists of a k-itemset ($k > 1$) using the Construct procedure can be performed in linear time if implemented as a two-way or three-way search [26]. The largest itemset in the Set-enumeration tree of the PHUI-List algorithm is equal to the length of the longest transaction in the database: $maxL = max\{|T_q|, \forall T_q \in D\}$. In the worst case, the database contains a transaction containing all items, and $max\{|T_q|, T_q \in D\} = n$, so $maxL = n$. In that case, there is up to $2^n - 1$ itemsets to be considered in the search space, and the time complexity of exploring the Set-enumeration tree is $O(2^n)$. Therefore, the worst case time complexity of Algorithm 4 is in theory $O(2^n)$. However, in real-life situations, transactional databases are rarely very sparse or dense. Thus, the largest itemset in the Set-enumeration tree generally contains few items compared to n. On overall, the worst case space complexity of the PHUI-List algorithm is $O(m \times n + nlogn + m \times n + 2^n)$, which requires approximately $O(2^n)$ time in the worst case. Note that by using a depth-first search, the proposed PHUI-List algorithm mines PHUIs at the same time as it explores the Set-enumeration tree and create the PU-list of each tree node, rather than building the whole tree in advance. Moreover, the algorithm can generally avoid exploring a large part of the search space thanks to its pruning strategies (depending on how thresholds are set). As a result, the practical runtime of the PHUI-List algorithm is smaller than the runtime of the PHUI-UP algorithm, as it will be demonstrated in the experimental evaluation, and in practice the PHUI-List algorithm does not suffer from the problem of combinatorial explosion.

In terms of space, the PU-list of each itemset is created once. Each PU-list requires at most $O(m)$ space, if it contains an entry for each transaction. But in practice, as larger itemsets are explored (k is increased), the size of PU-lists becomes smaller, as larger itemsets generally occur in less transactions.

4.2.5. An example of the PHUI-List algorithm

For the sake of consistency, consider the same database and thresholds used in Section 4.1.3. The PHUI-List algorithm first scans the original uncertain database once to extract the necessary information (i.e., ***tid***, ***prob***, ***iu*** and ***ru***) for each item in the database. Then, the PU-list structures of items are constructed. If an item X satisfies $TWU(X) \geq \varepsilon \times TU$ and $Pro(X) \geq \mu \times |D|$, it is added to the set of potential high transaction-weighted utilization itemsets (HTWPUIs). The final set of HTWPUIs[1] is shown in Fig. 1, sorted by ascending order of TWU values. The item {*D*} is first processed, and it is found that {*D*} does not satisfy the PHUI conditions: $D.IU = (12 + 12 + 30 + 24 + 12) (= 90) < (442 \times 25\%) (= 110.5)$, and $Pro(D) = (0.7 + 0.75 + 0.7 + 0.45 + 0.81) (= 3.41) > (10 \times 15\%) (= 1.5)$. Thus, the item {*D*} is considered as not being a PHUI, but $(D.IU + D.RU) = (90 + (1 + 18 + 32 + 20 + 12)) (= 119)$ is larger than the minimum utility count, indicating that its descendant nodes may be PHUIs, and thus the depth-first search from node {*D*} is performed. By applying the PU-list construction procedure, the PU-lists of 2-itemsets that are HTWPUIs[1] and have {*D*} as prefix are constructed, as shown in Fig. 3.

Then, the first child node {*DB*} is considered. It is found using the PU-list structure of {*DB*} that the sum of utilities and the sum of probabilities of itemset {*DB*} are respectively $DB.IU = (13 + 15 + 25) = 53 < 110.5$, and $Pro(DB) = (0.7 + 0.75 + 0.45) = 1.9 > 1.5$. Therefore, node {*DB*} and all its extensions (descendant nodes) are not PHUIs and can be pruned, and the depth-first is not performed for child nodes of {*DB*}. The next child node{*DE*} is then processed in the same way, as well as all the other nodes. After having considered all relevant nodes of the Set-enumeration tree, the complete set of PHUIs has been obtained by the PHUI-List algorithm without scanning the database multiple times. The final result is shown in Table 3.

5. Experimental results

In this section, substantial experiments are conducted to evaluate the two proposed algorithms, PHUI-UP and PHUI-List, for mining PHUIs in uncertain datasets, in terms of runtime, memory consumption, number of discovered patterns, and scalability. Notice that this is the first work considering the discovery of potential high-utility itemsets, and that the state-of-the-art HUI-Miner and FHM algorithms were shown to significantly outperform the previous well-known traditional HUIM algorithms, such as Two-Phase [27], IHUP [9], HUP-growth [24], UP-growth [34], and UP-growth+ [33]. Thus, the performance of the designed PHUI-UP and PHUI-List algorithms are only compared with HUI-Miner and FHM. Furthermore, this paper compares the derived PHUIs with the traditional HUIs obtained by the HUI-Miner and FHM algorithms. This pattern analysis comparison between PHUIs and HUIs is done to assess whether the proposed PHUIM framework is acceptable.

All the algorithms were implemented in Java and experiments were carried out on a personal computer equipped with an Intel Core i5-3460 dual-core processor and 4 GB of RAM, running the 32-bit Microsoft Windows 7 operating system. The source code of the HUI-Miner and FHM algorithms can be downloaded from the SPMF data mining library (http://www.philippe-fournier-viger.com/spmf/) [30]. Experimental results are presented and discussed thereafter.

5.1. Datasets

Both real-life and synthetic datasets were used in the experiments. Three real-life datasets, namely foodmart [28], accident [1], and retail [1], as well as a 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 containing transactions of products bought by customers at a chain store. The

Table 5
Parameters of used datasets.

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

Table 6
Characteristics of the datasets.

