Hindawi
Mathematical Problems in Engineering
Volume 2017, Article ID 8576829, 13 pages
https://doi.org/10.1155/2017/8576829

Research Article

PHUIMUS: A Potential High Utility Itemsets Mining Algorithm Based on Stream Data with Uncertainty

Ju Wang, Fuxian Liu, and Chunjie Jin

Air Force Engineering University, Xi'an, China

Correspondence should be addressed to Ju Wang; yonglingjuke@outlook.com

Received 8 October 2016; Revised 16 February 2017; Accepted 23 February 2017; Published 16 March 2017

Academic Editor: Haipeng Peng

High utility itemsets (HUIs) mining has been a hot topic recently, which can be used to mine the profitable itemsets by considering both the quantity and profit factors. Up to now, researches on HUIs mining over uncertain datasets and data stream had been studied respectively. However, to the best of our knowledge, the issue of HUIs mining over uncertain data stream is seldom studied. In this paper, PHUIMUS (potential high utility itemsets mining over uncertain data stream) algorithm is proposed to mine potential high utility itemsets (PHUIs) that represent the itemsets with high utilities and high existential probabilities over uncertain data stream based on sliding windows. To realize the algorithm, potential utility list over uncertain data stream (PUS-list) is designed to mine PHUIs without rescanning the analyzed uncertain data stream. And transaction weighted probability and utility tree (TWPUS-tree) over uncertain data stream is also designed to decrease the number of candidate itemsets generated by the PHUIMUS algorithm. Substantial experiments are conducted in terms of run-time, number of discovered PHUIs, memory consumption, and scalability on real-life and synthetic databases. The results show that our proposed algorithm is reasonable and acceptable for mining meaningful PHUIs from uncertain data streams.

1. Introduction

Knowledge discovery in databases (KDD) is an emerging issue since the important, implicit, unknown, and potential useful information can be found from huge databases [1, 2]. And frequent itemsets mining (FIM), which is used to mine the frequent itemsets that their occurrence frequencies are no less than minimum support threshold, is one of the most important and common tasks of data mining [3]. Apriori [4] based on bread first search and FP-growth [2] based on depth first search are well-known fundamental FIM algorithms. However, these traditional FIM algorithms assume that the profit of every item is the same and the frequency value of every item in transactions is 0 or 1. In real-life applications, the itemsets that bring high profit to retailers and managers are useful [5], not the most frequent itemsets. Thus, factors like quantity, price, and profit are needed to be included in the FIM.

To deal with the limitations of FIM, Chan et al. [6] first proposed high utility itemsets mining algorithm over the nonbinary databases with different profit values of items. The goal of HUIs mining is to discover itemsets that bring considerable profit to users, although they are not frequent itemsets. Aiming at the issue of HUIs mining, level-wise approaches [6, 7], pattern growth approaches [8–10], and list based approaches [11, 12] are three main frameworks to deal with the problem of undownward closure property and combinational explosion about it.

These traditional HUIs mining algorithms are proposed to deal with static databases, which ignore itemsets' timeliness. Therefore, Tseng et al. first proposed THUI-Mine [13] to mine HUIs from data stream according to the two-phase model based on sliding windows. Afterward, lots of improved algorithms [14–18] are proposed to handle this problem more efficiently. However, the above algorithms can only deal with the precise data streams, and they could not deal with uncertainty.

In real-life applications, while the data is collected from noisy data sources, uncertainty may be introduced. But most HUIs mining algorithms are developed to handle precise databases, which ignore itemsets' existential probability. In fact, for the uncertain databases, itemsets with high utility and high existential probability are useful to users, not itemsets with only one of them. To the best of our knowledge, Lin et al. [19] proposed PHUI-UP based on two-phase model and PHUI-List based on list structure, Lan et al. [20] proposed UHUI-apriori based on Apriori, and these are only algorithms that used to solve HUIs mining problem over uncertain databases. However, the above algorithms can only handle static data with uncertainty, and they could not deal with uncertain data stream.

Uncertain data streams, where the transactions data are added constantly, having the feature of continuous, unlimited, and uncertainty, play an important role in the real-life applications as they exist everywhere, such as wireless sensor, GPS, WIFI system, and RFID. But the issue of HUIs mining over uncertain data stream is seldom studied. Note that HUIs mining over uncertain data stream has to satisfy the following requirements. (1) The analyzed uncertain data stream can be scanned only once. (2) Memory usage for the mining process should be limited in the acceptable range. (3) All the data must be processed as fast as possible. (4) Itemsets with high utility and high existential probability can be output whenever users want the results.

In this paper, to deal with the new issue of HUIs mining over uncertain data stream, PHUIMUS algorithm is proposed to mine PHUIs over uncertain data stream based on sliding windows. For the realization of PHUIMUS, PUS-list is designed to keep exact potential utility of items and transactions, and TWPUS-tree is developed to maintain batch-by-batch information inside the nodes. Major contributions of this paper are summarized as follows:

(1) Previous works about HUIs mining mainly focus on the issue of mining HUIs efficiently in the static and precise databases. To the best of my knowledge, seldom researches are conducted to deal with the issue of mining HUIs over uncertain data stream that takes both uncertainty and timeliness into account.

(2) As HUIs mining over uncertain data stream brings existential probability and sliding windows into consideration, the calculation of items utility, itemsets utility, transaction utility, and transaction weighted utility is changed. In this paper, new definitions about them are given, and a novel type of itemsets named PHUIs is designed.

(3) PHUIMUS algorithm is proposed to mine PHUIs over uncertain data stream based on the developed PUS-list and TWPUS-tree in the current window, which can efficiently prune the unpromising itemsets and get PHUIs without rescanning the analyzed uncertain data stream.

(4) Substantial experiments have been conducted on real-life and synthetic databases. Results show that the designed algorithm can effectively discover PHUIs over uncertain data stream and has a good performance on run-time, number of discovered PHUIs, memory consumption, and scalability.

The remainder of this paper is organized as follows. In Section 2, we describe the related work. In Section 3, we present our new definitions and the problem of HUIs mining over uncertain data stream. In Section 4, we develop our proposed PUS-list and TWPUS-tree and design PHUIMUS algorithm to the stated problem. In Section 5, our experimental results are presented and analyzed. Finally, in Section 6, conclusions are drawn.

2. Related Work

In this section, related work about HUIs mining over data stream and uncertain database are briefly reviewed, respectively.

2.1. HUIs Mining over Data Stream. As an expansion of FIM, HUIs mining focuses on finding itemsets whose utilities are not lower than a minimum utility threshold which has been widely studied recently, which can be used in various areas, such as web click analysis, biological gene analysis, and retail marketing [18]. Its goal is to discover items or itemsets in transactions that are valuable to users, not the most frequent ones. Aiming at the issue of HUIs mining, several typical algorithms had been proposed to deal with the problem of undownward closure property and combinational explosion about it, such as two-phase model [7], HUP-growth [9], HUI-Miner [11], HUPEumu-GRAM [21], and HUIs mining-BPSO [22].

In contrast to discovering HUIs from static database, THUI-Mine [13] is the first algorithm for mining HUIs from data stream according to the two-phase model based on sliding windows [7] and thus suffers from the problem of level-wise candidate generation. Afterward, to reduce the number of THUI-Mine's candidate itemsets, Li et al. [14, 15] proposed MHUI-BT and MHUI-TID by using bit vectors and TID-lists for each distinct item. The experiments show that MHUI-TID is an outstanding algorithm for mining HUIs from data stream since TIDlist is an efficient data structure that can reduce the number of candidate itemsets sharply.

