An Efficient Method for Mining Closed Potential High-Utility Itemsets

BAY VO\textsuperscript{1}, (Member, IEEE), LOAN T. T. NGUYEN\textsuperscript{2,3}; NGUYEN BUI\textsuperscript{1}, TRINH D. D. NGUYEN\textsuperscript{4}, VAN-NAM HUYNH\textsuperscript{5}, AND TZUNG-PEI HONG\textsuperscript{6,7}, (Senior Member, IEEE)

\textsuperscript{1}Faculty of Information Technology, Ho Chi Minh City University of Technology (HUTECH), Ho Chi Minh City 700000, Vietnam
\textsuperscript{2}School of Computer Science and Engineering, International University, Ho Chi Minh City 700000, Vietnam
\textsuperscript{3}Vietnam National University, Ho Chi Minh City 700000, Vietnam
\textsuperscript{4}Institute of Research and Development, Duy Tan University, Da Nang 550000, Vietnam
\textsuperscript{5}School of Knowledge Science, Japan Advanced Institute of Science and Technology (JAIST), 923-1292, Japan
\textsuperscript{6}Department of Computer Science and Information Engineering, National University of Kaohsiung, Kaohsiung 811, Taiwan
\textsuperscript{7}Department of Computer Science and Engineering, National Sun Yat-sen University, Kaohsiung 804, Taiwan

Corresponding author: Loan T. T. Nguyen (nttloan@hcmiu.edu.vn)

ABSTRACT High-utility itemset mining (HUIM) has become a key phase of the pattern mining process, which has wide applications, related to both quantities and profits of items. Many algorithms have been proposed to mine high-utility itemsets (HUIs). Since these algorithms often return a large number of discovered patterns, a more compact and lossless representation has been proposed. The recently proposed closed high utility itemset mining (CHUIM) algorithms were designed to work with certain types of databases (e.g., those without probabilities). In fact, real-world databases might contain items or itemsets associated with probability values. To effectively mine frequent patterns from uncertain databases, several techniques have been developed, but there does not exist any method for mining CHUIs from this type of databases. This work presents a novel and efficient method without generating candidates, named CPHUI-List, to mine closed potential high-utility itemsets (CPHUIs) from uncertain databases. The proposed algorithm is DFS-based and utilizes the downward closure property of high transaction-weighted probabilistic mining to prune non-CPHUIs. It can be seen from the experiment evaluations that the proposed algorithm has better execution time and memory usage than the CHUI-Miner.

INDEX TERMS Uncertain database, data mining, knowledge discovery, high-utility itemset, closed high-utility itemset.

I. INTRODUCTION

Mining frequent itemsets (FI) from databases is an important problem in knowledge discovery. Many effective algorithms have been proposed for mining FIs (e.g., Apriori [1], FP-Growth [2]), frequent sequence patterns (e.g., GSP [3]), data streams [4], and high-utility itemsets (HUIs) [5]–[8].

The concept of HUIM was based on the problem of frequent itemset mining (FIM) and was first presented in [9]. An itemset is a HUI if it has utility value not less than a user-specified threshold. The objective of HUIM is to discover a set of patterns that yield high profit (utility). HUIM is considered a more challenging task than FIM since the downward closure property does not hold for utility measure [1], [2]. In addition, the number of HUIs returned and candidates generated during the process of HUIM are often huge, leading to high runtime and memory requirement. To solve this problem, Tseng et al. [10] proposed closed high-utility itemsets (CHUIs), which improve performance of the mining process and allow to derive a complete set of HUIs. An itemset is a CHUI if it has a utility value not less than a user-specified minimum utility threshold and has no supersets which have the same support.

As mentioned, the collected data in real-world applications can have uncertainty, such as locations obtained via RFID devices or GPS [11] or the habits of shoppers obtained from e-commercial websites [12], [13]. Traditional pattern mining algorithms (those for mining FIs or HUIs) are unable to work or to return incorrect outcomes when applying on input data either incomplete or contain erroneous values. There are many algorithms for discovering useful information from uncertain databases. The UApriori approach was...
proposed by Chui et al. [14] to mine FIs from uncertain databases, and it uses a generate-and-test and breadth-first search approach. Later, the UFP-Growth algorithm presented by Leung et al. [15] uses a tree structure called UFP-tree, to mine uncertain FIs without generating candidates. Lin and Hong [16] proposed a tree structure to compress uncertain frequent patterns, named CUF-tree. In addition, an algorithm was also proposed based on this tree structure to mine uncertain FIs from the built tree. PUF-Growth [17] reduces runtime by finding only frequent patterns (i.e., no false negatives and no false positives) from uncertain data. Recently, Lin et al. [18] proposed the PU-List algorithm for mining potential HUIs (PHUIs) from uncertain databases.

The present study proposes a model for mining closed potential high-utility itemset (CPHUI) based on the tuple uncertainty database model [19]. The proposed model is similar to the expected-support-based model [14], [20]. A method, namely CPHUI-List, is then proposed. It utilizes the PEU-List (potential extended utility) structure to mine CPHUIs. This work is an extension to the proposed algorithm in [21]. The main contributions of this paper can be summarized as follows: (i) It proposes a novel type of pattern named CPHUI along with a data structure named PEU-List for mining CPHUIs. (ii) A pruning strategy named Pr-Prune is proposed to prune the search space and reduce the cost of database scans by utilizing the proposed PEU-List. (iii) Based on the proposed PEU-List, Pr-Prune strategy, an effective algorithm named CPHUI-List algorithm is developed to directly mine CPHUIs from uncertain databases.

The rest of this paper is organized as follows. Section 2 presents basic concepts and a problem statement. Section 3 reviews related work on mining HUIs, CHUIs, and FIs from uncertain databases. In Section 4, we develop the PEU-List structure and the CPHUI-List algorithm. Section 5 compares the numbers of patterns produced by the proposed algorithm and CHUI-Miner. Conclusions and ideas for future work are presented in Section 6.