Datasets	#|D|	#|I|	AvgLen	MaxLen	Type
foodmart	21,556	1559	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

accident dataset contains anonymous data about traffic accidents on public roads of Belgium. The retail dataset contains anonymous retail market basket data from a Belgian retail store. The synthetic dataset T10I4D100K was generated using the IBM Quest Synthetic Data Generator [8]. Both quantity (internal) and profit (external) values were assigned to the items in the accident, retail, and T10I4D100K datasets using Liu's simulation model [27], whereas the foodmart dataset contains real values. In addition, due to the tuple uncertainty model, a unique probability value in the range of 0.5 to 1.0 is assigned to each transaction in these datasets. Parameters and characteristics of these datasets are respectively shown in Tables 5 and 6.

5.2. Runtime

The runtimes of the two proposed algorithms were compared with the state-of-the-art HUIM algorithms on the four uncertain datasets for various minimum utility thresholds (MUs) and minimum potential probability thresholds (MPs). Fig. 4 shows the runtimes of the two proposed algorithms and the state-of-the-art HUIM algorithms for different MUs and a fixed MP. Fig. 5 shows the runtimes of the two proposed algorithms for various MPs and a fixed MU value. Note that the datasets processed by HUI-Miner and FHM do not contain probability values. In other words, only precise versions of the datasets are used by HUI-Miner and FHM.

From Fig. 4, it can be observed that the proposed PHUI-List algorithm outperforms the baseline 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, and is always faster than the state-of-the-art HUI-Miner and FHM algorithms for HUIM. This indicates that the generate-and-test approach has worse performance than the proposed depth-first search approach that utilizes the vertical PU-list structure and additional pruning strategies. For example, for the accidents dataset in Fig. 4(c), the MP was set to 1% and the MU was varied from 0.2% to 1.0%, with a 0.2% increment. The runtime of PHUI-UP decreased dramatically from 822 to 562 s, while that of PHUI-List decreased steadily from 80 to 66 s. The reason is that when MU is set lower, the PHUI-UP algorithm may generate HTWPUIs that are longer (before the PHUIs are identified from HTWPUIs), and more computation is thus needed to perform the generate-and-test mechanism and calculate the upper bounds, especially for dense datasets. In contrast, the PHUI-List algorithm directly determines PHUIs from the Set-enumeration tree without generating candidates. This effectively avoids time-consuming database scans. Moreover, the PHUI-List algorithm applies two pruning strategies to effectively prune unpromising items early, based on the designed PU-list structure, thus greatly reducing the runtime for discovering PHUIs.

As shown in Fig. 5, it can also be observed that the PHUI-List algorithm outperforms the PHUI-UP algorithm under various MPs for the four datasets. Specifically, the runtime of PHUI-UP decreased sharply as the MP was increased, while the runtime of the PHUI-List algorithm decreased steadily, and the runtime of the

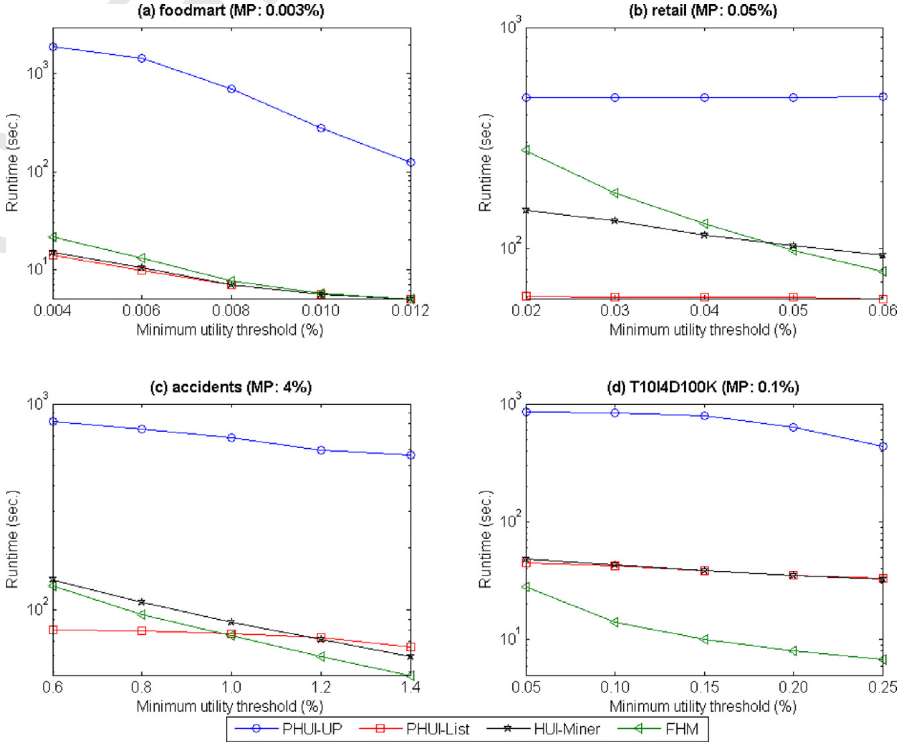

Fig. 4. Runtimes of the compared algorithms for various MUs and a fixed MP.