Moreover, Shie et al. [16] proposed efficient algorithms for mining maximal HUIs from data streams with different models. Ahmed et al. [17] designed an interactive mining algorithm of high utility patterns over data streams. Zihayat and An [18] suggested an algorithm in mining top-k high utility patterns over data streams. However, the above algorithms can only deal with the precise data streams, and they could not deal with uncertainty.

2.2. HUIs Mining over Uncertain Database. It is assumed by most HUIs mining algorithms that the information stored in the databases is precise, which ignore itemsets' existential probability. Thus, traditional HUIs mining algorithms are insufficient to process transactions with uncertainty in real-life applications. In fact, for the uncertain database, itemsets with high utility and high existential probability are useful to users, not itemsets with only one of them. To the best of our knowledge, Lin et al. [19] proposed PHUI-UP based on two-phase model and PHUI-List based on list structure, Lan et al. [20] proposed UHUI-apriori based on Apriori, and these are

Window	Batch	TID	Transaction
SW_1	B_1	T_1	$(a, 1, 0.81)\,(c, 1, 0.79)\,(d, 2, 0.93)$
SW_1	B_1	T_2	$(a, 2, 0.88)\,(c, 6, 0.92)\,(e, 2, 0.68)\,(f, 5, 0.83)$
SW_1, SW_2	B_2	T_3	$(a, 1, 0.83)\,(b, 2, 0.78)\,(c, 3, 0.91)\,(d, 3, 0.86)\,(e, 1, 0.76)$
SW_1, SW_2	B_2	T_4	$(b, 4, 0.72)\,(c, 3, 0.89)\,(d, 3, 0.86)\,(e, 2, 0.73)$
SW_1, SW_2	B_3	T_5	$(b, 2, 0.93)\,(c, 2, 0.87)\,(e, 1, 0.68)\,(f, 2, 0.63)$
SW_1, SW_2	B_3	T_6	$(a, 2, 0.93)\,(f, 5, 0.76)$
SW_2	B_4	T_7	$(a, 2, 0.81)\,(d, 1, 0.92)\,(f, 6, 0.78)$
SW_2	B_4	T_8	$(b, 4, 0.69)\,(c, 6, 0.83)\,(e, 3, 0.88)\,(f, 2, 0.63)$
		⋮	⋮

(a) Partial of an uncertain data stream

Item Name	a	b	c	d	e	f
External Utility	3	6	5	8	4	3

(b) Profit table

Figure 1: Partial of an uncertain data stream with its profit table.

only algorithms that used to solve HUIs mining problem over uncertain databases. However, the above algorithms can only handle static databases with uncertainty, and they could not deal with uncertain data stream.

In this paper, the concepts of HUIs mining over data stream and uncertain database to discover HUIs from uncertain data stream are combined. To the best of our knowledge, the proposed PHUIMUS algorithm is the first work that discovers HUIs over uncertain data stream.

3. New Definitions and Problem Statement

In this section, some definitions of HUIs mining are extended from the precise and static databases to the uncertain data streams. New definitions and problem statement related to HUIs mining over uncertain data stream are given.

3.1. New Definitions. Let $I = \{i_1, i_2, \ldots, i_L\}$ be a finite set of items; an uncertain data stream S is a continuous sequence of transactions $\{T_1, T_2, \ldots, T_n, \ldots\}$. A transaction $T_q = \{i_1, i_2, \ldots, i_l\}$ $(1 \leq l \leq L)$ in S contains a number of items, and each item i_p in T_q is associated with internal utility $\mathrm{iu}(i_p, T_q)$ and existential probability $p(i_p, T_q)$, which indicates quantity value of i_p in T_q and the likelihood of i_p being present in T_q, respectively [23]. In addition, external utility of item i_p in I is represented by $\mathrm{eu}(i_p)$.

In an uncertain data stream, data cannot be completely stored as infinite volume and storage structure are required in the dynamic adjustment for the purpose of reflecting the evolution of itemsets utility. So a sliding window is needed, which consists of m most recent batches, represented by $\mathrm{SW}_i = \{B_i, B_{i+1}, \ldots, B_{i+m-1}\}$. And a batch B_i consists of a certain number of continuously transactions in a time period; that is, $B_i = \{T_j, T_{j+1}, \ldots, T_{j+h-1}\}$. HUIs mining over uncertain data stream based on sliding windows is to mine PHUIs from every new window, which is formed once the oldest batch is removed from the window and the newest batch is inserted into the window.

For example, database in Figure 1 is partial of an uncertain data stream and its profit table, respectively. Assume that each batch includes two transactions and each sliding window includes three batches; there are four batches in the stream: $B_1 = \{T_1, T_2\}$, $B_2 = \{T_3, T_4\}$, $B_3 = \{T_5, T_6\}$, and $B_4 = \{T_7, T_8\}$, and the first three batches form the first sliding window: $\mathrm{SW}_1 = \{B_1, B_2, B_3\}$. When the fourth batch B_4 is filled up with transactions, the second sliding window is formed: $\mathrm{SW}_2 = \{B_2, B_3, B_4\}$.

What is different in PHUIs mining compared to traditional HUIs mining over data stream is that the items in the transactions have existential probability, which brings a change in the calculation of itemsets utility, transaction utility, and transaction weighted utilization. New definitions about them are presented below.

Definition 1 (potential utility of items). In a transaction T_q, potential utility of an item i_p, $\mathrm{pu}(i_p, T_q)$, is the product of its internal utility, existential probability, and external utility, denoted as

$$\mathrm{pu}\left(i_p, T_q\right) = \mathrm{iu}\left(i_p, T_q\right) \times p\left(i_p, T_q\right) \times \mathrm{eu}\left(i_p\right). \quad (1)$$

For example, in Figure 1, the potential utility of item $\{d\}$ in T_1 is $\mathrm{pu}(d, T_1) = \mathrm{iu}(d, T_1) \times p(i_p, T_1) \times \mathrm{eu}(d) = 2 \times 0.93 \times 8 = 14.88$.

Definition 2 (potential utility of itemsets in a batch). The potential utility of an itemset X in the batch B_i is defined as the sum of potential utilities of X in all transactions of B_i including X, denoted as

$$\mathrm{pu}_{B_i}(X) = \sum_{T_q \in B_i} \sum_{i_p \in X \subseteq T_q} \mathrm{pu}\left(i_p, T_q\right). \quad (2)$$

For example, in Figure 1, $\mathrm{pu}_{B_1}(ac) = \mathrm{pu}(ac, T_1) + \mathrm{pu}(ac, T_2) = 6.38 + 32.88 = 39.26$.

Definition 3 (potential utility of itemsets in a window). The potential utility of an itemset X in the window SW_k is defined as the sum of potential utilities of X in all batches of SW_k including X, denoted as

$$\mathrm{pu}_{\mathrm{SW}_k}(X) = \sum_{B_i \in \mathrm{SW}_k} \mathrm{pu}_{B_i}(X). \tag{3}$$

For example, in Figure 1, $\mathrm{pu}_{\mathrm{SW}_1}(ac) = \mathrm{pu}_{B_1}(ac) + \mathrm{pu}_{B_2}(ac) + \mathrm{pu}_{B_3}(ac) = 39.26 + 16.14 + 0 = 55.4$.

