Efficient Mining of Uncertain Data for High-Utility Itemsets

Jerry Chun-Wei Lin[1(✉)], Wensheng Gan[1], Philippe Fournier-Viger[2], Tzung-Pei Hong[3,4], and Vincent S. Tseng[5]

[1] School of Computer Science and Technology, Harbin Institute of Technology Shenzhen Graduate School, Shenzhen, China
jerrylin@ieee.org, wsgan001@gmail.com
[2] School of Natural Sciences and Humanities, Harbin Institute of Technology Shenzhen Graduate School, Shenzhen, China
phifv@hitsz.edu.cn
[3] Department of Computer Science and Information Engineering, National University of Kaohsiung, Kaohsiung, Taiwan
tphong@nuk.edu.tw
[4] Department of Computer Science and Engineering, National Sun Yat-sen University, Kaohsiung, Taiwan
[5] Department of Computer Science, National Chiao Tung University, Hsinchu, Taiwan
vtseng@cs.nctu.edu.tw

Abstract. High-utility itemset mining (HUIM) is emerging as an important research topic in data mining. Most algorithms for HUIM can only handle precise data, however, uncertainty that are embedded in big data which collected from experimental measurements or noisy sensors in real-life applications. In this paper, an efficient algorithm, namely Mining Uncertain data for High-Utility Itemsets (MUHUI), is proposed to efficiently discover potential high-utility itemsets (PHUIs) from uncertain data. Based on the probability-utility-list (PU-list) structure, the MUHUI algorithm directly mine PHUIs without candidate generation and can reduce the construction of PU-lists for numerous unpromising itemsets by using several efficient pruning strategies, thus greatly improving the mining performance. Extensive experiments both on real-life and synthetic datasets proved that the proposed algorithm significantly outperforms the state-of-the-art PHUI-List algorithm in terms of efficiency and scalability, especially, the MUHUI algorithm scales well on large-scale uncertain datasets for mining PHUIs.

Keywords: Data mining · Uncertainty · High-utility itemset · PU-list · Pruning strategies

1 Introduction

Knowledge Discovery in Database (KDD) aims at finding the meaningful and useful information from the amounts of mass data [4,10,11]. Depending on

B. Cui et al. (Eds.): WAIM 2016, Part I, LNCS 9658, pp. 17–30, 2016.
DOI: 10.1007/978-3-319-39937-9_2

different requirements in various domains and applications, frequent itemset mining (FIM) [11] and association rule mining (ARM) [4] are the important and fundamental issues in KDD. Instead of traditional FIM and ARM, high-utility itemset mining (HUIM) [8,21] incorporates both quantity and profit values of an item/set to measure how "useful" an item or itemset is. An itemset is defined as a high-utility itemset (HUI) if its utility value in a database is no less than a user-specified minimum utility count. In general, the "utility" of an item/set can be presented as the user-specified factor in real-life applications, i.e., weight, cost, risk, or unit profit. Chan et al. [8] first proposed the concept of HUIM. Yao et al. [21] then defined a unified framework for mining high-utility itemsets (HUIs). The goal of HUIM is to identify the rare items or itemsets in the transactions, but it can bring valuable profits for the retailers or managers. HUIM serves as a critical role in data analysis and has been widely utilized to discover knowledge and mine valuable information. Many approaches developed in HUIM have been extensively studied, such as Two-Phase [15], IHUP [6], UP-growth [18], UP-growth+ [19], HUI-Miner [14], FHM [17], among others. In real-world situations, the mass data may be uncertainly collected from the incomplete data sources, such as wireless sensor network, RFID, GPS, or WiFi systems [2,3], and the size of the collected uncertain data is very large. Traditional data mining technologies for handling precise data cannot be directly, however, applied to the incomplete or inaccurate data for discovering the required information on business service.

In real-life applications, utility and probability are two different measures for an object (e.g., an useful pattern). The utility is a semantic measure (how "utility" of a pattern is based on the user's priori knowledge and goals), while probability is an objective measure (the probability of a pattern is an objective existence) [10]. Up to now, most algorithms of HUIM have been extensively developed to handle precise data, which are not suitable to mine the data with uncertainty. It may be useless or misleading if the discovered results of HUIs with low existential probability. To the best of our knowledge, the PHUIM framework [13] is the first work to address the issue of mining HUIs from uncertain data. However, the task of mining HUIs from uncertain data, especially large-scale uncertain data, remains very costly in terms of execution time. Therefore, it is a non-trivial task and an important challenge to design more efficient algorithms to solve the limitation. In this paper, an efficient mining model namely Mining large-scale Uncertain data for High-Utility Itemsets (MUHUI) is proposed to effectively discover the Potential High-Utility Itemsets (PHUIs). Major contributions of this paper are summarized as follows:

- Fewer studies on HUIM have addressed the mining of HUIs from uncertain data by taking into account both the semantics-based utility measure and objective probability measure. In this paper, an efficient MUHUI algorithm is designed to successfully and directly mine the HUIs from uncertain databases without candidate generation.
- Based on the utility and probability properties, several pruning strategies are developed to efficiently and early prune the search space and the unpromising

itemsets. Thus, the search space for mining HUIs can be greatly reduced, and the mining performance can be significantly improved.
– Extensive experiments show that the proposed algorithm is more efficient than the state-of-the-art PHUI-List algorithm for mining uncertain HUIs in terms of runtime, effectiveness of prune strategies and scalability, especially MUHUI scales well than PHUI-List in large-scale uncertain databases.

2 Related Works

HUIM is different from FIM and ARM, which incorporates the measurements of local transaction utility (occur quantity) and external utility (unit profit) to discover the profitable itemsets from the quantitative databases. The HUIM was first proposed by Chan et al. [8]. Yao et al. [21] then defined a strict unified framework for HUIM. Since the downward closure property of ARM does no longer hold for HUIM, Liu et al. [15] designed the TWU model to maintain the transaction-weighted downward closure (TWDC) property, which can be used to greatly reduce the number of unpromising candidates for mining HUIs in a level-wise mechanism. Several tree-based approaches for mining HUIs such as IHUP [6], UP-growth [18], and UP-growth+ [19] have been extensively studied. Based on these pattern-growth approaches, more computations are still required to generate and keep the huge number of discovered candidates for mining the actual HUIs.

To solve the above limitations of traditional HUIM, the HUI-Miner algorithm [14] was proposed to directly mine HUIs to avoid the multiple database scans without candidate generation based on the designed utility-list structure. The FHM algorithm [17] was further proposed to enhance the performance of HUI-Miner by analyzing the co-occurrences among 2-itemsets. Instead of traditional HUIM, the variants of HUIM have been also extended and developed [12,20]. The development of other algorithms for HUIM is still in progress, but most of them are processed to handle precise data, the PHUIM framework [13] is the only work which focuses on mining high-utility itemsets on uncertain data.

3 Preliminaries and Problem Statement

3.1 Preliminaries

Based on the mentioned reason in [13], the tuple uncertainty model [3,7] and the expected support-based model [9] are also adopted in the proposed algorithm. Let $I = \{i_1, i_2, \ldots, i_m\}$ be a finite set of m distinct items in an uncertain quantitative database $D = \{T_1, T_2, \ldots, T_n\}$, where each transaction $T_q \in D$ is a subset of I, contains several items with their purchase quantities $q(i_j, T_q)$, and has an unique identifier, *TID*. In addition, each transaction has a unique probability of existence $p(T_q)$, which indicates that T_q exists in D with probability $p(T_q)$ based on a tuple uncertainly model. A corresponding profit table,

Table 1. An example database

TID	Transaction	Probability
T_1	$(A,1);(C,3);(D,4)$	0.95
T_2	$(B,1);(C,1);(D,2)$	0.85
T_3	$(A,2);(B,2);(E,3)$	0.85
T_4	$(C,2);(E,2)$	0.5
T_5	$(B,1);(D,2);(E,2)$	0.75
T_6	$(A,1);(C,2);(D,1)$	0.7
T_7	$(A,3);(C,1);(D,3);(E,4)$	0.45
T_8	$(B,1);(C,4);(E,1)$	0.36
T_9	$(B,3);(D,5)$	0.81
T_{10}	$(D,5);(E,2)$	0.6

Table 2. Derived PHUIs

Itemset	Utility	Pro
(C)	130	5.26
(E)	70	4.61
(AC)	90	2.75
(CD)	80	3.55
(CE)	105	2.51
(DE)	50	2.40
(ACD)	98	2.75

$ptable = \{pr_1, pr_2, \ldots, pr_m\}$, in which pr_j is the profit value of an item i_j, is created. A k-itemset X, denoted as X^k, is a set of k distinct items $\{i_1, i_2, \ldots, i_k\}$. Given two user-specified thresholds, the minimum utility threshold (ε) and the minimum potential probability threshold (μ).

Table 1 shows an example of a tuple uncertainty quantitative database. The profit table is defined as $\{pr(A)$: 6; $pr(B)$: 3; $pr(C)$: 10; $pr(D)$: 1; $pr(E)$: 5$\}$, and the two thresholds are respectively set at ε (= 15 %) and μ (= 18 %).

Definition 1. The utility of an item i_j in a transaction T_q is denoted as $u(i_j, T_q)$ and defined as: $u(i_j, T_q) = q(i_j, T_q) \times pr(i_j)$.