Please cite this article as: J.C.-W. Lin et al., Efficient algorithms for mining high-utility itemsets in uncertain databases, Knowledge-Based Systems (2016), http://dx.doi.org/10.1016/j.knosys.2015.12.019

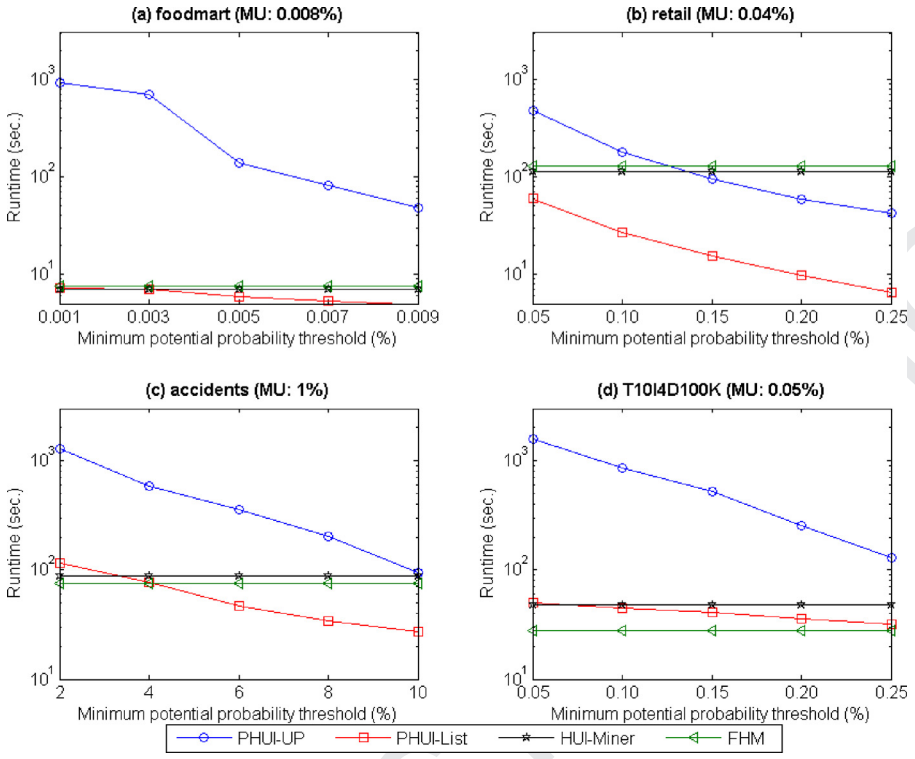

Fig. 5. Runtime of the compared algorithms for various MPs and a fixed MU.

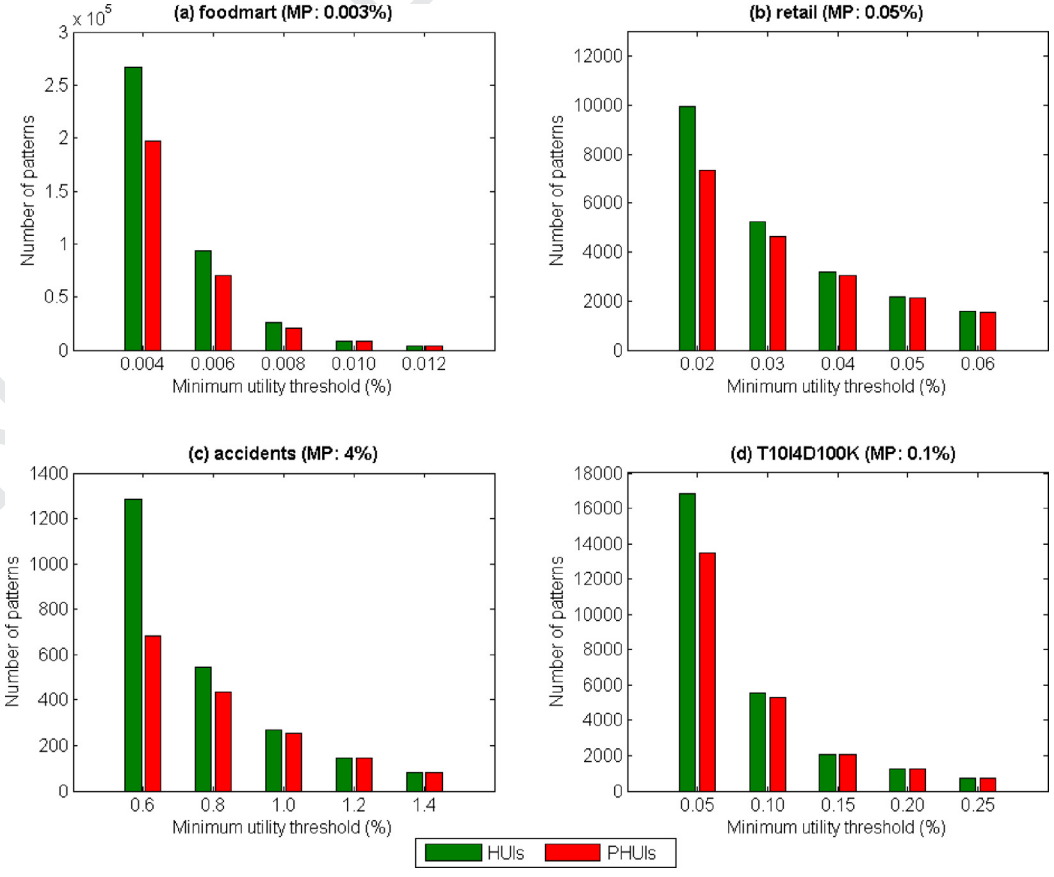

Fig. 6. Number of HUIs and PHUIs under various MUs and a fixed MP.