**II. BASIC CONCEPTS AND PROBLEM STATEMENTS**

**A. BASIC CONCEPTS**

Let $\mathcal{D}$ be a transactional uncertain database defined as $\mathcal{D} = \{T_1, T_2, \ldots, T_n\}$. A finite set of distinct items $I = \{i_1, i_2, \ldots, i_m\}$, in which each item $i_j$ appears in a transaction $T_k \in \mathcal{D}$ has a positive value $q(i_j, T_k)$, it is the quantity of purchase of item $i_j \in I$ in transaction $T_k$, each transaction $T_k$ has its own unique transaction ID (TID). Furthermore, each transaction $T_k$ is associated with a unique probability of existence $pe(T_k)$ based on the tuple uncertainty model [19]. A set of positive numbers called $Protable$, an external utility table (e.g., unit profit table), is defined as $Protable = \{p(i_1, p(i_2), \ldots, p(i_m))\}$, where $p(i_j)$ is the profit of item $i_j$. A transaction $T_k \in \mathcal{D}$ contain an itemset $X \subseteq I$ if $X \subseteq T_k$. Two user-specified constraints, the minimum utility threshold (MUT) and the minimum potential probability threshold (MPPT), are called $\gamma$ and $\delta$, respectively.

| Table 1. Example of an uncertain database. |
|-----------------|------------------|------------------|
| TID | Transaction (item, quantity) | Probability |
| 1 | (A, 2); (B, 3); (D, 2) | 0.9 |
| 2 | (B, 1); (C, 2); (E, 3) | 0.8 |
| 3 | (A, 1); (B, 2); (C, 1); (E, 3) | 0.75 |
| 4 | (A, 2); (C, 2) | 0.6 |
| 5 | (B, 3); (C, 2); (E, 1) | 0.7 |
| 6 | (A, 1); (C, 3); (D, 5) | 0.8 |

| Table 2. Unit profits for items in Table 1. |
|-----------------|------------------|
| Item | Unit profit ($) |
| A | 6 |
| B | 3 |
| C | 10 |
| D | 7 |
| E | 15 |

An example of an uncertain database with six transactions is presented in Table 1 along with the unit profits of each item is given in Table 2. In this example, $\gamma = 25\%$, $\delta = 15\%$.

**Definition 1:** Let $X$ and $Y$ be non-empty itemsets. $X$ is a subset of $Y$, and $Y$ is a superset of $X$ if and only if $X \subseteq Y$.

**Definition 2:** The number of transactions containing $X$ in database $\mathcal{D}$, called the support count of itemset $X$, is denoted as $SC(X)$. The tidset of itemset $X$ is the set of identifiers of transactions that contain $X$, denoted as $TidSet(X)$.

Based on Definition 2, $SC(X) = |TidSet(X)|$. For example, for the database in Table 1, $TidSet(BC) = \{2, 3, 5\}$ and $SC(BC) = |TidSet(BC)| = 3$.

**Property 1:** Given an m-itemset $X = \{i_1, i_2, \ldots, i_m\}$, $SC(X) = |TidSet(i_1) \cap TidSet(i_2) \cap \ldots \cap TidSet(i_m)|$.

**Property 2:** If $Y$ is a superset of $X$, then $TidSet(Y) \subseteq TidSet(X)$.

**Definition 3:** The utility of an item $i_j \in I$ in transaction $T_k$ is defined as $u(i_j, T_k) = q(i_j, T_k) \times p(i_j)$.

For example, the utility of item $A$ in transaction $T_1$ is $u(A, T_1) = q(A, T_1) \times p(i_1) = 2 \times 6 = 12$.

**Definition 4:** The existence probability of itemset $X$ in transaction $T_k$, denoted as $pe(X, T_k)$, is defined as $pe(X, T_k) = pe(T_k)$.

For example, the probability of item $A$ in $T_3$ is $pe(A, T_3) = pe(T_3) = 0.7$.

**Definition 5:** The utility of itemset $X$ in transaction $T_k$ is defined as $u(X, T_k) = \sum_{i_j \in X} u(i_j, T_k)$ if $X \subseteq T_k$. The utility of itemset $X$ in uncertain database $\mathcal{D}$ is defined as $u(X) = \sum_{T_k \in \mathcal{D}} u(X, T_k)$, where $k \in TidSet(X)$ [18].

For example, the utility of itemset $BCE$ in transaction $T_2$ is calculated as $u(BCE, T_2) = u(B, T_2) + u(C, T_2) + u(E, T_2) = 3 + 20 + 45 = 68$, and the utility of item $X$ in uncertain database $\mathcal{D}$ is calculated as $u(BCE) = u(BCE, T_2) + u(BCE, T_3) + u(BCE, T_5) = 68 + 61 + 44 = 173$. 


Definition 6: The potential probability of itemset $X$ in an uncertain database $\mathcal{D}$ [18], denoted as $\text{Prob}(X)$, is defined as follows: $\text{Prob}(X) = \sum_{X \subseteq T, \mid k \in \text{TidSet}(X)} pe(X, T_k)$.

For instance, considering the sample database $\mathcal{D}$ given in Table 2, we have $\text{Prob}(A) = pe(A, T_1) + pe(A, T_3) + pe(A, T_4) + pe(A, T_6) = 3.05$.

Definition 7: The utility of transaction $T_k \in \mathcal{D}$ is defined as $TU(T_k) = \sum_{i=1}^{n} u(i, T_k)$.

For example, the transaction utility of transaction $T_1$ is calculated as $TU(T_1) = u(A, T_1) + u(B, T_1) + u(D, T_1) = 12 + 9 + 14 = 35$.

Definition 8: Total utility of uncertain database $\mathcal{D}$ is defined as $\text{sum}(TU) = \sum_{T_k \in \mathcal{D}} TU(T_k)$.

For example, the total utility of $\mathcal{D}$, shown in Table 1, is calculated as $\text{sum}(TU) = \sum_{i=1}^{35} TU(T_i) = 35 + 68 + 67 + 32 + 44 + 111 = 357$.

Definition 9 (Potential High-Utility Itemset): An itemset $X$ is defined as a PHUI if $X$ is a HUI and $X$ has a high existence probability; i.e., $X$ satisfies the following two conditions: $u(X) \geq \gamma \times \text{sum}(TU)$ and $\text{Prob}(X) \geq \delta \times |\mathcal{D}|$.