For example, $u(A,\ T_1) = q(A,\ T_1) \times pr(A) = (1 \times 6) = 6$.

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

For example, $p(A,\ T_1) = p(T_1) = 0.95$, and $p(AD,\ T_1) = p(T_1) = 0.95$.

Definition 3. The utility of an itemset X in transaction T_q is denoted as $u(X,\ T_q)$, which can be defined as: $u(X, T_q) = \sum_{i_j \in X \wedge X \subseteq T_q} u(i_j, T_q)$.

For example, the utility of (AD) in T_1 is calculated as $u(AD,\ T_1) = u(A,\ T_1) + u(D,\ T_1) = q(A,\ T_1) \times pr(A) + q(D,\ T_1) \times pr(D) = (1 \times 6) + (4 \times 1) = 10$.

Definition 4. The utility of an itemset X in D is denoted as $u(X)$, which can be defined as: $u(X) = \sum_{X \subseteq T_q \wedge T_q \in D} u(X, T_q)$.

Definition 5. An itemset X is defined as a HUI if its utility value is no less than the minimum utility count as: $u(X) \geq \varepsilon \times TU$, in which TU is the total utility of the be processed database.

Definition 6. The potential probability of an itemset X in D is denoted as $Pro(X)$, which can be defined as: $Pro(X) = \sum_{X \subseteq T_q \wedge T_q \in D} p(X, T_q)$.

Definition 7. An itemset X in an uncertain database D is defined as a potential high-utility itemset (PHUI) if it satisfies the following two conditions: (1) X is a HUI w.r.t. $u(X) \geq \varepsilon \times TU$; (2) $Pro(X) \geq \mu \times |D|$. A desired PHUI indicates the itemset has both high potential probability and high utility value.

Problem Statement: Given an uncertain database D with total utility is TU, the minimum utility threshold and the minimum potential probability threshold are respectively set as ε and μ. The problem of potential high-utility itemset mining (PHUIM) from uncertain data is to mine PHUIs whose utilities are larger than or equal to $(\varepsilon \times TU)$, and its potential probability is larger than or equal to $(\mu \times |D|)$.

4 Proposed MUHUI Algorithm for Mining HUIs

Alougth the PHUI-List algorithm has better performance compared to the upper-bound-based PHUI-UP algorithm [13], however, it explores the search space of itemsets by generating itemsets, and a costly join operation of probability-utility-list (PU-list) has to be performed recursively to evaluate the probability and utility information of each itemset. By utilizing the PU-list structure, a more efficient MUHUI algorithm is proposed here to improve the performance for efficiently mining PHUIs.

4.1 The PU-list Structure

The PU-list structure [13] is a new vertical data structure, it incorporates the probability and utility properties to keep necessary information from uncertain data in terms of TID information, probability, utility, and remaining utility information. Let 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 $X = \{CD\}$ and transaction T_7 in Table 1, $T_7 \backslash X = \{AE\}$, and $T_7/X = \{E\}$.

Definition 8 (Probability-Utility-list, PU-list). The PU-list of an itemset X in a database is denoted as $X.PUL$. It contains an entry (element) for each transaction T_q where X appears $(X \in T_q \subseteq D)$. An element 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***), in which ***ru*** is defined as $X.ru(T_q) = \sum_{i_j \in (T_q/X)} u(i_j, T_q)$.

Therefore, all necessary information from uncertain data can be compressed into the designed PU-list structure without losing any useful information. Thanks to the property of PU-list, the probability and utility information of the longer

k-itemset can be built by joining its parent node and uncle node, i.e., (k-1)-itemset. The join operation can be easily done without rescanning the database. The construction procedure of the PU-list is recursively processed if it is necessary to determine the k-itemsets in the search space, details of the construction can be referred to [13]. Note that it is necessary to initially construct the PU-list of the complete set of HTWPUI[1] [13] as the input for the later recursive process. The PU-list is constructed in TWU ascending order as ($B \prec A \prec D \prec E \prec C$), which is shown in Fig. 1.

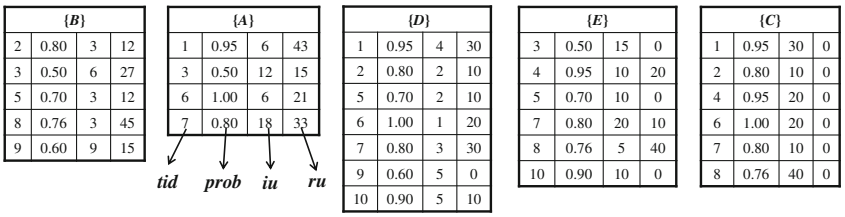


Fig. 1. Constructed PU-list structure of HTWPUI[1].