Please cite this article as: J.C.-W. Lin et al., Efficient algorithms for mining high-utility itemsets in uncertain databases, Knowledge-Based Systems (2016), http://dx.doi.org/10.1016/j.knosys.2015.12.019

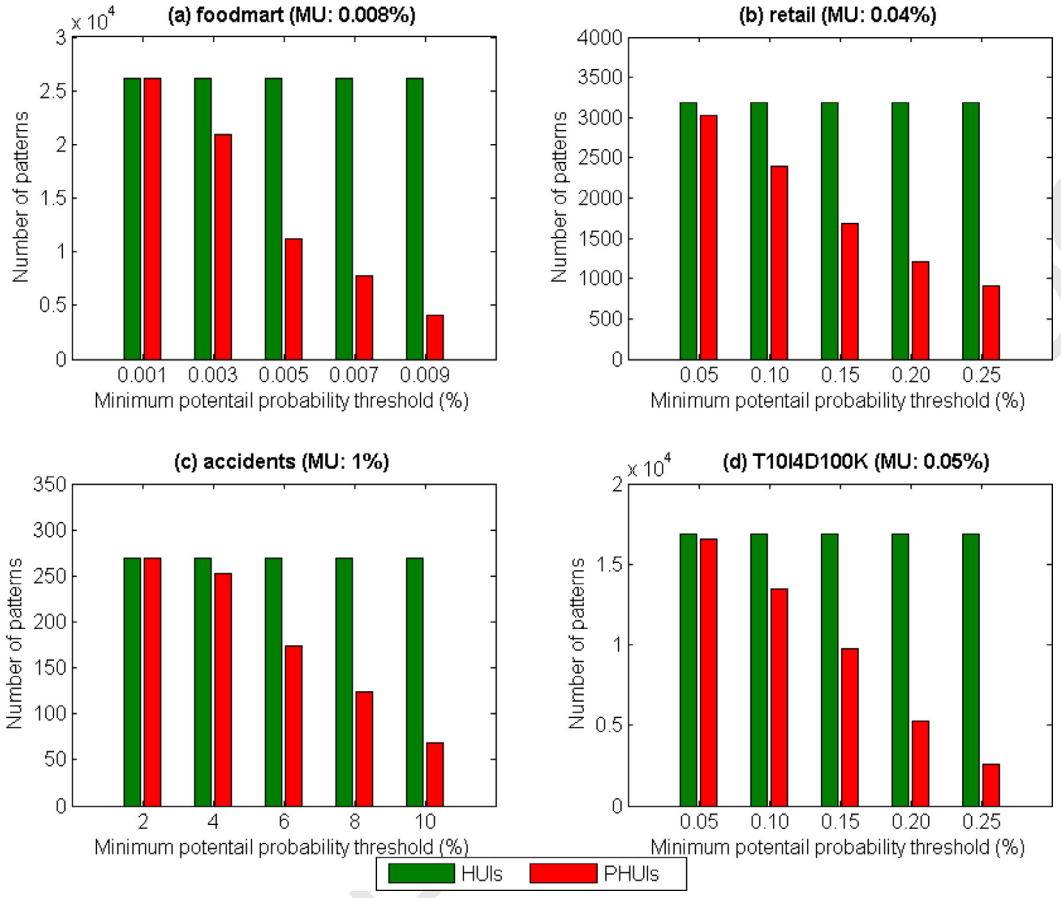

Fig. 7. Number of HUIs and PHUIs under various MPs and a fixed MU.

HUI-Miner and FHM algorithms remained unchanged. This result is reasonable since when the MP is set higher, fewer candidates are generated by the PHUI-UP algorithm to mine the PHUIs, while the performance of the traditional HUI-Miner and FHM algorithms are not affected by the MP parameter. Although the PHUI-UP algorithm uses the TWPUDC property to reduce the search space, it still adopts a level-wise approach and can generate an enormous amount of candidates for mining PHUIs. Moreover, only a small percentage of these candidates are PHUIs. Therefore, when the MP is set higher, many redundant unpromising candidates are pruned early, and thus the search space and runtime of the PHUI-UP algorithm decreases sharply, while runtimes of the HUI-Miner and FHM algorithms remains unchanged.

5.3. Patterns analysis

In this subsection, PHUIs discovered by the PHUI-UP and PHUI-List algorithms in the uncertain datasets are evaluated. Notice that no algorithm had been previously developed for discovering PHUIs in uncertain datasets. In an uncertain dataset, when the probability of each transaction is set to 1.0, it means that each transaction has a 100% existential probability, and the uncertain dataset becomes a precise quantitative dataset. In that case, PHUI mining algorithms produce the same output as traditional high-utility itemset mining algorithms, that is the whole set of HUIs. In this experiment, the state-of-the-art HUI-Miner and FHM algorithms are thus used to mine HUIs by ignoring probability values in uncertain datasets. This result is compared to the designed PHUI-UP and PHUI-List algorithms for mining PHUIs. The results of pattern analysis for HUIs and PHUIs under various MUs and a fixed MP and under various MPs and a fixed MU are respectively shown in Figs. 6 and 7.

From Fig. 6, it can be observed that the number of PHUIs is always smaller than the number of HUIs for various MUs on both sparse and dense datasets. This indicates that numerous HUIs are discovered but few PHUIs are produced when considering the probability value of each transaction in an uncertain dataset. Therefore, in real-world applications, numerous HUIs may not be the patterns required by managers or retailers for taking efficient decisions. This situation especially occurs when the MU is set low. When the MU is set high, few PHUIs are produced by the designed algorithms compared to the number of HUIs discovered by HUI-Miner and FHM. This is because the proposed algorithms discover PHUIs by considering both the utility and the probability constraints, while HUI-Miner and FHM discover HUIs by only checking the utility constraint. Because PHUIs have distinct probability values, PHUIs are fewer and more valuable compared to the general HUI patterns. From the experimental results, it can also be seen that both the number of HUIs and the number of PHUIs decrease as the MU increases.