Definition 10 (Closed Itemset): Let $X$ be an itemset. If there does not exist a superset $Y \supseteq X$ in $\mathcal{D}$ such that $SC(X) = SC(Y)$, then $X$ is called a closed itemset.

Definition 11 (Itemset’s Closure): Let $X$, $Y$ be the closed itemsets, and itemset $Y \supseteq X$ be a superset of itemset $X$. If $SC(Y) = SC(X)$, then $Y$ is a closure of $X$, which is defined as follows: closure($X$) = $\cap_{i \in \text{TidSet}(X)} T_k$ [22].

Definition 12 (Closed Potential High-Utility Itemset): An itemset $X$ is called a CPHUI if $X$ is a CHUI and $X$ has high existence probability; i.e., $X$ is a CHUI if $X$ is closed and $u(X) \geq \gamma \times \text{sum}(TU)$; $X$ is a closed high-potential itemset (CHPI) if $X$ is closed and $\text{Prob}(X) \geq \delta \times |\mathcal{D}|$.

B. PROBLEM STATEMENT

Given an uncertain database $\mathcal{D}$ with total utility $\text{sum}(TU)$, MUT threshold $\gamma$, and MPPT threshold $\delta$. The task of mining CPHUIs from $\mathcal{D}$ is to discover the complete set of closed itemsets having utility not less than $\gamma \times \text{sum}(TU)$ and whose probability is not less than $\delta \times |\mathcal{D}|$.

Consider the database $\mathcal{D}$ given in Table 1, MUT $\gamma = 25\%$ and MPPT $\delta = 15\%$. Table 3 shows the discovered set of CPHUIs from $\mathcal{D}$.

B. CLOSED HIGH-UTILITY ITEMSET MINING

In 2015, Tseng et al. [10] come up with the idea of a more compact and lossless representation of HUIs and the task of mining this type of pattern for the first time. The authors also proposed an algorithm, named CHUD, to mine this new type of itemset representation. The process of mining CHUIs decreases execution time and memory usage. Wu et al. proposed an algorithm named CHUI-Miner [22] for mining CHUIs without generating candidates. The algorithm stores the utility information of itemsets in transactions by utilizing

**TABLE 3. Discovered CPHUIs from $\mathcal{D}$.**

| Itemset | Utility | Prob. |
|---------|---------|-------|
| $C$     | 100     | 3.65  |
| $AC$    | 84      | 2.15  |
| $BCE$   | 173     | 2.25  |

Liu et al. [6] proposed an algorithm named Two-Phase, that can effectively prune the number of candidates to discover the complete set of HUIs. However, this algorithm requires scanning the database multiple times and generates a huge number of candidates. The BAHUI algorithm, proposed by Song et al. [23], uses bitmaps and the divide-and-conquer strategy to mine FIs. BAHUI effectively reduces memory usage and uses efficient bitwise operations. The UP-Growth algorithm and the utility pattern UP-Tree, proposed by Tseng et al. [24], only require only two database scans to discover the complete set of HUIs. The UP-Tree helps reducing the number of generated candidates and significantly reduce the algorithm’s execution time than previously proposed methods. Later, Tseng et al. [5] proposed an improved version of UP-Growth, called UP-Growth+, which reduces overestimated utilities. To minimize the cost of candidate generation and utility calculation from two-phase HUIM algorithms, Qu et al. [25] proposed a novel tree structure for storing candidates and a tree-based algorithm to quickly identify HUIs. Experimental results showed that the proposed candidate tree structure and the algorithm outperforms the performance of two-phase algorithms. The HUI-Miner algorithm, proposed by Liu and Qu [7] uses the utility list structure to prune the search space and thus efficiently mine all HUIs. Lan et al. [26] proposed a projection-based index approach. However, it consumes a high amount of memory usage and requires long execution time to complete. The FHM algorithm, proposed by Fournier-Viger et al. [27], introduced a structure named EUCS (Estimated Utility Co-occurrence Structure) [27] and a corresponding pruning strategy Estimated Utility Co-occurrence Pruning (EUCP) [27] to speed up the discovery of HUIs by considering item co-occurrences. Extending from the HUI-Miner algorithm, Krishnamoorthy proposed the HUP-Miner algorithm [28], in which the input database is partitioned to prune the search space. In addition, it sped up the utility-list construction by utilizing a look ahead strategy [28]. Recently, Zida et al. [8] proposed an algorithm called EFIM to effectively mine HUIs. The authors also proposed two novel and tighter upper bounds, namely sub-tree utility and local utility, and two efficient strategies called high-utility database projection (HDP) and high-utility transaction merging (HTM) to significantly prune the search space and reduce database scans.
the proposed EU-List data structure. Thus, from the complete set of discovered CHUIs, it can derive all the HUIs.

C. POTENTIAL HIGH-UTILITY ITEMSET MINING
When it comes to the context of uncertain databases, FIs now have different denotations. In the case of uncertain database, the support of an itemset is now called the expected support of this itemset. Based on this definition, an itemset is frequent if the expected support is no less than the minimum expected support threshold [20]. Chui and Kao [36] applied frequent itemset mining to datasets under the existential uncertain data model. Bernecker et al. [37] proposed a framework for efficient probabilistic frequent itemset mining. Sun et al. [12] then proposed the p-Apriori algorithm, which is an improvement of the Apriori algorithm [1] for uncertain databases. This algorithm determines the probability of itemset \( X \) being frequent as follows: \( P(\text{sup}(X) \geq \text{minSup}) \geq \text{minPro} \), where \( P \) is the frequency probability and \( \text{minSup} \) is the user-specified minimum support; \( \text{minPro} \) is the probability threshold. Many methods have been presented to mine FIs from uncertain databases. The UApriori algorithm, which was presented by Chui and Kao [14], is a level-wise method and mostly based on the Apriori algorithm. The UFP-Growth algorithm, which was proposed by Leung et al. [15], extended the FP-Tree structure to mine frequent patterns from uncertain databases without generating candidates. Lin and Hong [16] proposed the CUFP-Growth algorithm, which utilizes a tree structure known as the compressed uncertain frequent-pattern tree to mine FIs effectively. Later, Leung and Tanbeer [17] presented the PUF-tree structure and an algorithm named PUF-Growth algorithm, which only requires three probabilistic dataset scans to mine FIs.