Definition 9. The sum of the utilities and remaining utilities of an itemset X in D, denoted as $X.IU$ and $X.RU$, respectively, which can be defined as:

$$X.IU = \sum_{X \subseteq T_q \wedge T_q \in D} (X.iu), X.RU = \sum_{X \subseteq T_q \wedge T_q \in D} (X.ru). \quad (1)$$

4.2 Search Space and Properties

Based on the PU-list structure, the search space of the proposed MUHUI algorithm can be represented as the Set-enumeration tree by the TWU values of the 1-items in the set of HTWPUI[1] in ascending order, as shown in Fig. 2 (left). Based on the constructed Set-enumeration tree, the following lemmas can be obtained.

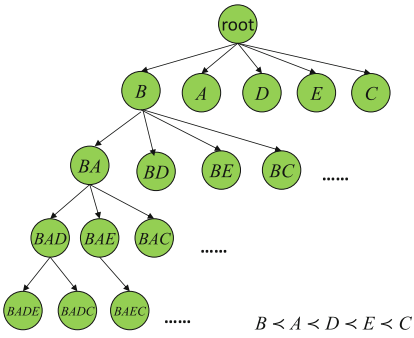
Lemma 1. *The sum of all the probabilities of any node in the Set-enumeration tree is greater than or equal to the sum of all the probabilities of any of its child nodes.*

Proof. Assume a (k-1)-itemset w.r.t. a node in the Set-enumeration tree be $X^{k-1}(k \leq 2)$, and any of its child nodes be denoted as X^k. Since $p(X^k, T_q) = p(T_q)$ for any transaction T_q in D, it can be found that: $\frac{p(X^k, T_q)}{p(X^{k-1}, T_q)} = \frac{p(T_q)}{p(T_q)} = 1$. Since X^{k-1} is subset of X^k, the $TIDs$ of X^k is the subset of the $TIDs$ of X^{k-1}, thus,

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

Lemma 2. *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 child nodes.*

Proof. From [13], this lemma holds.



Item	*A*	*B*	*C*	*D*	*E*
B	33	-	-	-	-
C	118	63	-	-	-
D	118	44	133	-	-
E	84	96	129	81	-

The built *EUCS*

Fig. 2. Constructed Set-enumeration tree and *EUCS*.

4.3 Proposed Pruning Strategies

Based on the PU-list and the properties of probability and utility, five efficient pruning strategies are designed in MUHUI to early prune unpromising itemsets.

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

Proof. Let X^{k-1} be a (k-1)-itemset and its superset k-itemset is denoted as X^k, then

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

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

Theorem 2 (PHUIs $\subseteq$ HTWPUIs). *The transaction-weighted probability and utilization downward closure (TWPUDC) property ensures that PHUIs $\subseteq$ HTWPUIs, which indicates that if an itemset is not a HTWPUI, then none of its supersets will be PHUIs [13].*

By utilizing the TWPUDC property, we only need to construct the PU-list for those promising itemsets w.r.t. the HTWPUIs. Furthermore, the following several pruning strategies are proposed in MUHUI to speed up the computations.

Strategy 1. *After the first database scan, we can obtain the TWU and probability value of each 1-item in database. If the TWU of a 1-item i (TWU(i)) and the sum of all the probabilities of i (Pro(i)) do not satisfy the two conditions of HTWPUI, this item can be directly pruned, and none of its supersets is a desired PHUI.*

Strategy 2. *When traversing the Set-enumeration tree based on a depth-first search strategy, if the sum of all the probabilities of a tree node X w.r.t. Pro(X) in its constructed PU-list is less than the minimum potential probability, then none of the child nodes of this node is a desired PHUI.*

Strategy 3. *When traversing the Set-enumeration tree based on a depth-first search strategy, if the sum of X.IU and X.RU of any node X is less than the minimum utility count, any of its child node is not a PHUI, they can be regarded as irrelevant and be pruned directly.*

Theorem 3 (Estimated Utility Co-occurrence Pruning strategy, EUCP). *If the TWU of 2-itemset is less than the minimum utility count, any superset of this 2-itemset is not a HTWUI and would not be a HUI either [17].*

To effectively apply the EUCP strategy, a structure named Estimated Utility Co-occurrence Structure (*EUCS*) [17] is built in the proposed algorithm. It is a matrix that stores the TWU values of the 2-itemsets, as shown in Fig. 2 (right). Note that *EUCS* is built in the first database scan after discovering the $HTWPUI^1$.