Definition 4 (potential utility of transactions). Potential utility of a transaction (PTU) T_q is defined as the sum of utilities of items in T_q, represented as

$$\mathrm{ptu}(T_q) = \sum_{i_p \in T_q} \mathrm{pu}(i_p, T_q). \tag{4}$$

For example, in Figure 1, $\mathrm{ptu}(T_1) = \mathrm{pu}(a, T_1) + \mathrm{pu}(c, T_1) + \mathrm{pu}(d, T_1) = 2.43 + 3.95 + 14.88 = 21.26$.

Definition 5 (see [20] (minimum potential utility value in a window)). Given the minimum utility threshold δ_{SW_k}, minimum potential utility value in the window SW_k, $\mathrm{minutil}_{\mathrm{SW}_k}$, is defined as

$$\mathrm{minutil}_{\mathrm{SW}_k} = \delta_{\mathrm{SW}_k} \times \sum_{T_q \in \mathrm{SW}_k} \mathrm{ptu}(T_q). \tag{5}$$

For example, in Figure 1, when set $\delta_{\mathrm{SW}_1} = 0.2$, $\mathrm{minutil}_{\mathrm{SW}_1} = 0.2 \times 221.66 = 44.332$.

Definition 6 (see [20] (PHUIs in a window)). An itemset X is a PHUI in the window SW_k, if $\mathrm{pu}_{\mathrm{SW}_k}(X) \geq \mathrm{minutil}_{\mathrm{SW}_k}$. Finding PHUIs in window SW_k means finding out all the itemsets X having criteria $\mathrm{pu}_{\mathrm{SW}_k}(X) \geq \mathrm{minutil}_{\mathrm{SW}_k}$.

For example, in Figure 1, when set $\delta_{\mathrm{SW}_1} = 0.2$, $\mathrm{minutil}_{\mathrm{SW}_1} = 44.332$, as $\mathrm{pu}_{\mathrm{SW}_1}(ac) = 55.4 > 44.332$, ac is a PHUI.

It can be seen that potential utility of itemsets has no down closure property [4], showing that the potential utility constraint is not monotone and antimonotone. Hence, unlike FIM, the potential utility of an itemset cannot be used to prune the search space. To deal with this problem, itemset's overestimate utility, transaction weighted probability and utility (TWPU), is used in the PHUIs mining process to prune the search space.

Definition 7 (TWPU in a batch). The TWPU of an itemset X in the batch B_i, $\mathrm{twpu}_{B_i}(X)$, is defined by

$$\mathrm{twpu}_{B_i}(X) = \sum_{X \subseteq T_q \in B_i} \mathrm{ptu}(T_q). \tag{6}$$

For example, in Figure 1, $\mathrm{twpu}_{B_1}(d) = \mathrm{ptu}(T_1) = 21.26$.

Definition 8 (TWPU in a window). The TWPU of an itemset X in the window SW_k, $\mathrm{twpu}_{\mathrm{SW}_k}(X)$, is defined by

$$\mathrm{twpu}_{\mathrm{SW}_k}(X) = \sum_{B_i \in \mathrm{SW}_k} \mathrm{twpu}_{B_i}(X). \tag{7}$$

For example, in Figure 1, $\mathrm{twpu}_{\mathrm{SW}_1}(d) = \mathrm{twpu}_{B_1}(d) + \mathrm{twpu}_{B_2}(d) + \mathrm{twpu}_{B_3}(d) = 21.26 + 106.29 + 0 = 127.55$.

Definition 9 (high transaction weighted probabilistic and utilization itemsets in a window). X is a high transaction weighted probabilistic and utilization itemset (HTWPUI) in the window SW_k, if $\mathrm{twpu}_{\mathrm{SW}_k}(X) \geq \mathrm{minutil}_{\mathrm{SW}_k}$.

For example, in Figure 1, when set $\delta_{\mathrm{SW}_1} = 0.2$, $\mathrm{minutil}_{\mathrm{SW}_1} = 44.332$, as $\mathrm{twpu}_{\mathrm{SW}_1}(d) = 127.55 > 44.332$, d is a HTWPUI.

Lemma 10. *The TWPU value of an itemset X in the window SW_k maintains the downward closure property.*

Proof. Let X be an itemset that is contained in SW_k and Y be a superset of itemset X. Then, if X is absent, Y cannot be presented in any transaction. So according to Definition 8, the TWPU value of Y is no larger than $\mathrm{twpu}_{\mathrm{SW}_2}(X)$, denoted as $\mathrm{twpu}_{\mathrm{SW}_2}(Y) \leq \mathrm{twpu}_{\mathrm{SW}_2}(X)$, and if $\mathrm{twpu}_{\mathrm{SW}_2}(X)$ is less than $\mathrm{minutil}_{\mathrm{SW}_2}$, Y cannot be a HTWPUI. □

Lemma 11. *For a window SW_k and a minimum utility threshold δ_{SW_k}, the set of PHUIs (S_{SW}) is a subset of the set of HTWPUIs (TS_{SW}).*

Proof. Let X be a PHUI in SW_k. According to Definitions 3 and 8, $\mathrm{pu}_{\mathrm{SW}_k}(X)$ must be less than or equal to $\mathrm{twpu}_{\mathrm{SW}_k}(X)$. So, if X is a PHUI, it must be a HTWPUI in SW_k, Furthermore, it can obtain that X is a member of the set TS_{SW}, and $S_{\mathrm{SW}} \subseteq TS_{\mathrm{SW}}$.

Since $\mathrm{twpu}_{\mathrm{SW}_k}(X)$ is an overestimate of $\mathrm{pu}_{\mathrm{SW}_k}(X)$, any PHUI will not be missed. But the true potential utility of the generated HTWPUIs may be lower than the minimum utility threshold. So in our algorithm, while finding HTWPUIs in SW_k by TWPUS-tree, we calculate PHUIs from them by PUS-list. □

3.2. Problem Statement. Given a continuous uncertain data stream, a predefined profit table and a user specified minimum utility threshold δ_{SW_k}, the problem of HUIs mining over uncertain data stream is to find the PHUIs whose potential utilities are no lower than $\mathrm{minutil}_{\mathrm{SW}_k}$.

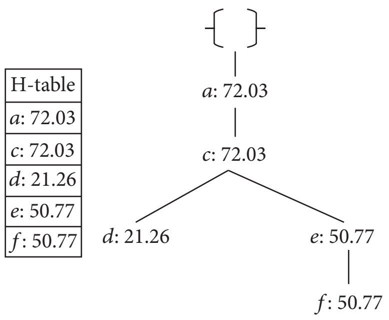
4. The Proposed Algorithm for Mining HUIs over Uncertain Data Stream

In this section, we first develop our proposed PUS-list and TWPUS-tree, respectively. Then, algorithm for HUIs mining over uncertain data stream based on sliding windows, PHUIs mining, is designed. Lastly, the proposed algorithm is thoroughly described and analyzed.