Mining high-utility itemsets from an uncertain database is a new and challenging problem, one that is closer to the situation with real-world applications. Lin et al. proposed the first algorithm to mine HUIs from uncertain databases [18] named PHUI-UP, and it is based on the PHUI mining model and very similar to the model used for probabilistic databases. The presented algorithm is based on an Apriori-like [1] approach with a probabilistic measure, and thus it still generates candidates. Furthermore, the authors also proposed the PU-List structure and integrated it into the HUI-Miner algorithm to mine HUIs. The proposed algorithm by Lin et al. is the most recent algorithm to mine HUIs from uncertain databases.

**IV. PROPOSED ALGORITHM**
This section presents the novel PEU-List data structure along with an improved version of the CHUI-Miner algorithm, namely the CPHUI-List algorithm. CPHUI-List is able to mine CPHUIs from uncertain databases. CPHUI-List integrates the proposed PEU-List structure to discover CPHUIs from the set-enumeration tree.

**A. PROBABILITY-EXTENDED-UTILITY-LIST STRUCTURE**
The Apriori property dramatically reduces the number of generated candidates required in the task of mining association rule. For the case of uncertain databases, the property is used for measuring the probability in CPHUI-List to mine CPHUIs.

1) PRUNING STRATEGY BASED ON DOWNWARD CLOSURE PROPERTY

**Theorem 1 (High-Probability Itemsets’s Closure Property):** Let \( X \) be a high probability itemset in an uncertain database. It has the downward closure property: \( \text{Prob}(X') \leq \text{Prob}(X')^{-1} \).

**Definition 14:** The transaction-weighted utilization (TWU) of itemset \( X \) is the sum of the transaction utility of transactions containing \( X \), i.e., \( \text{TWU}(X) = \sum_{\text{TidSet}(X)} \text{TU}(T_k) \).

For example, \( \text{TWU}(A) = \text{TU}(T_1) + \text{TU}(T_3) + \text{TU}(T_4) + \text{TU}(T_6) = 205 \). The TWU values for the remaining items are shown in Table 4.

**Definition 15:** An itemset \( X \) in \( D \) is called a PHUI if \( \text{TWU}(X) \geq \gamma \times \text{sum(TU)} \) and \( \text{Prob}(X) \geq \delta \times |D| \).

Item \( i \in I \) is considered promising if \( \text{TWU}(i) \geq \text{min}_{util} \) and \( \text{Prob}(i) \geq \text{min}_\text{prob} \). Other items are called unpromising items [22].

2) PEU-LIST STRUCTURE
To the efficiently and directly mine CPHUIs from uncertain databases and reducing database scans, the information of itemsets in transactions is recorded in PEU-List. A PEU-List is initiated by scanning the database two times. The PEU-List structure of itemset \( X \) comprise of a PU-List [18], the support of \( X \), and two itemsets, namely \( \text{PrevSet}(X) \) and \( \text{PostSet}(X) \), which are the itemsets precede and succeed \( X \), respectively. The PU-List of \( X \) contains lists of items. Every list in the PU-List of \( X \) contains the utility and probability of promising items.

**Definition 16 (Precede and Succeed):** Consider a set of items \( I = \{i_1, i_2, \ldots, i_n\} \) and the total order relation \( R : i_1 < i_2 < \ldots < i_n \). An item \( i_j \) precedes item \( i_k \) (\( 1 \leq j \leq k \leq n \)). In contrast, \( i_j \) succeeds \( i_k \).

**Definition 17 (PU-List):** The PU-List [18] of itemset \( X \), denoted as \( \text{PUL}(X) \), is an ordered list containing \( |\text{TidSet}(X)| \) tuples in the form \( \langle \text{TID}, \text{IU}(X, T_k), \text{RU}(X, T_k), \text{PR}(X, T_k) \rangle \), where \( \text{TID} \) is the tid of \( X \), \( \text{IU}(X, T_k) \) is the utility of \( X \) in \( T_k \), defined as \( \text{IU}(X, T_k) = \sum_{\text{X} \subseteq T_k} u(i, T_k) \). \( \text{RU}(X, T_k) \) is the

### Table 4. TWU value of items, sorted in ascending order.

| Item | TWU |
|------|-----|
| D    | 106 |
| E    | 179 |
| A    | 205 |
| B    | 214 |
| C    | 282 |
remaining utility of $X$ in $T_q$, defined as $RU(X, T_k) = \sum_{i \in X \cap \gamma(i)} u(i, T_k)$, and PR is defined as $PR(X, T_k) = pe(X, T_k)$.

For instance, Figure 1 presents the PU-List constructed for one-member itemsets (a), two-member itemsets (b), and three-member itemsets (c). In this example, $\prec$ is the ascending order of item’s TWUs ($D < E < A < B < C$).

Definition 18: The PU-List structure in Definition 17, denoted as $\sum(IU)$, is the sum of utilities of $X$ in $D$ and $\sum(RU)$ is the sum of remaining utilities of $X$ in $D$. In this:
- $X. \sum(IU) = \sum_{i \subseteq T_k \land T \in D} IW(X, T_k)$
- $X. \sum(RU) = \sum_{i \subseteq T_k \land T \in D} RU(X, T_k)$

Pruning strategy (Pr-Prune). If $X. \sum(IU) + X. \sum(RU)$ is less than $\minutil$ or $\Prob(X)$ is less than $\minprob$, then $X$ is not a CPHUI; therefore, $X$ is not a CPHUI.

Definition 19 (Subsume): Itemset $Y$ is subsumed by itemset $X$ if $Y \subseteq X$ and $SC(Y) = SC(X)$ [22].

For example, $E$ is subsumed by $BCE$ because $E \subseteq BCE$ and $SC(E) = SC(BCE)$.

Property 3: For two itemsets $Y$ and $X$, if $Y \subseteq X$ and $SC(Y) = SC(X)$, then $closure(X) = closure(Y)$.