From Fig. 7, it can be observed that few PHUIs are discovered by the proposed algorithms compared to the number of HUIs discovered by HUI-Miner and FHM under various MPs on the four uncertain datasets. The number of discovered HUIs remains steady as the MP increases. This is reasonable since HUI-Miner and FHM only consider the high-utility constraint for discovering HUIs, and thus the number of HUIs is not affected when the MP is increased. The number of PHUIs decreases dramatically as the MP increases. The reason is the same as the one given for Fig. 6, i.e. that the proposed algorithms utilize two constraints for mining PHUIs. In particular,

Please cite this article as: J.C.-W. Lin et al., Efficient algorithms for mining high-utility itemsets in uncertain databases, Knowledge-Based Systems (2016), http://dx.doi.org/10.1016/j.knosys.2015.12.019

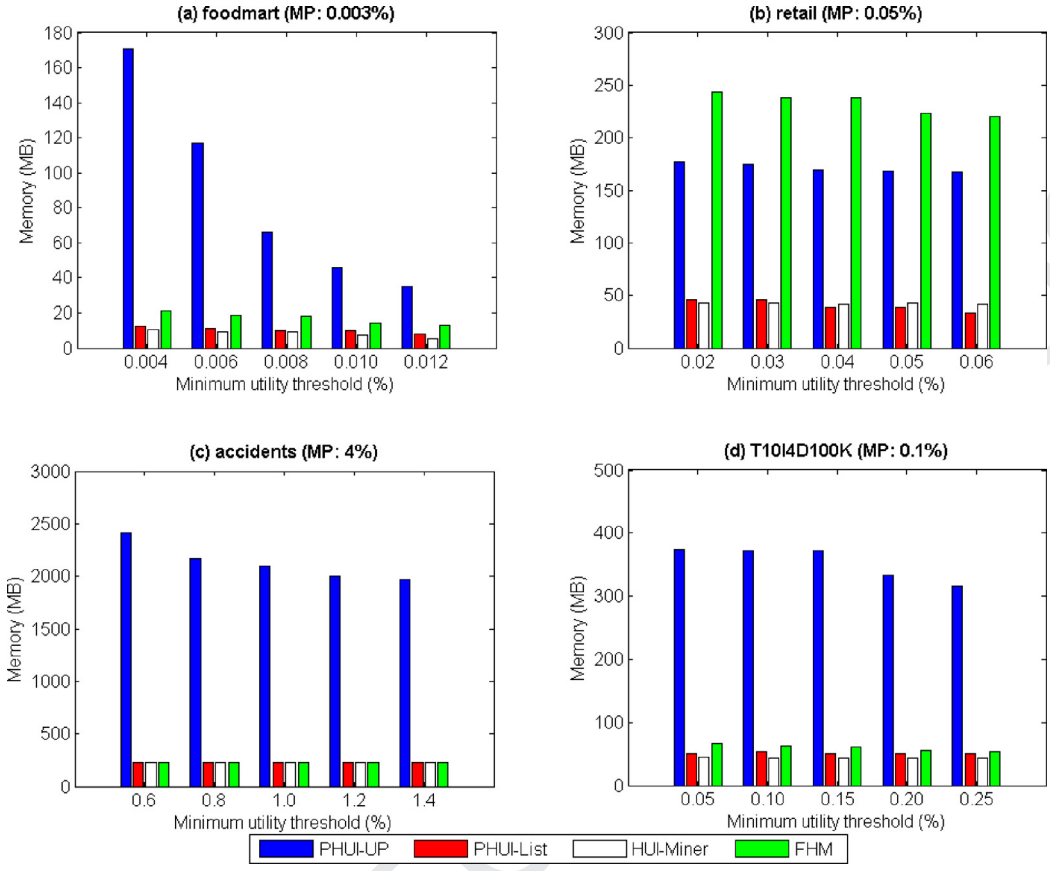

Fig. 8. Memory consumption under various MUs and a fixed MP.

the discovered PHUIs can be considered as more valuable than the HUIs found by traditional approaches, since probability values often occurs in real-life situations.

5.4. Memory Consumption

The memory consumption of the two designed algorithms and the HUI-Miner and FHM algorithms were also compared. Memory measurements were done using the Java API. The results under various MUs and a fixed MP, and under various MPs and a fixed MU, are respectively shown in Figs. 8 and 9. In these figures, the peak memory consumption of the compared algorithms is shown for all datasets.

From Figs. 8 and 9, it can be clearly seen that the proposed PHUI-List algorithm requires less memory compared to the PHUI-UP algorithm and the state-of-the-art HUI-Miner and FHM algorithms, both under various MUs and a fixed MP and under various MPs and a fixed MU, for the four datasets. Specifically, the PHUI-List algorithm requires nearly constant memory under various parameter values for the four datasets. This performance is somehow similar to the HUI-Miner and FHM algorithms except for the retail dataset. This result is reasonable since both HUI-Miner and FHM are utility-list-based algorithms, and they can easily prune unpromising itemsets with the help of actual utilities and remaining utilities. The FHM algorithm consumes slightly more memory than HUI-Miner because it stores information about co-occurrence of 2-itemsets in an additional data structure (this can be observed in Figs. 8 and 9 for the four datasets). Thanks to the advantages provided by the designed vertical PU-list data structure, the proposed PHUI-List algorithm discovers PHUIs by considering the utility and probability of itemsets, and thus more efficient pruning strategies are proposed in PHUI-List to improve its performance. Hence, the memory consumption of the PHUI-List algorithm is somehow similar to those of the HUI-Miner and FHM algorithms.