Strategy 4. *Let X be an itemset (node) encountered during the depth-first search of the Set-enumeration tree. If the TWU of a 2-itemset $Y \subseteq X$ according to the constructed EUCS is less than the minimum utility threshold, X is not a HTWPUI and would not be a PHUI; none of its child nodes is a PHUI. The construction of the PU-lists of X and its children is unnecessary to be performed.*

Strategy 5. *Let X be an itemset (node) encountered during the depth-first search of the Set-enumeration tree. After constructing the PU-list of an itemset, if X.PUL is empty or the Pro(X) value is less than the minimum probability threshold, X is not a PHUI, and none of X its child nodes is a PHUI. The construction of the PU-lists for the child nodes of X is unnecessary to be performed.*

4.4 Proposed MUHUI Algorithm

As shown in the main procedure of MUHUI (Algorithm 1), it first scans the uncertain database to calculate the $TWU(i)$ and $Pro(i)$ values of each item $i \in I$ (Line 1), and then find the set of I^* w.r.t. $HTWPUI^1$ (Line 2, pruning Strategy 1). After sort I^* in TWU ascending order (Line 3), the MUHUI algorithm scans D again to construct the PU-list for each 1-item $i \in I^*$, and build the *EUCS* structure (Line 4). After that, recursively using a depth-first search procedure PHUI-Search (Line 5) to mine PHUIs. Details of the PHUI-Search procedure are shown in Algorithm 2.

Input: D, *ptable*, ε, μ.
Output: The set of potential high-utility itemsets (PHUIs).
1 scan D to calculate the $TWU(i)$ and $Pro(i)$ of each item $i \in I$;
2 find $I^* \leftarrow \{i \in I | TWU(i) \geq TU \times \varepsilon \wedge Pro(i) \geq |D| \times \mu\}$; **/* strategy 1*/** ;
3 sort I^* in TWU ascending order $\prec$;
4 scan D once again to construct the PU-list of each $i \in I^*$ and build the $EUCS$;
5 **call PHUI-Search**$(\phi, I^*, EUCS, \varepsilon, \mu)$;
6 **return** *PHUIs*;

Algorithm 1. MUHUI algorithm

The PHUI-Search procedure takes the inputs as: X, $extendOfX$, $EUCS$, ε and μ. Firstly, each itemset X_a is determined to directly produce the PHUIs (Lines 2 to 4). Two pruning strategies, both Strategy 2 and Strategy 3, are applied to further determine whether its extensions satisfy the PHUI conditions for executing the later depth-first search (Line 5). Before executes the PU-list construction procedure to build PU-lists for itemsets with prefix itemset, the MUHUI algorithm utilizes the EUCP strategy to check whether those itemsets are need to build PU-list or not (Line 8, pruning Strategy 4). If X_a is promising, the $Construct(X, X_a, X_b)$ is executed to construct a set of PU-list of all 1-extensions of itemset X_a (w.r.t. $extendOfXa$) (Lines 8 to 12). Note that each constructed X_{ab} is a 1-extension of itemset X_a (Line 10), if the built PU-list of X_{ab} satisfies the pruning Strategy 5, X_{ab} should be put into the set of $extendOfXa$ for executing the later depth-first search; otherwise, X_{ab} would be directly pruned (Lines 11 to 12). The designed PHUI-Search procedure is recursively processed to mine PHUIs (Line 13). Based on the above pruning strategies, the designed MUHUI algorithm can prune the itemsets with lower potential probability and utility count early, without constructing their PU-lists of extensions.

5 Experiments

We performed extensive experiments to evaluate the proposed MUHUI algorithm. Note that the PHUI-UP and PHUI-List algorithms are the first work focus on mining uncertain data for HUIs, and the state-of-the-art PHUI-List algorithm significantly outperforms the PHUI-UP algorithm [13]. Only the PHUI-UP and PHUI-List algorithms for mining PHUIs are compared against the proposed MUHUI algorithm, in terms of runtime, memory usage, the effect of different pruning strategies, and scalability. Note that the MUHUI2 algorithm adopts the all designed pruning strategies, the MUHUI1 algorithm does not use the pruning Strategy 5.

In order to perform a fair comparison, all algorithms used in the experiments are also implemented in Java language and performed on a personal computer with 4 GB of RAM and running the 32-bit Microsoft Windows 7 operating system. And experiments are also conducted on four datasets including both real-world datasets (foodmart [16], accident [1], and retail [1]), and synthetic dataset