3) SET-ENUMERATION TREE

As mentioned above, the search space of the proposed algorithm can be represented logically as a set-enumeration tree and is defined as follows. An itemset is represented as a node in the tree and is an extension of its parent node. The root node of the tree is $\emptyset$, nodes at level 1 are 1-itemsets. Nodes are arranged from left to right based on the total order relation $R$, where $R$ is in descending order of TWUs of 1-itemsets.

The algorithm CPHUI-List traverses the tree using depth-first search (DFS) strategy. Each node of the set-enumeration tree is a member of the PU-List, which consists of four members, namely $(i, IU, RU, PR)$. The pruning strategy

Algorithm 1 CPHUI-List Algorithm

Input: Uncertain database $D$, $\gamma$: MUT, $\delta$: MPPT;
Output: Complete set of discovered CPHUIs;
1: Scan $D$ to calculate $TWU(i)$, $\Prob(i)$ for every item $i \in I$;
2: Construct set of promising 1-itemsets $I_p = \{i \in I | TWU(i) \geq \minutil, \Prob(i) \geq \minprob\}$;
3: Sort $I_p$ by $TWU$ in ascending order;
4: Scan $D$ again to construct PEU-List of promising items;
5: GEN-CPHUI($\emptyset, \emptyset, I_p, \epsilon, \mu$);
6: return CPHUIs;

Pr-Prune is used to eliminate branches whose child node is an unpromising itemset. If a node satisfies two conditions, namely the sum of $IU$ and $RU$ of the processed node is more than or equal to the $MUT(\gamma \times \sum(TU))$ and the $PR$ of the node is more than or equal to the $MPPT(\delta \times |D|)$, then its subset can be a CPHUI. The traversal is done recursively to extend the set-enumeration tree. The utility and potential probability are recorded in the PEU-List. Finally, CPHUIs can be mined without multiple database scans.

B. PROPOSED CPHUI-LIST ALGORITHM

The first step of CPHUI-List, whose pseudo-code is given in Algorithm 1, is to scan the input database $D$ to calculate $TWU(i)$, $\Prob(i)$ of 1-itemsets $i \in I$. Unpromising 1-itemsets are pruned if $TWU(i) \leq \minutil$ or $\Prob(i) \leq \minprob$, and the set of remain items called $I_p$. Next, items in $I_p$ are sorted based on ascending order of $TWUs$. Then, the algorithm performs a second scan on $D$ and constructs the PEU-List structure of 1-itemsets. The set-enumeration tree can be used to represent the search space of the algorithm. Every 1-itemset $i \in I_p$ and its PEU-List is recursively explored by the DFS.
The high utility and subsumed checking procedure of $X$ is then carried out (lines #7 and #8). The subsumed checking procedure is shown in Pseudo-Code 4. If generated itemset $new_gen$ passes the subsumed check, the ComputeClosure procedure with the following parameters: $(new_gen, PostSet(Closed_setnew), PostSet(X))$ is then invoked in order to calculate the closure of $new_gen$ and its PEU-List by updating the PEU-List of $new_gen$. The algorithm for computing the itemset’s closure is given in Pseudo-Code 2.

V. EXPERIMENTAL STUDIES

Experiments were carried out to evaluate the performance of the proposed CPHUI-List algorithm. They were conducted on a computer equipped with a 3rd-generation 64-bit

---

**Pseudo-Code 1** PEU-List Construction

**Input:** PU-List of itemset $X$, PU-List of 1-item $i$;  
**Output:** PEUL($new_gen$): PEU-List of $new_gen$;  
1: for each element $E_i \in$ PUL($X$) do  
2: if $\exists E_z \in$ PUL ($i$) $\land E_z.tid = E_i.tid$ then  
3: $E_{new_gen} \leftarrow \{E_i.tid, E_i.IU + E_z.IU, E_i.RU, E_z.PR\}$;  
4: PUL($new_gen$) $\leftarrow$ PUL($new_gen$) $\cup$ $E_{new_gen}$;  
5: end if  
6: end for  
7: TidSet($new_gen$) $\leftarrow$ PUL($new_gen$).getTids();  
8: SC($new_gen$) $\leftarrow$ TidSet($new_gen$);  
9: PreSet($new_gen$) $\leftarrow$ PreSet($X$);  
10: PEU_List($new_gen$) $\leftarrow$ (PUL($new_gen$), SC($new_gen$), VP($new_gen$), PreSet(new_gen), $\emptyset$)  
11: return PEUL($new_gen$);

---

**Pseudo-Code 2** ComputeClosure Algorithm

**Input:** $new_gen$, PostSet($new_gen$), PreSet($X$);  
**Output:** Closed_setnew: closure of $new_gen$ element, update PostSet(Closed_setnew) and PEU-List of Closed_setnew;  
1: Closed_setnew $\leftarrow$ new_gen;  
2: PostSet(new_gen) $\leftarrow$ $\emptyset$;  
3: for each item $i \in$ PostSet($X$) do  
4: if TidSet(Closed_setnew) $\subset$ TidSet($X$) then  
5: Closed_setnew $\leftarrow$ Closed_setnew $\cup$ $i$;  
6: for each Tid $r$ $\in$ TidSet(Closed_setnew) do  
7: PUL(Closed_setnew, $T_r$).IU $\leftarrow$ PUL(Closed_setnew, $T_r$).IU + PUL($i$, $T_r$).RU;  
8: PUL(Closed_setnew, $T_r$).RU $\leftarrow$ PUL($i$, $T_r$).RU  
9: end for  
10: else  
11: PostSet(Closed_setnew) $\leftarrow$ PostSet(Closed_setnew) $\cup$ $i$  
12: end if  
13: end for  
14: return Closed_setnew;