The PHUI-UP algorithm requires much more memory when the MU or MP are set lower, unlike the PHUI-List algorithm. For example, PHUI-UP requires 2,100 MB of memory on average, but PHUI-List only requires about 225 MB of memory on average, as it can be observed in Fig. 8(c). The reason is that the proposed PHUI-List algorithm utilizes two strategies to prune unpromising itemsets early, without building their PU-lists. It can also be observed that the performance gap between PHUI-UP and PHUI-List is smaller when MU is increased and MP is fixed or when MP is increased and MU is fixed. For instance, when MP is set to 0.001 and MU is set to 0.008%, the PHUI-UP and the PHUI-List algorithms respectively require 76.5 and 9 MB of memory on the foodmart dataset, as shown in Fig. 9(a). When MP is set to 0.009% and MU is set to 0.008%, the PHUI-UP and PHUI-List algorithms respectively require 20 and 5.5 MB of memory, as shown in Fig. 9(a). Since unpromising candidates can be pruned early, less memory is required to maintain candidates for the later mining process. Generally, the PHUI-List algorithm has better performance compared to the PHUI-UP algorithm under various parameter values for the four datasets. When the MU or MP are set lower, the “combinational explosion” problem always occurs during an iteration of the generate-and-test mechanism, producing numerous redundant patterns using a level-wise approach, compared to the PHUI-List algorithm.

5.5. Scalability

The scalability of all algorithms has also been compared on the synthetic dataset T10I4N4KD|X|K, where the dataset size has been varied from 100 to 500 K transactions, using increments of 100 K. Results under different MPs and MUs are shown in Fig. 10.

Please cite this article as: J.C.-W. Lin et al., Efficient algorithms for mining high-utility itemsets in uncertain databases, Knowledge-Based Systems (2016), http://dx.doi.org/10.1016/j.knosys.2015.12.019

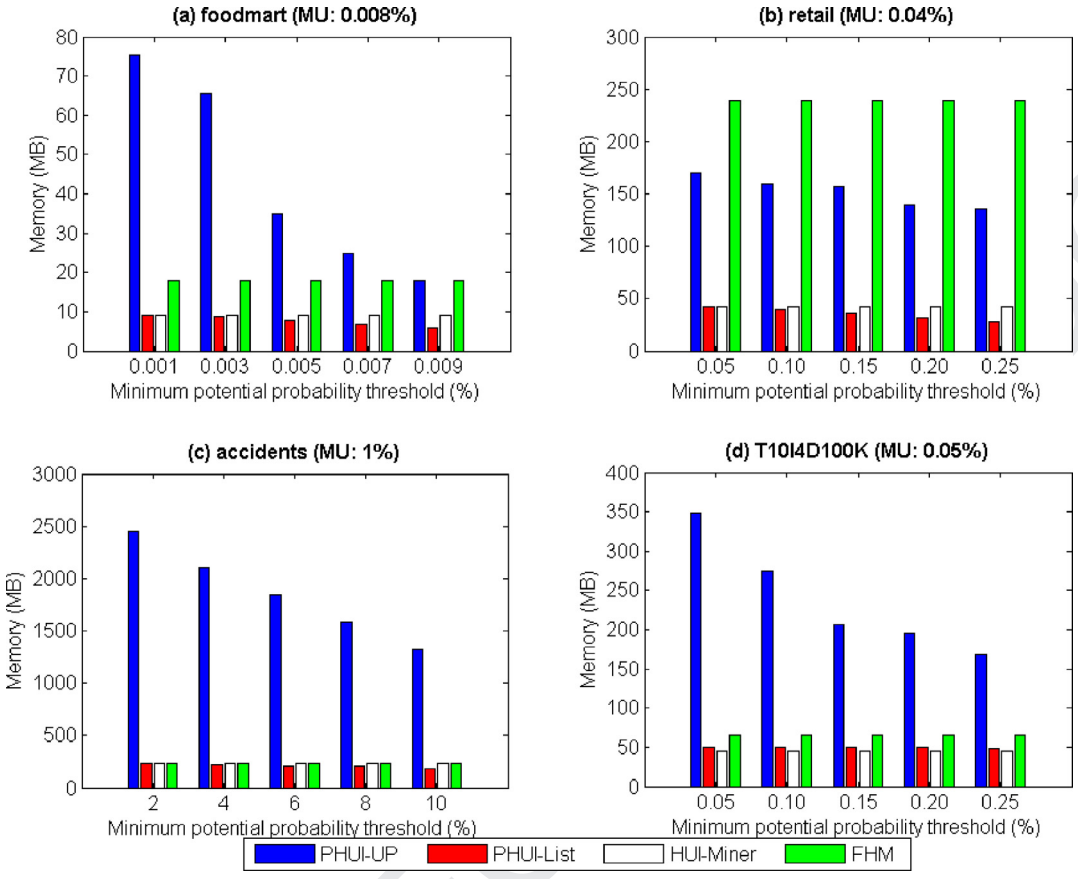

Fig. 9. Memory consumption under various MPs and a fixed MU.