Input: X: an itemset, $extendOfX$: a set of all extensions of X, $EUCS$, ε, μ.
Input: The set of potential high-utility itemsets (PHUIs)
for *each itemset* $X_a \in extendOfX$ **do**
  obtain the $X_a.IU$, $X_a.RU$ and $Pro(X_a)$ values from the built $X_a.PUL$;
  if $X_a.IU \geq TU \times \varepsilon \wedge Pro(X) \geq |D| \times \mu$ **then**
    $PHUIs \leftarrow PHUIs \cup X_a$;
  if $X_a.IU + X_a.RU \geq TU \times \varepsilon \wedge Pro(X_a) \geq |D| \times \mu$ **then**
    $extendOfX_a \leftarrow \emptyset$;
    for *each itemset* $X_b \in extendOfX_a$ *such that* X_b *after* X_a **do**
      if $\exists\, TWU(ab) \in EUCS \wedge TWU(ab) \geq TU \times \varepsilon$ **then**
        $X_{ab} \leftarrow X_a \cup X_b$;
        $X_{ab}.PUL \leftarrow construct(X, X_a, X_b)$;
        if $X_{ab}.PUL \neq \emptyset \wedge Pro(X_{ab}) \geq |D| \times \mu$ **then**
          $extendOfX_a \leftarrow extendOfX_a \cup X_{ab}$;
    call PHUI-Search$(X_a, extendOfX_a, EUCS, \varepsilon, \mu)$;
return PHUIs;

Algorithm 2. PHUI-Search Procedure

(T10I4D100K) generated using the IBM Quest Synthetic Data Generator [5]. Both the quantity (internal) and profit (external) values are assigned to the items in the test datasets by the simulation method in previous studies [13,15,19] except for foodmart dataset. In addition, due to the tuple uncertainty property, each transaction in these datasets is randomly assigned a unique probability value in the range of (0.0, 1.0].

5.1 Execution Time

Experiments are compared under varied minimum utility thresholds (abbreviated as MUs) with the fixed minimum potential probability threshold (abbreviated as MP). The runtime results under varied MUs with a fixed MP are shown in Fig. 3.

From Fig. 3, it can be observed that the runtime of all the algorithms is decreased along with the increasing of MU. In particular, the proposed MUHUI algorithm is generally up to almost one or two orders of magnitude faster than the PHUI-UP algorithm, and also outperforms the state-of-the-art PHUI-List algorithm on all datasets. It is reasonable since the upper-bound-based generate-and-test mechanism has worse results than the vertical PU-list-based approaches. Besides, the MUHUI algorithm uses five pruning strategies to early prune unpromising itemsets and search space, which can avoid the costly join operations of a huge number of PU-lists for mining PHUIs, but the PHUI-List. When the MU is set quite low, longer patterns of HTWPUIs are first discovered by the PHUI-UP algorithm, and thus more computations are needed to process with the generate-and-test mechanism, especially in a dense dataset. While the

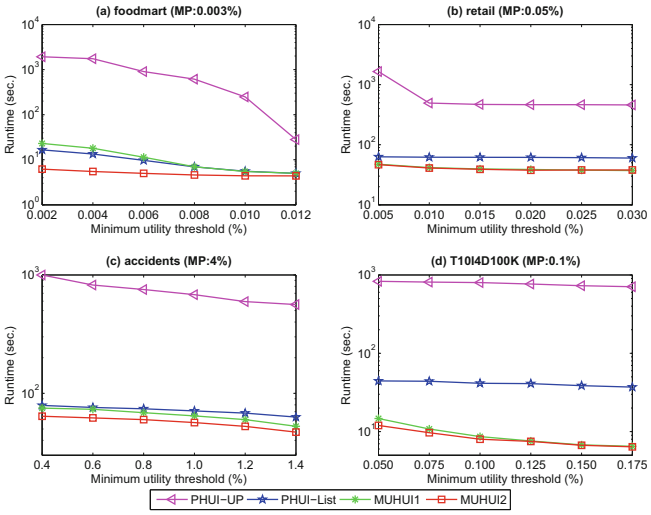


Fig. 3. Runtime under varied MUs with a fixed MP (Color figure online).

PHUI-List, MUHUI1 and MUHUI2 algorithms directly determine the PHUIs from the Set-enumeration tree without candidate generation in a level-wise way, it can effectively avoid the time-consuming dataset scan. Moreover, the MUHUI algorithm applies five pruning strategies to early prune the unpromising items, thus greatly reducing the computations than the PHUI-List algorithm.

5.2 Memory Usage

The memory usage of the compared algorithms were evaluated by using the Java API. In the same way, the performance was tested and examined under varied MUs with a fixed MP. From Fig. 4, it can be clearly seen that the proposed MUHUI algorithm requires less memory usage compared to PHUI-UP, but consumes little more memory than the state-of-the-art PHUI-List algorithm except for the retail dataset. Specially, the memory usage of the two PU-list-based algorithms, PHUI-List and MUHUI, change smoothly under varied parameters in four datasets. This performance is somehow similar to the conclusion given as the above analysis of runtime. This result is reasonable since both PHUI-List and MUHUI are PU-list-based algorithms, they can easily prune unpromising itemsets with the help of actual utilities and remaining utilities. The reason why a little more memory than PHUI-List always consumes for MUHUI is that it has to spend the extra memory cost for storing the additional *EUCS* data structure. Hence, the memory usage of the MUHUI algorithm is somehow similar to that of PHUI-List.