---

**Pseudo-Code 3** GEN-CPHUI Algorithm

**Input:** itemset $X$, PostSet($X$), PreSet($X$), $\gamma$: MUT, $\delta$: MPPT;  
**Output:** CPHUIs;  
1: while PostSet($X$) $\neq$ $\emptyset$ do  
2: $i \leftarrow$ min$_r$.($PostSet(X)$);  
3: $PostSet(X) \leftarrow$ PostSet($X$) $i$;  
4: $new_gen \leftarrow X \cup i$;  
5: Construct PEU-List of $new_gen$; $\rightarrow$ Build PEU-List for $new_gen$  
6: $PreSet(new_gen) \leftarrow$ PreSet($X$);  
7: if $X.sum(IU) + X.sum(RU) \geq \gamma$ and $Prob(X) \geq \delta$ then  
8: if $\neg$ is_duplicate($new_gen$, PreSet($X$)) then  
9: PEUL(Closed_setnew) $\leftarrow$ PEUL($new_gen$)  
10: $PreSet(Closed_setnew) \leftarrow PreSet(new_gen)$;  
11: $PostSet(Closed_setnew) \leftarrow \emptyset$  
12: Closed_setnew $\leftarrow$ ComputeClosure($new_gen$, $PostSet(Closed_setnew)$, $PostSet(Closed_setnew)$, $\emptyset$);  
13: if $X.sum(IU) \geq \gamma$ and $Prob(X) \geq \delta$ then  
14: Output Closed_setnew, SC(Closed_setnew);  
15: end if  
16: GEN-CPHUI($Closed_setnew$, $PostSet(Closed_setnew)$, $PostSet(Closed_setnew)$, $\emptyset$);  
17: $PreSet(X) \leftarrow PreSet(X) \cup i$;  
18: end if  
19: end if  
20: end while  
21: return CPHUIs;

---

**Pseudo-Code 4** Subsume Checking Algorithm

**Input:** $new_gen$, PreSet($new_gen$);  
**Output:** false if $new_gen$ is subsumed by already mined closed itemset;  
1: for each item $j \in$ PreSet($new_gen$) do  
2: if TidSet($new_gen$) $\subseteq$ TidSet($j$) then  
3: return true;  
4: end if  
5: end for  
6: return false;
TABLE 5. Characteristics of datasets.

| Dataset | #transactions | #items | Avg. length | Type |
|---------|---------------|--------|-------------|------|
| Foodmart | 4,141 | 1,559 | 4.4 | Sparse |
| Retail | 88,162 | 16,470 | 10.3 | Sparse |
| Chainstore | 1,112,949 | 46,086 | 7.2 | Sparse |
| Mushroom | 8,124 | 119 | 23 | Dense |
| Chess | 3,196 | 75 | 43 | Dense |
| Accidents | 340,183 | 468 | 33.8 | Dense |

FIGURE 2. Runtimes of CHUI-Miner and CPHUI-List for various MUTs and fixed MPPT for various datasets.

FIGURE 3. Runtimes for CHUI-Miner and CPHUI-List for various MPPTs and fixed MUT for various datasets.

FIGURE 4. Memory consumption for various MUTs and fixed MPPT for various datasets.

Intel®Core-i5™2.7GHz processor, having 16 GB RAM, operated by Windows 10 Professional edition. The performance of the algorithm CPHUI-List was then compared to that of the CHUI-Miner algorithm. All algorithms were written in Java, using JDK 1.8.0. The initial and maximum heap memory sizes of the JVM were fixed at 4GB. The source code for CHUI-Miner can be downloaded in [38].

A. DATASETS

Experiments were performed on the following datasets: Foodmart, Chainstore, Mushroom, Retail, Chess and Accident. Mushroom, Chess, Retail and Accident are available from the FIMI Repository, and Foodmart is the Microsoft Foodmart 2000 database. The characteristics of these datasets are given in Table 5. For all datasets, the external utilities of items were generated between 1 and 1,000 using a log-normal distribution. The quantities of items were randomly generated between 1 and 5. The probabilities were randomly generated to be between 0.1 and 1.0.

B. RUNTIME

The runtime of the proposed CPHUI-List algorithm was compared with that of CHUI-Miner using six uncertain databases.
and various MUT and MPPT thresholds. Figure 2 presents the runtime of the two algorithms for various MUTs and a fixed MPPT. In reverse, Figure 3 presents the runtime of the evaluated algorithms for various MPPTs and a fixed MUT. Since the CHUI-Miner is an algorithm designed to operate on certain datasets, not uncertain ones, thus it does not handle potential probability values. In Figure 2, it can be seen that CPHUI-List has better execution time than CHUI-Miner on all tested datasets. For example, for the Chess dataset at minutil = 20%, CPHUI-List is faster than CHUI-Miner for over 20 times. From Figure 3, it can be observed that CPHUI-List is also much faster than CHUI-Miner. As we increase the value of MPPT, the runtime time of CPHUI-List is decreased, while that of CHUI-Miner remained unchanged. This can be addressed to the higher MPPT values, such that CPHUI-List generated fewer candidates while the performance of CHUI-Miner is affected only by varying MUT values, not MPPTs. In this case, MUT is fixed on each dataset.

C. PATTERN ANALYSIS
Pattern analysis was also done to compare our proposed algorithm to CHUI-Miner. The number of discovered CPHUIs is always lower than that of CHUIs for a fixed MPPT and various MUT. This can be addressed to the CPHUI-List mines CPHUIs by considering both the utility and probability constraints, whereas CHUI-Miner mines CHUIs (i.e., it only checks the utility constraint). The numbers of CHUIs and CPHUIs both decrease as the MUT increases. It also can be observed that the number of CHUIs is unchanged when the MPPT is raised. The reason is that CHUI-Miner only considers the high utility threshold for discovering CHUIs, and thus when the MPPT is raised, it does not affect the number of CHUIs. As we increase the MPPT, the number of CPHUIs drops dramatically. Since the proposed algorithm utilizes two thresholds for mining CPHUIs, thus the number of discovered patterns is always lower than that of CHUIs.

D. MEMORY CONSUMPTION
The memory consumption for the two algorithms is shown in Figures 4 and 5. For both a fixed MPPT and various MUT values and a fixed MUT and various MPPT values, CPHUI-List consumed less memory than CHUI-Miner. The reason for this result is the same as that given for Figure 4. From Figure 5, when minprob = 2% for the Mushroom dataset, the memory consumed by CPHUI-List is less than one-fifth that consumed by CHUI-Miner.