4.1. Construction of PUS-List and TWPUS-Tree over Uncertain Data Stream. The construction procedure of our proposed PUS-list and TWPUS-tree that are used to deal with the problem of HUIs mining over uncertain data stream is described. PUS-list is employed to calculate potential utility of candidates that are generated by mining TWPUS-tree without rescanning the analyzed uncertain data stream, which consists of potential utility of the items and transactions in the current window, respectively. Its number of rows is

TID	Item						
	a	b	c	d	e	f	ptu
SW1_$T1$	2.43	0	3.95	14.88	0	0	21.26
SW1_$T2$	5.28	0	27.6	0	5.44	12.45	50.77
SW1_$T3$	0	0	0	0	0	0	0
SW1_$T4$	0	0	0	0	0	0	0
SW1_$T5$	0	0	0	0	0	0	0
SW1_$T6$	0	0	0	0	0	0	0

(a) PUS-list of B_1



(b) TWPUS-tree of B_1

FIGURE 2: Construction of PUS-list and TWPUS-tree for B_1.

equal to the number of transactions in the current window, and its number of columns is equal to the number of items in utility table. The items in the TWPUS-tree with its header table are arranged in the lexicographic order. Item-id and TWPU value of the items are maintained in the header table to get HTWPUIs. Item-id and batch-by-batch TWPU information are maintained by each node in the TWPUS-tree to keep the window sliding environment. And adjacent links are also maintained in the tree structure to facilitate the tree traversals.

For the example of uncertain data stream in Figure 1, when B_1 arrives, sorting items in the lexicographic order, current window SW_1 is formed, and SW1_Ti represents the ith transaction in SW_1. Calculate potential utility of each item and transaction in B_1, respectively, when scanning the uncertain data stream, and potential PUS-list in Figure 2(a) can be obtained. Subsequently, as the first transaction T_1 has a PTU value of 21.26 and includes three items "a," "c," and "d," "a" is inserted into TWPUS-tree by creating a node with a TWPU value of 21.26, and items "c" and "d" are inserted with TWPU values of 21.26 later. At the same time, their TWPU values are also inserted into the header table in the lexicographic order. After inserting transactions T_1 and T_2 into TWPUS-tree, Figure 2(b) shows the TWPUS-tree constructed for B_1.

Similarly, B_2 and B_3 are inserted in PUS-list and TWPUS-tree, since SW_1 contains the first three batches; Figures 3(a) and 3(b) are the final PUS-list and TWPUS-tree for it, respectively.

When B_4 arrives, the information of B_4 needs to be inserted into PUS-list and TWPUS-tree, and the information of B_1 also needs to be deleted from PUS-list and TWPUS-tree. In detail, PUS-list needs to remove the transactions of B_1 from top of the list, insert transactions of B_4 from bottom of the list, and change SW1_Ti to SW2_Ti correspondingly. The TWPU counters of the nodes in TWPUS-tree are shifted one position left to remove the TWPU information of B_1, and the TWPU information of B_4 is inserted subsequently. Figures 4 and 5 indicate the deleting and inserting process, respectively. In Figure 4(b), as "a" contains information for B_1, B_2, and B_3, its new information is now $\{a\text{: } 49.18, 16.98, 0\}$. On the other hand, since its child "c" does not include any information of B_2 and B_3, it becomes $\{c\text{: } 0, 0, 0\}$ after shift operation and is deleted from the tree. Perform the same operations with the nodes in Figure 4(b). Subsequently, B_4 is inserted into the tree, and the result is shown in Figure 5.

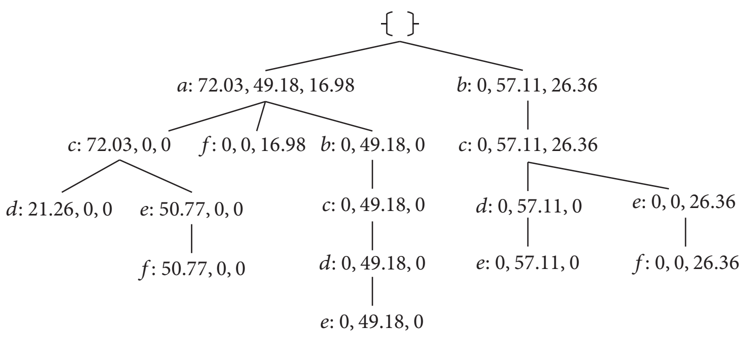
4.2. Mining Process of PHUIMS. This section deals with the mining procedure of our proposed PHUIMUS algorithm, which combines pattern growth approach with list based approach. In the proposed algorithm, a prefix tree is created from the bottommost item, where all the branches prefixing that item are taken with their TWPU values. For facilitation, all the TWPU values of each node in the prefix tree are added to one value. Subsequently, conditional tree is established based on the prefix tree, by removing those nodes with low TWPU value for that particular item. Lastly, potential utility of candidates that are generated by mining TWPUS-tree are calculated from current PUS-list, which can avoid rescanning the uncertain data stream.

For the example in Figure 1, mining the recent PHUIs means all the PHUIs in SW_2 must be found. Let $\delta_{SW_2} = 0.2$ and $\text{minutil}_{SW_2} = 231.69 \times 0.2 = 46.338$, the prefix tree of item "f," which is the bottom item, is shown in Figure 6(a). It demonstrates that items "a" and "d" cannot form any candidate itemsets with item "f" as their TWPU values are lower than minutil_{SW_2}. Hence, by deleting all the nodes that contain items "a" and "d" from the prefix tree of "f," the conditional tree of item "f" is constructed and shown in Figure 6(b). Candidate itemsets $\{b, c, e, f\text{: } 82.16\}$, $\{b, c, f\text{: } 82.16\}$, $\{b, e, f\text{: } 82.16\}$, $\{c, e, f\text{: } 82.16\}$, $\{b, f\text{: } 82.16\}$, $\{c, f\text{: } 82.16\}$, $\{e, f\text{: } 82.16\}$, and $\{f\text{: } 82.16\}$ are generated here.

Judge whether TWPU value of the other items is less than minutil_{SW_2}. If yes, any super itemsets of it/them cannot be candidate itemsets as well as PHUIs according to the downward closure property, so prefix/conditional tree for it/them need not be created. If no, generate candidate itemsets in the same way as item "f" for the items. All the candidate itemsets are added to global candidate list, and it is reset to NULL when the sliding window changed. Exact potential utility in SW_2 for candidate itemsets is calculated from PUS-list in Figure 5(b) directly. For example, for the candidate

TID	Item *a*	*b*	*c*	*d*	*e*	*f*	ptu
SW1_*T*1	2.43	0	3.95	14.88	0	0	21.26
SW1_*T*2	5.28	0	27.6	0	5.44	12.45	50.77
SW1_*T*3	2.49	9.36	13.65	20.64	3.04	0	49.18
SW1_*T*4	0	17.28	13.35	20.64	5.84	0	57.11
SW1_*T*5	0	11.16	8.7	0	2.72	3.78	26.36
SW1_*T*6	5.58	0	0	0	0	11.4	16.98

(a) PUS-list of SW_1



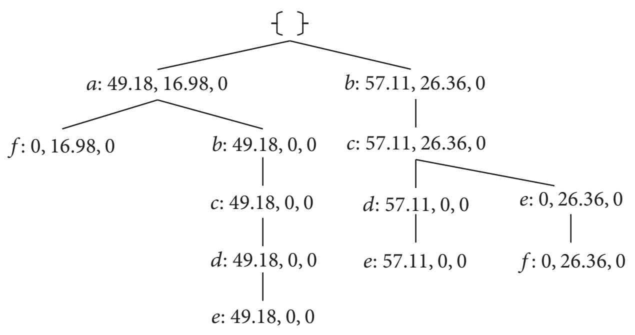
(b) TWPUS-tree of SW_1

Figure 3: Construction of PUS-list and TWPUS-tree for SW_1.