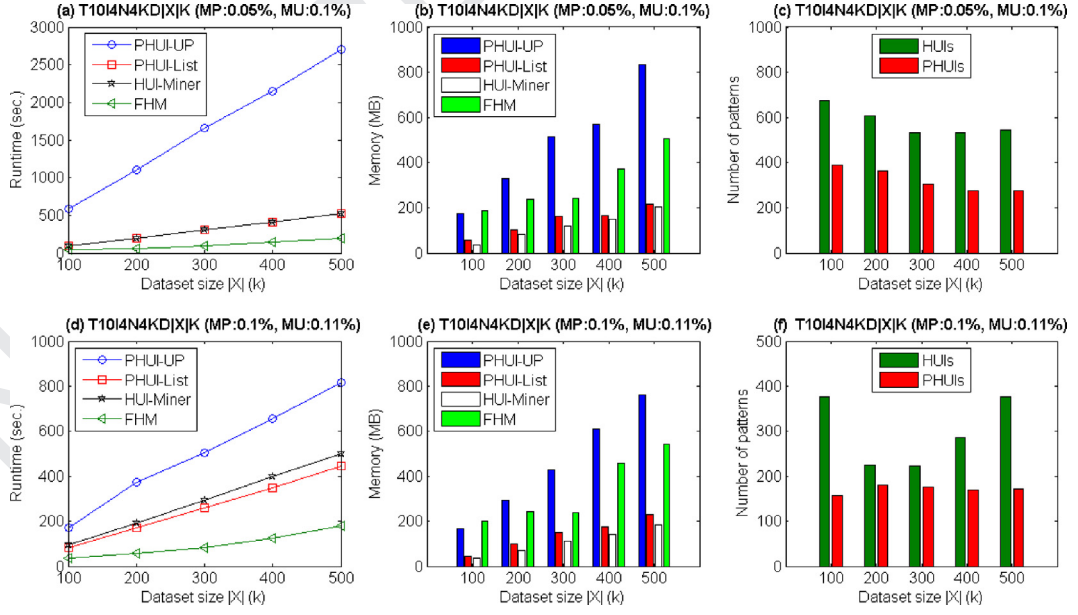

Fig. 10. Scalability results of the compared algorithms.

From Fig. 10, it can be observed that not only the state-of-the-art traditional HUI-Miner and FHM algorithms, but also the two proposed algorithms have good scalability when the dataset size is varied, both in terms of runtime, memory consumption, and number of patterns. From the results of Fig. 10(a)–(d), it can be observed that the two designed algorithms require more computations and memory to find the PHUIs as dataset size increases. The proposed PHUI-List algorithm always has better results than the PHUI-UP algorithm in terms of runtime and memory consumption. An interesting observation is that the runtime and memory consumption of the proposed PHUI-List algorithm increases steadily but those of PHUI-UP increase sharply as the dataset size increases. For example, PHUI-List was almost two orders of magnitude faster than the PHUI-UP algorithm when MP was set to 0.05% and MP

was set to 0.1%, and the dataset size was increased from 100 to 500 K, as shown in Fig. 10(a). Moreover, fewer PHUIs are produced by the proposed algorithms compared to the number of HUIs found by the HUI-Miner and FHM algorithms, as can be observed in Fig. 10(c) and (f). From the observed results of the scalability experiments, it can be concluded that the two proposed algorithms have good scalability and the proposed PU-list-based PHUI-List algorithm has better performance in terms of runtime, memory consumption 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, to our knowledge no research had previously considered mining high-utility itemsets in uncertain databases. In this paper, a novel framework is proposed 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 presented to mine itemsets that not only have high utilities but also have high existential probabilities. The designed PHUI-UP algorithm relies on a level-wise and generate-candidate-and-test approach for mining PHUIs. Although it utilizes a designed downward closure property to reduce the search space, the combinational explosion problem still occurs in the baseline PHUI-UP algorithm. The second PHUI-List algorithm is then developed to mine PHUIs more efficiently than using the PHUI-UP algorithm, based on the designed PU-list structure and the set-enumeration tree for mining PHUIs directly (without candidate generation). Substantial experiments were conducted on both real-life and synthetic databases to assess the performance of the two proposed algorithms in terms of runtime, number of discovered patterns, memory consumption, and scalability.

Since this is the first work addressing the problem of mining PHUIs in an uncertain database, there are numerous opportunities for extending this research such as dynamic data mining, stream mining, and top-*k* pattern mining. Besides, designing more efficient and condensed structures based on a different uncertain model for mining the desired information is also another critical issue.