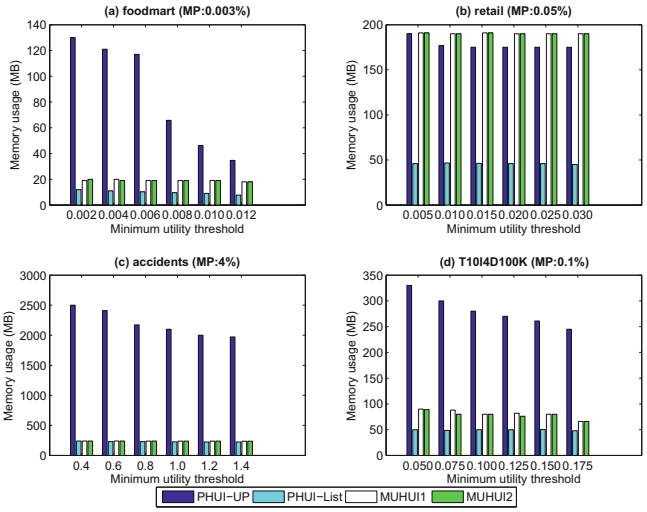


Fig. 4. Memory usage under varied MUs with a fixed MP. (Color figure online)

5.3 Scalability Analysis

As shown in Fig. 5, the scalability of the four algorithms is compared in the synthetic dataset T10I4N4KD|X|K with different scales, which is set MP = 0.05 %, MU = 0.1 %, and |X| is set varying from 100 to 500. It can be observed that the runtime of all compared algorithms is linear increased along with the increasing of dataset size |X|. The performance of MUHUI1 and MUHUI2 are, however, relatively stable to the variations of |X|. With the increasing of the size of dataset, the time of MUHUI1 is close to that of MUHUI2, but significantly faster than that of PHUI-List. Specially, the gap of runtime among them grows wider with the increasing of dataset size. With the increasing of dataset size, the running time of algorithms is linearly increasing as well, and the proposed MUHUI algorithm scales well on large-scale dataset. Figure 5(b) shows the memory usages of four algorithms which indicates the linearity in term of dataset size. In addition, we can find that the memory usage of the two PU-list-based algorithms is steady increasing than that of PHUI-UP.

To evaluated the effect of proposed different pruning strategies, the number of visited nodes in the search space are further compared, as can be observed in Fig. 5(c). Note that the number of visited nodes in the PHUI-List, MUHUI1 and MUHUI2 algorithms are denoted as N_1, N_2, and N_3, respectively. It shows that the number of the visited nodes of MUHUI1 and MUHUI2 are quite less than that of PHUI-List.

6 Conclusion

In this paper, an efficient algorithm called Mining of Uncertain data for High-Utility Itemsets (MUHUI) is proposed to consider the mining of those itemsets

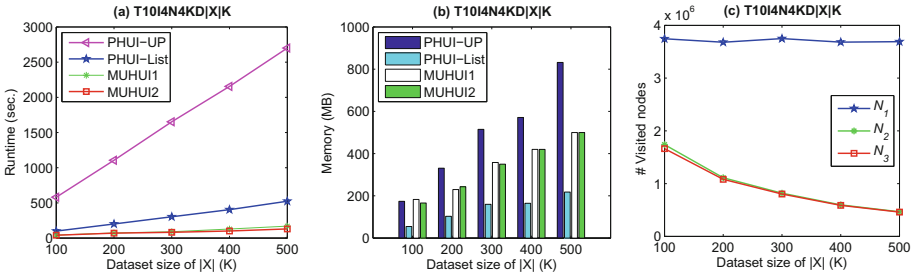


Fig. 5. Scalability of compared algorithms. (Color figure online)

with high-utility and high probability. The MUHUI algorithm is developed by proposing several efficient pruning strategies, and to improve the performance. Based on the PU-list structure, MUHUI utilizes the properties of probability and utility. Several efficient pruning strategies are also designed to speed up the computations, which can effectively avoid the construction of PU-lists of the huge number of unpromising itemsets. Substantial experiments both on real-life and synthetic datasets show that the proposed algorithm consumes a little more memory than the PHUI-List algorithm, but has great performance in terms of runtime, the number of visited nodes in the enumeration tree, and scalability compared to the past works. Specifically, the MUHUI algorithm is more scalable than the PHUI-UP and PHUI-List algorithms on large-scale uncertain databases.