VI. CONCLUSION AND FUTURE WORK
This study proposed an approach for mining CPHUIs from uncertain databases without generating candidates. The CPHUI-List algorithm mines HUIs and high-probability itemsets. The algorithm is developed using the PEU-List structure and the set-enumeration tree as basis. It uses the DFS-based approach to directly mine CPHUIs. Experimental results for real databases show that CPHUI-List outperforms CHUI-miner in terms of runtime and consumed memory.

This is the first step towards mining CPHUIs from uncertain databases. In the future, we will improve the structure of PEU-List to improve the performance of the CPHUI-List algorithm and extend this work to mine maximal-potential HUIs and top-k-potential HUIs.

REFERENCES
[1] R. Agrawal and R. Skirant, “Fast algorithms for mining association rules in large databases,” in Proc. 20th Int. Conf. Very Large Database, 1994, pp. 487–499.
[2] J. Han, J. Pei, Y. Yin, and R. Mao, “Mining frequent patterns without candidate generation: A frequent-pattern tree approach,” Data Mining Knowl. Discovery, vol. 8, no. 1, pp. 53–87, Jan. 2004.
[3] R. Agrawal and R. Srikant, “Mining sequential patterns,” in Proc. 11th Int. Conf. Data Eng., 1995, pp. 3–14.
[4] C. Leung and Q. Khan, “DSTree: A tree structure for the mining of frequent sets from data streams,” in Proc. 6th Int. Conf. Data Mining (ICDM), Dec. 2006, pp. 928–932.
[5] V. S. Tseng, B.-E. Shie, C.-W. Wu, and P. S. Yu, “Efficient algorithms for mining high utility itemsets from transactional databases,” IEEE Trans. Knowl. Data Eng., vol. 25, no. 8, pp. 1772–1786, Aug. 2013.
[6] Y. Liu, W. Liao, and A. Choudhary, “A two-phase algorithm for fast discovery of high utility itemsets,” in Proc. 9th Pacific–Asia Conf. Knowl. Discovery Data Mining, 2005, pp. 689–695.
[7] M. Liu and J. Qu, “Mining high utility itemsets without candidate generation,” in Proc. 21st ACM Int. Conf. Inf. Knowl. Manage. (CIKM), 2012, pp. 55–64.
[8] S. Zida, P. Fournier-Viger, J. C.-W. Lin, C.-W. Wu, and V. S. Tseng, “EFIM: A fast and memory efficient algorithm for high-utility itemset mining,” Knowl. Inf. Syst., vol. 51, no. 2, pp. 595–625, May 2017.
[9] R. J. Hilderman, C. L. Carter, H. J. Hamilton, and N. Cercone, “Mining market basket data using share measures and characterized itemsets,” in Proc. PAKDD, 1998, pp. 159–173.
B. Vo et al.: Efficient Method for Mining CPUIs

[10] V. S. Tseng, C.-W. Wu, P. Fournier-Viger, and P. S. Yu, “Efficient algorithms for mining the concise and lossless representation of high utility itemsets,” IEEE Trans. Knowl. Data Eng., vol. 27, no. 3, pp. 726–739, Aug. 2015.

[11] C. C. Aggarwal and P. S. Yu, “A survey of uncertain data algorithms and applications,” IEEE Trans. Knowl. Data Eng., vol. 21, no. 5, pp. 609–623, May 2009.

[12] L. Sun, R. Cheng, D. Cheung, and J. Cheng, “Mining uncertain data with probabilistic guarantees,” in Proc. 16th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, 2010, pp. 273–282.

[13] L. Wang, L. Feng, and M. Wu, “AT-Mine: An efficient algorithm of frequent itemset mining on uncertain dataset.” J. Comput., vol. 8, no. 6, pp. 1417–1426, May 2013.

[14] C.-K. Chui, B. Kao, and E. Hung, “Mining frequent itemsets from uncertain data,” in Proc. Pacific–Asia Conf. Knowl. Discovery Data Mining, Jun. 2007, pp. 47–58.

[15] C. K.-S. Leung, M. A. F. Mateo, and D. A. Brajčučak, “A tree-based approach for frequent pattern mining from uncertain data,” in Proc. Pacific–Asia Conf. Knowl. Discovery Data Mining, May 2008, pp. 653–661.

[16] C.-W. Lin and T.-P. Hong, “A new mining approach for uncertain databases using CUEP trees,” Expert Syst. Appl., vol. 39, no. 4, pp. 4084–4093, Mar. 2012.

[17] C. K.-S. Leung and S. K. Tanbeer, “PUF-tree: A compact tree structure for frequent pattern mining of uncertain data,” in Proc. 17th Pacific–Asia Conf. Knowl. Discovery Data Mining, 2013, pp. 13–25.

[18] J.-C.-W. Lin, W. Gan, P. Fournier-Viger, T.-P. Hong, and V. S. Tseng, “Efficient algorithms for mining high-utility itemsets in uncertain databases,” Knowl-Based Syst., vol. 96, pp. 171–187, Mar. 2016.

[19] D. Nilesh and S. Dan, “Efficient query evaluation on probabilistic databases,” VLDB J., vol. 16, no. 4, pp. 523–544, 2007.

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

[21] N. Bui, B. Vo, V.-N. Huynh, C.-W. Lin, and L. T. T. Nguyen, “Mining closed high utility itemsets in uncertain databases,” in Proc. 7th Symp. Inf. Commun. Technol., Ho Chi Minh City, Vietnam, 2016, pp. 7–14.

[22] C.-W. Wu, P. Fournier-Viger, J.-Y. Gu, and V. S. Tseng, “Mining closed+ high utility itemsets without candidate generation,” in Proc. Conf. Technol. Appl. Artif. Intell. (TAAI), Nov. 2015, pp. 187–194.

[23] W. Song, Y. Liu, and J. Li, “BHAUFI: Fast and memory efficient mining of high utility itemsets based on bitmap,” Int. J. Data Warehousing Mining, vol. 10, no. 1, pp. 1–15, Jan. 2014.

[24] V. S. Tseng, C.-W. Wu, B.-E. Shie, and P. S. Yu, “UP-growth: An efficient algorithm for high utility itemset mining,” in Proc. 16th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, 2010, pp. 253–262.