Acknowledgment

This research was partially supported by the Tencent Project under grant CCF-TencentRAGR20140114, by the Shenzhen Peacock Project, China, under grant KQC201109020055A, by the National Natural Science Foundation of China (NSFC) under grant no.61503092, 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, Managing and mining uncertain data, Manag. Min. Uncertain Data (2010).
[3] C.C. Aggarwal, Y. Li, J. Wang, J. Wang, Frequent pattern mining with uncertain data, in: Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2009, pp. 29–38.
[4] C.C. Aggarwal, P.S. Yu, A survey of uncertain data algorithms and applications, IEEE Trans. Knowl. Data Eng. 21 (5) (2009) 609–623.
[5] R. Agrawal, T. Imielinski, A. Swami, Database mining: a performance perspective, IEEE Trans. Knowl. Data Eng. 5 (6) (1993) 914–925.
[6] R. Agrawal, T. Imielinski, A. Swami, Mining association rules between sets of items in large database, in: Proceedings of the ACM SIGMOD International Conference on Management of Data, 1993, pp. 207–216.
[7] R. Agrawal, R. Srikant, Fast algorithms for mining association rules in large databases, in: Proceedings of the International Conference on Very Large Data Bases, 1994, pp. 487–499.
[8] R. Agrawal, R. Srikant. Quest synthetic data generator. Available: http://www.Almaden.ibm.com/cs/quest/syndata.html (1994).
[9] C.F. Ahmed, S.K. Tanbeer, B.S. Jeong, Y.K. Le, Efficient tree structures for high utility pattern mining in incremental databases, IEEE Trans. Knowl. Data Eng. 21 (12) (2009) 1708–1721.
[10] T. Bernecker, H.P. Kriegel, M. Renz, F. Verhein, A. Zuefl, Probabilistic frequent itemset mining in uncertain databases, in: Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2009, pp. 119–128.
[11] R. Chan, Q. Yang, Y.D. Shen, Mining high utility itemsets, in: Proceedings of the IEEE International Conference on Data Mining, 2003, pp. 19–26.
[12] M.S. Chen, J. Han, P.S. Yu, Data mining: an overview from a database perspective, IEEE Trans. Knowl. Data Eng. 8 (6) (1996) 866–883.
[13] C.K. Chui, B. Kao, E. Hung, Mining frequent itemsets from uncertain data, in: Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining, 2007, pp. 47–58.
[14] P. Fournier-Viger, C.W. Wu, S. Zida, V.S. Tseng, FHM: faster high-utility itemset mining using estimated utility co-occurrence pruning, in: Proceedings of the International Symposium on Methodologies for Intelligent Systems, 8502, 2014, pp. 83–92.
[15] G.C. Lan, T.P. Hong, V.S. Tseng, Discovery of high utility itemsets from on-shelf time periods of products, Expert Syst. Appl. 38 (5) (2011) 5851–5857.
[16] L. Geng, H.J. Hamilton, Interestingness measures for data mining: a survey, ACM Comput. Surv. 38 (3) (2006) 1–32 Article 9.
[17] J. Han, J. Pei, Y. Yin, R. Mao, Mining frequent patterns without candidate generation: a frequent-pattern tree approach, Data Mining Knowl. Discov. 8 (1) (2004) 53–87.
[18] J.C.W. Lin, W. Gan, P. Fournier-Viger, T.P. Hong, Mining high-utility itemsets with multiple minimum utility thresholds, in: Proceedings of the ACM International Canadian Conference on Computer Science & Software Engineering, 2015, pp. 9–17.
[19] J.C.W. Lin, W. Gan, T.P. Hong, B. Zhang, An incremental high-utility mining algorithm with transaction insertion, Sci. World J. (2015).
[20] J.C.W. Lin, W. Gan, T.P. Hong, V.S. Tseng, Efficient algorithms for mining up-to-date high-utility patterns, Adv. Eng. Inf. 29 (3) (2015) 648–661.
[21] C.K.S. Leung, B. Hao, Mining of frequent itemsets from streams of uncertain data, IEEE Int. Conf. Data Eng. (2009) 1663–1670.
[22] C.K.S. Leung, M.A.F. Mateo, D.A. Brajczuk, A tree-based approach for frequent pattern mining from uncertain data, in: Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining, 2008, pp. 653–661.
[23] C.W. Lin, T.P. Hong, A new mining approach for uncertain databases using cufp trees, Expert Syst. Appl. 39 (4) (2012) 4084–4093.
[24] C.W. Lin, T.P. Hong, W.H. Lu, An effective tree structure for mining high utility itemsets, Expert Syst. Appl. 38 (6) (2011) 7419–7424.
[25] C. Liu, L. Chen, C. Zhang, Summarizing probabilistic frequent patterns: a fast approach, in: Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2013, pp. 527–535.
[26] M. Liu, J. Qu, Mining high utility itemsets without candidate generation, in: Proceedings of the ACM International Conference on Information and Knowledge Management, 2012, pp. 55–64.
[27] Y. Liu, W.K. Liao, A. Choudhary, A two-phase algorithm for fast discovery of high utility itemsets, in: Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining, 2005, pp. 689–695.
[28] Microsoft. Example database foodmart of microsoft analysis services. Available: http://msdn.microsoft.com/en-us/library/aa217032(SQL.80).aspx.
[29] D. Nilesh, S. Dan, Efficient query evaluation on probabilistic databases, VLDB J. 16 (4) (2007) 523–544.
[30] P. Fournier-Viger, A. Gomariz, T. Gueniche, A. Soltani, C.W. Wu, V.S. Tseng, SPMF: a java open-source pattern mining library, J. Mach. Learn. Res. 15 (1) (2014) 3389–3393.
[31] L. Sun, R. Cheng, D.W. Cheung, J. Cheng, Mining uncertain data with probabilistic guarantees, in: Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2010, pp. 273–282.
[32] Y. Tong, L. Chen, Y. Cheng, P.S. Yu, Mining frequent itemsets over uncertain databases, VLDB Endow. 5 (11) (2012) 1650–1661.
[33] V.S. Tseng, B.E. Shie, C.W. Wu, P.S. Yu, Efficient algorithms for mining high utility itemsets from transactional databases, IEEE Trans. Knowl. Data Eng. 25 (8) (2013) 1772–1786.
[34] V.S. Tseng, C.W. Wu, B.E. Shie, P.S. Yu, UP-growth: An efficient algorithm for high utility itemset mining, in: Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2010, pp. 253–262.
[35] L. Wang, R. Cheng, S.D. Lee, D. Cheung, Accelerating probabilistic frequent itemset mining: A model-based approach, in: Proceedings of the ACM International Conference on Information and Knowledge Managemen, 2010, pp. 429–438.
[36] L. Wang, D.L. Cheung, R. Cheng, S.D. Lee, X.S. Yang, Efficient mining of frequent item sets on large uncertain databases, IEEE Trans. Knowl. Data Eng. 24 (12) (2012) 2170–2183.

[37] C.W. Wu, B.E. Shie, V.S. Tseng, P.S. Yu, Mining top-k high utility itemsets, in: Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2012, pp. 78–86.
[38] H. Yao, H.J. Hamilton, Mining itemset utilities from transaction databases, Data Knowl. Eng. 59 (3) (2006) 603–626.
[39] H. Yao, H.J. Hamilton, C.J. Butz, A foundational approach to mining itemset utilities from databases, in: Proceedings of the SIAM International Conference on Data Mining, 2004, pp. 211–225.
[40] M. Zihayat, A. An, Mining top-k high utility patterns over data streams, Inf. Sci. 285 (2014) 138–161.

Please cite this article as: J.C.-W. Lin et al., Efficient algorithms for mining high-utility itemsets in uncertain databases, Knowledge-Based Systems (2016), http://dx.doi.org/10.1016/j.knosys.2015.12.019