TID	Item *a*	*b*	*c*	*d*	*e*	*f*	ptu
SW2_*T*1	2.49	9.36	13.65	20.64	3.04	0	49.18
SW2_*T*2	0	17.28	13.35	20.64	5.84	0	57.11
SW2_*T*3	0	11.16	8.7	0	2.72	3.78	26.36
SW2_*T*4	5.58	0	0	0	0	11.4	16.98
SW2_*T*5	0	0	0	0	0	0	0
SW2_*T*6	0	0	0	0	0	0	0

(a) PUS-list of deleting B_1

H-table
a: 66.16
b: 132.65
c: 132.65
d: 106.29
e: 132.65
f: 43.34



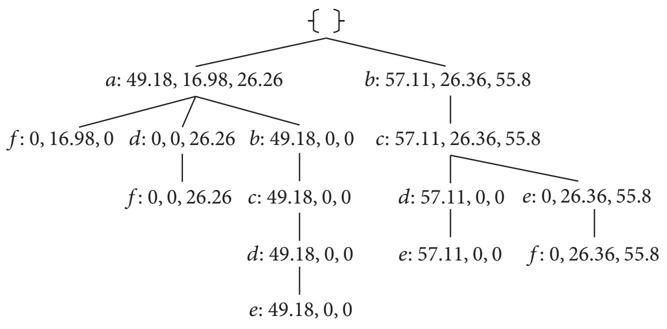
(b) TWPUS-tree of deleting B_1

Figure 4: Construction of PUS-list and TWPUS-tree for deleting B_1.

TID	Item *a*	*b*	*c*	*d*	*e*	*f*	ptu
SW2_*T*1	2.49	9.36	13.65	20.64	3.04	0	49.18
SW2_*T*2	0	17.28	13.35	20.64	5.84	0	57.11
SW2_*T*3	0	11.16	8.7	0	2.72	3.78	26.36
SW2_*T*4	5.58	0	0	0	0	11.4	16.98
SW2_*T*5	4.86	0	0	7.36	0	14.04	26.26
SW2_*T*6	0	16.56	24.9	0	10.56	3.78	55.8

(a) PUS-list of inserting B_4

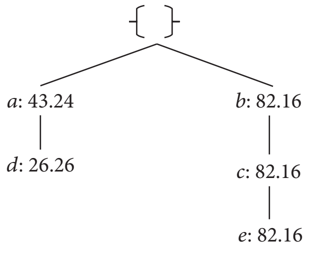
H-table
a: 92.42
b: 188.45
c: 188.45
d: 132.55
e: 188.45
f: 125.4



(b) TWPUS-tree of inserting B_4

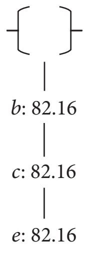
Figure 5: Construction of PUS-list and TWPUS-tree for inserting B_4.

H-table
a: 43.24
b: 82.16
c: 82.16
d: 26.26
e: 82.16



(a) Prefix tree for item "*f*"

H-table
b: 82.16
c: 82.16
e: 82.16



(b) Conditional tree for item "*f*"

Figure 6: Mining process.

itemset $\{b, c, f\colon 82.16\}$, it exists in SW2_$T3$ and SW2_$T6$, so we can calculate that $\mathrm{pu}_{\mathrm{SW}_2}(b,f) = (\mathrm{pu}(b,T_3) + \mathrm{pu}(b,T_6)) + (\mathrm{pu}(c,T_3) + \mathrm{pu}(c,T_6)) + (\mathrm{pu}(f,T_3) + \mathrm{pu}(f,T_6)) = (11.16 + 16.56) + (8.7 + 24.9) + (3.78 + 3.78) = 68.88$ from the PUS-list in Figure 5(b) without rescanning the analyzed uncertain data stream. And as $\mathrm{pu}_{\mathrm{SW}_2}(b,c,f) > \mathrm{minutil}_{\mathrm{SW}_2}$, $\{b,c,f\}$ is a PHUI. Perform the same calculation process for the other candidate itemsets in the global candidate list; then PHUIs for the current window SW_2 can be obtained.

4.3. Algorithm Description and Analysis. In this section, PHUIMUS algorithm is described and analyzed. At first, the description of PHUIMUS algorithm is shown in Algorithm 1.

In Algorithm 1, Steps 1 to 3 are used to initialize a global header table H that is used to keep all the items in the lexicographic order, TWPUS-tree that is initialized as NULL, and a global PUS-list that can keep potential utility of items and transactions are created, respectively. Step 3 to Step 6 are TWPUS-tree's construction process. When a new batch B_j arrives, Step 4 sorts items of transaction T_k in batch B_j and Step 5 calculates potential utility of items and transactions and inserts them into PUS-list; Step 6 updates header table H. If current batch number j is no more than M, B_j must be included in the first sliding window SW_1, Step 7 only changes the jth position of TWPU counter arrays, whose size is M, no matter whether the items exist in the TWPUS-tree before. On the contrary, when current batch number j is larger than M, Step 8 first performs one time left shift operation for all the TWPU counter arrays to remove the oldest batch. Then, remove the transactions of B_{j-M+1} and insert transactions of B_j in the PUS-list. Subsequently, it updates header table H and deletes the nodes that all the values in their corresponding TWPU counter arrays are zero. Lastly, Step 8 changes the rightest position of TWPU counter arrays and keeps the size of the arrays as M, no matter whether the items exist in the TWPUS-tree before. Step 9 to Step 15 are the mining process of PHUIs from the current window SW_i. From each bottom item β of H, prefix tree PT_β with its header table HT_β is created. According to user specified δ_{SW_i}, the items that TWPU values are less than $\mathrm{minutil}_{\mathrm{SW}_i}$ are deleted from PT_β and HT_β, and conditional tree CT_β and its header table HC_β are created. Subsequently, mine all the candidate itemsets CT_β_list from CT_β and add $\{\mathrm{CT}_\beta\text{\_list}, \beta\}$ in the global candidate list. Furthermore, calculate PHUIs from the global candidate list by PUS-list. Finally, delete current bottom item β of H, and when it becomes NULL, jump out current loop.

What is more, it is not difficult to find that following properties are satisfied by our proposed algorithm.

Lemma 12. *The number of candidate itemsets generated by the PHUIMUS algorithm (N_1) is no larger than that of the level-wise based algorithms (N_2), denoted as $N_1 \le N_2$.*

Proof. When all the subsets of an itemset X are candidate itemsets (HTWPUIs), it becomes a candidate itemset in the existing level-wise based algorithms. Therefore, X may have low TWPU value that cannot be a candidate or does not appear in the current window. In PHUIMUS algorithm, if X does not appear in the current window, it cannot appear in any branch of the tree and becomes a candidate. What is more, after determining X is not a HTWPUI, it is pruned. So the candidate set of PHUIMUS contains only the true HTWPUIs, and N_1 cannot be larger than N_2. □

Lemma 13. *The PHUIMUS algorithm can find out PHUIs without rescanning the analyzed uncertain data stream.*

Proof. As the proposed PHUIMUS algorithm combines pattern growth approach with list based approach, exact potential utility can be calculated by PUS-list from the global candidate itemsets that are generated by TWPUS-tree directly, without rescanning the analyzed uncertain data stream. □

Table 1: Parameters of used databases.