[25] J.-F. Qu, M. Liu, C. Xin, and Z. Wu, “Fast identification of high utility itemsets from candidates,” Information, vol. 9, no. 5, pp. 119, May 2018.

[26] G.-C. Lan, T.-P. Hong, and V. S. Tseng, “An efficient projection-based indexing approach for mining high utility itemsets,” Knowl. Inf. Syst., vol. 38, no. 1, pp. 85–107, Jan. 2014.

[27] P. Fournier-Viger, C.-W. Wu, S. Zida, and V. S. Tseng, “FHMI: Faster high-utility itemset mining using estimated utility co-occurrence pruning,” in Proc. 21st Int. Symp. Methodol. Intell. Syst. (ISMIS), Cham, Switzerland: Springer, 2014, pp. 83–92.

[28] S. Krishnamoorthy, “Pruning strategies for mining high utility itemsets,” Expert Syst. Appl., vol. 42, no. 5, pp. 2371–2381, Apr. 2015.

[29] T.-P. Hong, C.-H. Lee, and S.-L. Wang, “Effective utility mining with the measure of average utility,” Expert Syst. Appl., vol. 38, no. 7, pp. 8259–8265, Jul. 2011.

[30] C.-W. Lin, T.-P. Hong, and W.-H. Lu, “Efficiently mining high average utility itemsets with a tree structure,” in Proc. Asian Conf. Intell. Inf. Database Syst., 2010, pp. 131–139.

[31] T. Lu, B. Vo, H. Nguyen, and T.-P. Hong, “A new method for mining high average utility itemsets,” in Proc. IFIP Int. Conf. Comput. Syst. Ind. Manage., 2014, pp. 33–42.

[32] J.-C.-W. Lin, T. Li, P. Fournier-Viger, T.-P. Hong, and J.-H. Su, “Efficient mining of high average-utility itemsets with multiple minimum thresholds,” in Proc. Ind. Conf. Data Mining, 2016, pp. 14–28.

[33] J. C.-W. Lin, S. Ren, P. Fournier-Viger, and T.-P. Hong, “EHAUMP: Efficient high average-utility pattern mining with tighter upper bounds,” IEEE Access, vol. 5, pp. 12927–12940, 2017.

[34] J. M.-T. Wu, J. C.-W. Lin, M. Pirouz, and P. Fournier-Viger, “TUB-HAUMP: Tighter upper bound for mining high average-utility patterns,” IEEE Access, vol. 6, pp. 18655–18669, 2018.

[35] B. Zhang, J. Lin, Y. Shao, P. Fournier-Viger, and Y. Djenouri, “Maintenance of discovered high average-utility itemsets in dynamic databases,” Appl. Sci., vol. 8, no. 5, p. 769, May 2018.

[36] C.-K. Chui and B. Kao, “A decremental approach for mining frequent itemsets from uncertain data,” in Proc. 12th Pacific–Asia Conf. Knowl. Discovery Data Mining, May 2008, pp. 64–75.

[37] T. Bernecker, H.-P. Kriegel, M. Renz, F. Verhein, and A. Zuefle, “Probabilistic frequent itemset mining in uncertain databases,” in Proc. 15th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD), 2009, pp. 119–128.

[38] P. Fournier-Viger, A. Gomariz, T. Gueniche, A. Soltani, C.-W. Wu, and V. S. Tseng, “SPMF: A java open-source pattern mining library,” J. Mach. Learn. Res., vol. 15, no. 1, pp. 3389–3393, 2014.
VAN-NAM HUYNH received the B.S. degree in mathematics from Quy-Nhon University, in 1990, the Ph.D. degree in mathematics from the Institute of Information Technology, Vietnam Academy of Science and Technology, in 1999, and the Habilitation in informatics from the Université de Technologie de Compiègne, France, in 2012. After serving as a Lecturer with the Departments of Mathematics and Computer Science, Quy-Nhon University, for over ten years. In 2003, he joined the School of Knowledge Science, Japan Advanced Institute of Science and Technology (JAIST), where he rose through the ranks and is currently a Professor. He was also an Adjunct Professor with Chiang Mai University, Thailand, from 2015 to 2017, and a part-time Lecturer with Tsukuba University, Japan, from 2011 to 2015. He has (co-)edited over fifteen Springer volumes and seven special issues for the International Journal of Approximate Reasoning, Annals of Operations Research, and Data and Knowledge Engineering. His current research interests include decision analysis and management science, data mining and machine learning, modeling and reasoning with uncertain knowledge, multiagent systems, and kansei information processing and applications.

Dr. Huynh was a recipient of the Postdoctoral Fellowship from the Inoue Foundation for Science, Japan, in the fiscal year 2001. He currently serves as an Associate Editor for the International Journal of Approximate Reasoning (Elsevier), the International Journal of Knowledge and Systems Science (IGI Global), and Array (Elsevier).

TZUNG-PEI HONG received the B.S. degree in chemical engineering from National Taiwan University, in 1985, and the Ph.D. degree in computer science and information engineering from National Chiao-Tung University, in 1992. He was with the Department of Computer Science, Chung-Hua Polytechnic Institute, from 1992 to 1994, and the Department of Information Management, I-Shou University, from 1994 to 2001. He was in charge of the whole computerization and library planning with the National University of Kaohsiung in Preparation, from 1997 to 2000. He was the First Director of the Library and Computer Center, National University of Kaohsiung, from 2000 to 2001, the Dean of Academic Affairs, from 2003 to 2006, the Administrative Vice President, from 2007 to 2008, and the Academic Vice President, in 2010. He is currently a Distinguished and Chair Professor with the Department of Computer Science and Information Engineering and the Department of Electrical Engineering, and the Director of the AI Research Center, National University of Kaohsiung, Taiwan. He has published more than 600 research articles in international/national journals and conferences, and planned more than 50 information systems. He is also the Board Member of more than 40 journals and the Program Committee Member of more than 500 conferences. His current research interests include knowledge engineering, data mining, soft computing, management information systems, and www applications.

Dr. Hong received the First National Flexible Wage Award from the Ministry of Education, Taiwan.