Acknowledgment. This research was partially supported by the National Natural Science Foundation of China (NSFC) under grant No. 61503092 and by the Tencent Project under grant CCF-TencentRAGR20140114.

References

1. Frequent itemset mining dataset repository. http://fimi.ua.ac.be/data/
2. Aggarwal, C.C.: Managing and mining uncertain Data (2010)
3. Aggarwal, C.C., Yu, P.S.: A survey of uncertain data algorithms and applications. IEEE Trans. Knowl. Data Eng. **21**(5), 609–623 (2009)
4. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules in large databases. In: The International Conference on Very Large Data Bases, pp. 487–499 (1994)
5. Agrawal, R., Srikant, R.: Quest synthetic data generator. http://www.Almaden.ibm.com/cs/quest/syndata.html
6. Ahmed, C.F., Tanbeer, S.K., Jeong, B.S., Le, Y.K.: Efficient tree structures for high utility pattern mining in incremental databases. IEEE Trans. Knowl. Data Eng. **21**(12), 1708–1721 (2009)
7. Bernecker, T., Kriegel, H.P., Renz, M., Verhein, F., Zuefl, A.: Probabilistic frequent itemset mining in uncertain databases. In: ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 119–128 (2009)

8. Chan, R., Yang, Q., Shen, Y.D.: Mining high utility itemsets. In: IEEE International Conference on Data Mining, pp. 19–26 (2003)
9. Chui, C.-K., Kao, B., Hung, E.: Mining frequent itemsets from uncertain data. In: Zhou, Z.-H., Li, H., Yang, Q. (eds.) PAKDD 2007. LNCS (LNAI), vol. 4426, pp. 47–58. Springer, Heidelberg (2007)
10. Geng, L., Hamilton, H.J.: Interestingness measures for data mining: a survey. ACM Comput. Surv. **38** (2006)
11. Han, J., Pei, J., Yin, Y., Mao, R.: Mining frequent patterns without candidate generation: a frequent-pattern tree approach. Data Min. Knowl. Disc. **8**(1), 53–87 (2004)
12. Lin, J.C.W., Gan, W., Hong, T.P., Tseng, V.S.: Efficient algorithms for mining up-to-date high-utility patterns. Adv. Eng. Inform. **29**(3), 648–661 (2015)
13. Lin, J.C.W., Gan, W., Fournier-Viger, P., Hong, T.P., Tseng, V.S.: Mining potential high-utility itemsets over uncertain databases. In: ACM ASE BigData & Social Informatics, p. 25 (2015)
14. Liu, M., Qu, J.: Mining high utility itemsets without candidate generation. In: ACM International Conference on Information and Knowledge Management, pp. 55–64 (2012)
15. Liu, Y., Liao, W., Choudhary, A.K.: A two-phase algorithm for fast discovery of high utility itemsets. In: Ho, T.-B., Cheung, D., Liu, H. (eds.) PAKDD 2005. LNCS (LNAI), vol. 3518, pp. 689–695. Springer, Heidelberg (2005)
16. Microsoft: Example Database foodmart of Microsoft Analysis Services. http://msdn.microsoft.com/en-us/library/aa217032(SQL.80).aspx
17. Fournier-Viger, P., Wu, C.-W., Zida, S., Tseng, V.S.: FHM: faster high-utility itemset mining using estimated utility co-occurrence pruning. In: Andreasen, T., Christiansen, H., Cubero, J.-C., Raś, Z.W. (eds.) ISMIS 2014. LNCS, vol. 8502, pp. 83–92. Springer, Heidelberg (2014)
18. Tseng, V.S., Wu, C.W., Shie, B.E., Yu, P.S.: UP-growth: an efficient algorithm for high utility itemset mining. In: ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 253–262 (2010)
19. Tseng, V.S., Shie, B.E., Wu, C.W., Yu, P.S.: Efficient algorithms for mining high utility itemsets from transactional databases. IEEE Trans. Knowl. Data Eng. **25**(8), 1772–1786 (2013)
20. Wu, C.W., Shie, B.E., Tseng, V.S., Yu, P.S.: Mining top-k high utility itemsets. In: ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 78–86 (2012)
21. Yao, H., Hamilton, H.J., Butz, C.J.: A foundational approach to mining itemset utilities from databases. In: SIAM International Conference on Data Mining, pp. 211–225 (2004)

http://www.springer.com/978-3-319-39936-2

Web-Age Information Management
17th International Conference, WAIM 2016, Nanchang, China, June 3-5, 2016, Proceedings, Part I
Cui, B.; Zhang, N.; Xu, J.; Lian, X.; Liu, D. (Eds.)
2016, XXI, 534 p. 193 illus., Softcover
ISBN: 978-3-319-39936-2