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

5. Experimental Results

In this section, four experiments are used to evaluate the performance of our proposed algorithm over uncertain data stream in terms of run-time, memory consumption, number of discovered PHUIs, and scalability. Because it is considered to be the first work HUIs mining over uncertain data stream and MHUI-TID is an outstanding algorithm for mining HUIs from data streams, the performance of the designed PHUIMUS algorithm is only compared with MHUI-TID. And the comparison between PHUIs and HUIs is made to evaluate whether the proposed algorithm is acceptable.

The overall algorithms are carried out in Matlab, with experiments in a PC with Intel Core's i5-4590 dual core processor, 4 GB RAM, and 32-bit windows operation system of the Microsoft company. Experimental results and discussions are followed.

5.1. Datasets. Experiments were performed on synthetic database T10I4D100K and real-life databases mushroom, connect, and accidents, which are widely used in the issue of HUIs mining. Parameters and characteristics of these databases are, respectively, shown in Tables 1 and 2.

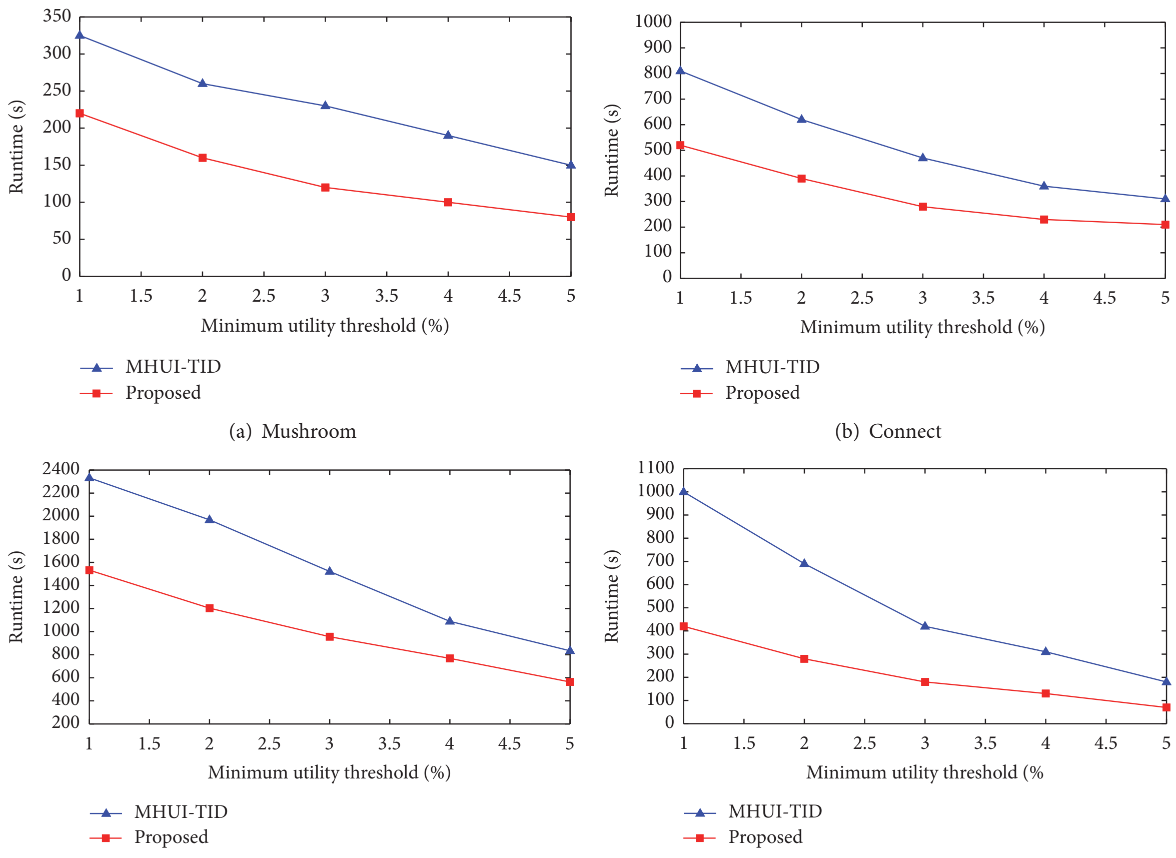
As these databases do not provide external utility, internal utility, and existential probability of each item, a simulation model [7] is employed. The model generates random numbers that obey log-normal distribution in the $[1, 5]$ interval and $[1, 1000]$ interval, which correspond to internal and external utility, respectively. In addition, due to uncertainty property of items in each transaction, their existential probability obeys uniform distribution in the $[0.5, 1]$ interval. What is more, as the study object of our proposed algorithm is uncertain data stream, we divide these databases into some windows containing a fixed number of batches, and the Batchsize (N) and Winsize (M) of each database are also shown in Table 2.

```
Input: A transaction uncertain data stream, utility table, $\delta_{\mathrm{SW}_i}$ for current window $\mathrm{SW}_i$,
       number of batches in a window ($M$), number of transactions in a batch ($N$).
Output: PHUIs for the current window $\mathrm{SW}_i$
Step 1. Create a global table H to keep the items in the lexicographic order;
Step 2. Create the root of TWPUS-tree and initialize it as NULL;
Step 3. Create a global PUS-list to keep potential utility of items and transactions, whose number of row is
        $M \times N$, and number of column is equal to the number of items in utility table;
        while a new batch $B_j$ arrives do
          for transaction $T_k$ in batch $B_j$ do
Step 4.     Sort the items of $T_k$ in the lexicographic order;
Step 5.     Calculate potential utility of items and transactions, inserting them into PUS-list;
Step 6.     Update twpu value in the header table H;
Step 7.     If $j \le M$ then
              If the item $\alpha$ in $T_k$ isn't exist in the TWPUS-tree then
                Create a new node for it, which consists by the item's name and twpu counter array
                  $(0_{B_1}, \ldots, \mathrm{twpu}_{B_j}(\alpha), \ldots, 0_{B_M})$;
              else
                Insert twpu value of $\alpha$ to its twpu counter array's $j$th position, denoted as
                  $(\mathrm{twpu}_{B_1}(\alpha), \ldots, \mathrm{twpu}_{B_j}(\alpha), \ldots, 0_{B_M})$;
              End if
Step 8.     else
              Perform one time left shift operation for all the twpu counter arrays;
              Remove the transactions of $B_{j-M+1}$ and insert transactions of $B_j$ in the PUS-list;
              Update them in the header table H;
              Delete the nodes that all the values in their corresponding twpu counter array are zero;
              If the item in $T_k$ isn't exist in the TWPUS-tree then
                Create a new node for it, which consists by the item's name and twpu counter array
                  $(0_{B_{j-M+1}}, \ldots, 0, \ldots, \mathrm{twpu}_{B_j})$;
              else
                Insert twpu value of $\alpha$ to its twpu counter array's rightest position, denoted as
                  $(\mathrm{twpu}_{B_{j-M+1}}(\alpha), \ldots, \mathrm{twpu}_{B_j}(\alpha))$;
              End if
            End if
          End for
          for each window $\mathrm{SW}_i$ do
            for each item $\beta$ from the bottom of H do
Step 9.       Add itemset $\beta$ to the global candidate list;
Step 10.      Create Prefix tree $\mathrm{PT}_\beta$ with its header table $\mathrm{HT}_\beta$ for item $\beta$;
Step 11.      For each item $\gamma$ of $\mathrm{HT}_\beta$ do
                If $\mathrm{twpu}_{\mathrm{SW}_i}(\gamma) < \mathrm{minutil}_{\mathrm{SW}_i}$ then
                  Delete $\gamma$ from $\mathrm{PT}_\beta$ and $\mathrm{HT}_\beta$ to create conditional tree $\mathrm{CT}_\beta$ and its header table $\mathrm{HC}_\beta$;
                End if
              End for
Step 12.      Create all the candidate itemsets from $\mathrm{CT}_\beta$, represented by $\mathrm{CT}_\beta$_list;
Step 13.      Add {$\mathrm{CT}_\beta$_list, $\beta$} to the global candidate list;
Step 14.      Calculate PHUIs from the global candidate list by PUS-list;
Step 15.      Delete current bottom item of H, when it becomes NULL, jump out current loop;
            End for
          End for
        End while
```

Algorithm 1: Description of PHUIMUS algorithm.

TABLE 2: Characteristics of the databases.

Dataset	# $\|D\|$	# $\|I\|$	AvgLen	Type	Batchsize (N)	Winsize (M)
T10I4D100K	100000	870	10.1	Sparse	10000	2
Mushroom	8124	119	7.2	Dense	500	4
Connect	67557	132	43	Dense	5000	3
Accidents	340183	468	33.8	Dense	15000	4



(a) Mushroom

(b) Connect

(c) Accidents

(d) T10I4D100K

FIGURE 7: Run-time of the compared algorithms for various δ.

5.2. Run-Time. Set δ_{SW_k} is equal in all the windows, abbreviated as δ, run-time of the proposed algorithm is compared with that of MHUI-TID on the above four databases for various δ, and the result is shown in Figure 7. Notice that the databases processed by MHUI-TID do not contain probability values, so only precise versions of the four databases are used by MHUI-TID.

It is indicated in Figure 7 that the algorithm is superior to MHUI-TID. This result is reasonable since the proposed algorithm discovers PHUIs from the TWPUS-tree and PUS-list directly, which can effectively avoid consuming time on database scans. What is more, this also indicates that the combination of pattern growth approach and list based approach has a good performance on dealing with the problem of HUIs mining over uncertain data stream.

5.3. Number of Discovered PHUIs. In an uncertain data stream, when the existential probability of items is set to 1, it degraded for an accurate data stream. For this case, PHUIs mining algorithms also get the whole set of HUIs, which is the same as the result of traditional HUIs mining algorithms. As no algorithm had been developed for discovering HUIs over uncertain data stream previously, the result of the designed algorithm is compared to that of MHUI-TID algorithm by ignoring probability values of uncertain data stream. This comparison is made between PHUIs and HUIs, which is employed to evaluate whether the proposed algorithm can be accepted. The number of discovered HUIs and PHUIs under various δ is shown in Figure 8.

From Figure 8, for various δ on four databases, it is presented that the number of PHUIs is usually no larger

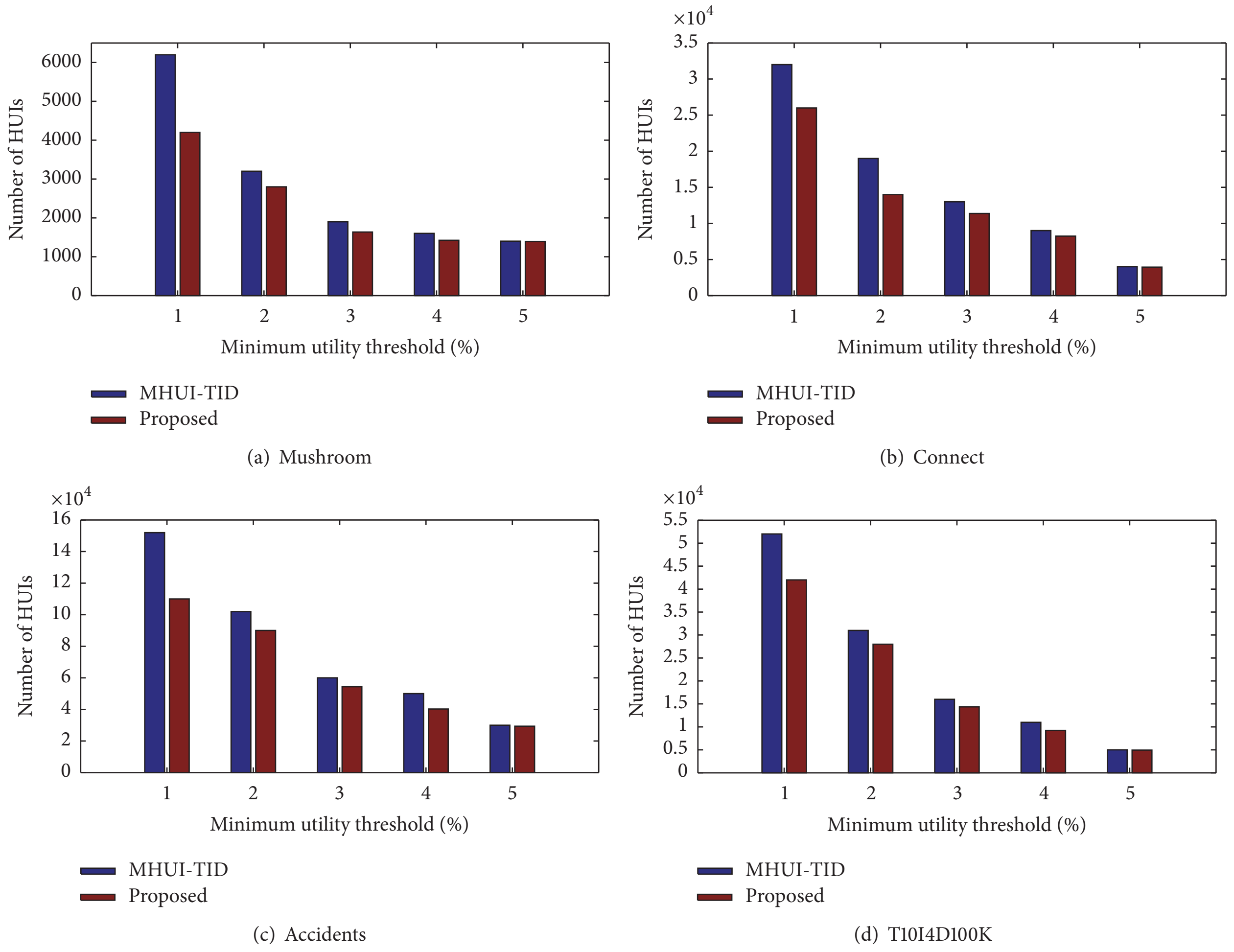


Figure 8: Number of HUIs and PHUIs under various δ.

compared with that of HUIs. Besides, both the numbers of HUIs and PHUIs are inversely proportional to δ. This is because that the proposed algorithm considers both the probability and the utility, and MHUI-TID only considers the utility. This result also shows that few PHUIs are produced from numerous discovered HUIs when considering the probability constraint. So, in real-life applications, lots of HUIs may not be the itemsets needed by users for making efficient decisions, especially when the δ is set high. Therefore, PHUIs are more valuable and fewer compared to HUIs as PHUIs have distinct probability values.

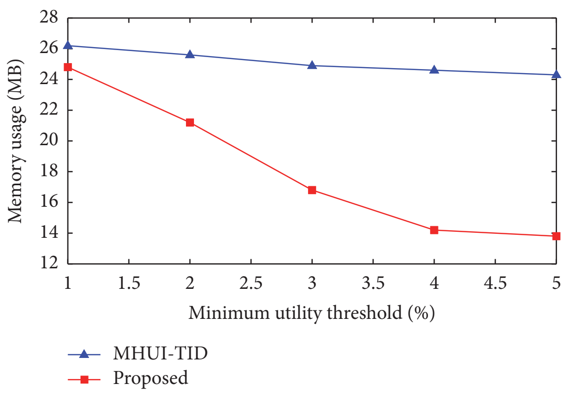
5.4. Memory Consumption. The peak memory consumption of the proposed algorithm and MHUI-TID algorithm is compared, and the results under various δ are shown in Figure 9.

From Figure 9, in various δ for the four databases, the proposed algorithm has a slightly good performance on memory consumption compared with MHUI-TID algorithm. This result is reasonable since the proposed PHUIMUS algorithm discovers PHUIs by taking both the probability and utility constraints into consideration through the designed TWPUS-tree and PUS-list, so more efficient pruning strategies can be applied in them to improve its performance. As a result, the memory consumption of the PHUIMUS algorithm is somehow a little better than MHUI-TID algorithm.

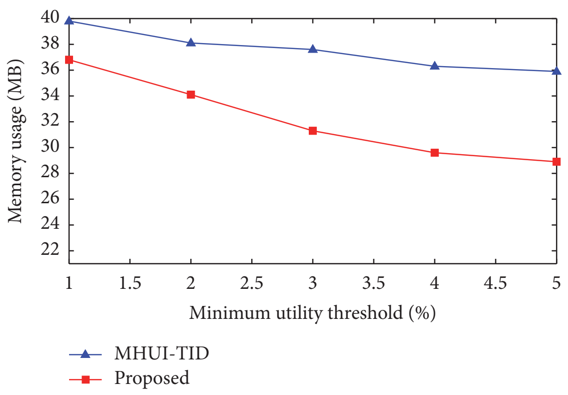
5.5. Efficiency of PHUIMUS with Window Size Variation. In terms of run-time and memory consuming, stream data mining algorithms based on sliding windows are greatly influenced by window sizes. As usual, window size depends on the number of transactions and batches in a window. Therefore, given a certain δ, by varying both of these two parameters, it compares the run-time of the algorithm and that of the existing MHUI-TID algorithm, as shown in Figure 10.

When the window size changes, it is presented in Figure 10 that our algorithm is better than the existing one. Particularly when the window size is bigger or the number of distinct items increases, the efficiency of the proposed algorithm is more prominent.

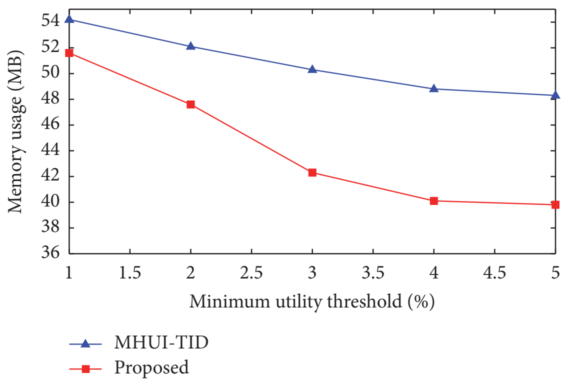
What is more, Figure 11 shows memory consumption of the proposed algorithm and MHUI-TID algorithm under various sizes of windows.



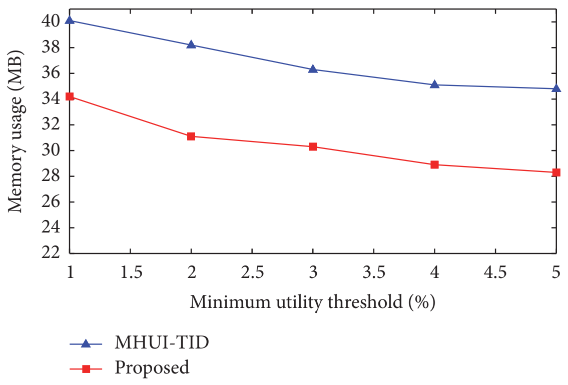
(a) Mushroom



(b) Connect

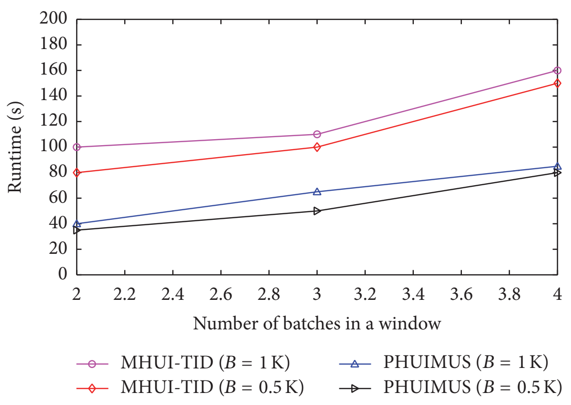


(c) Accidents

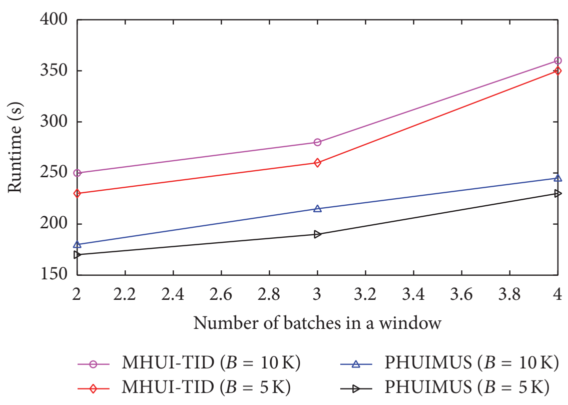


(d) T10I4D100K

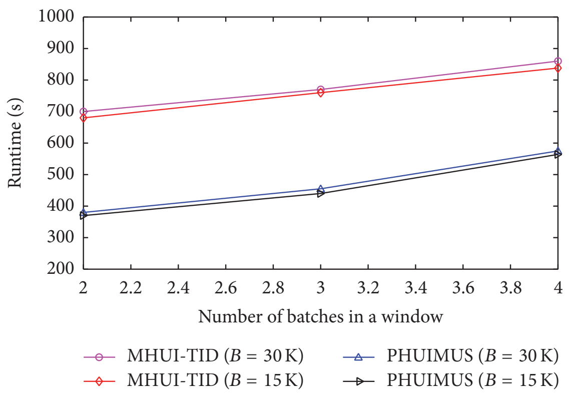
FIGURE 9: Memory consumption under various δ.



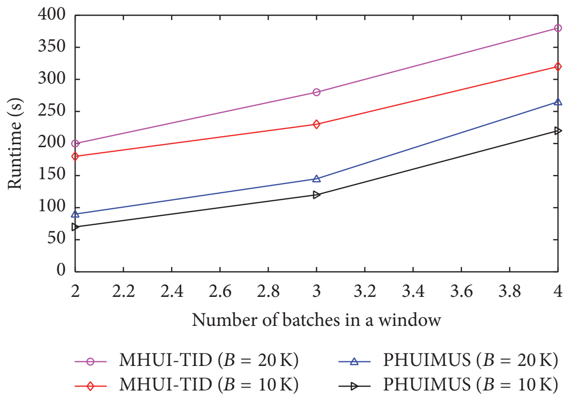
(a) Mushroom 5%



(b) Connect 5%



(c) Accidents 5%



(d) T10I4D100K 5%

FIGURE 10: Effect of window size variation on run-time.

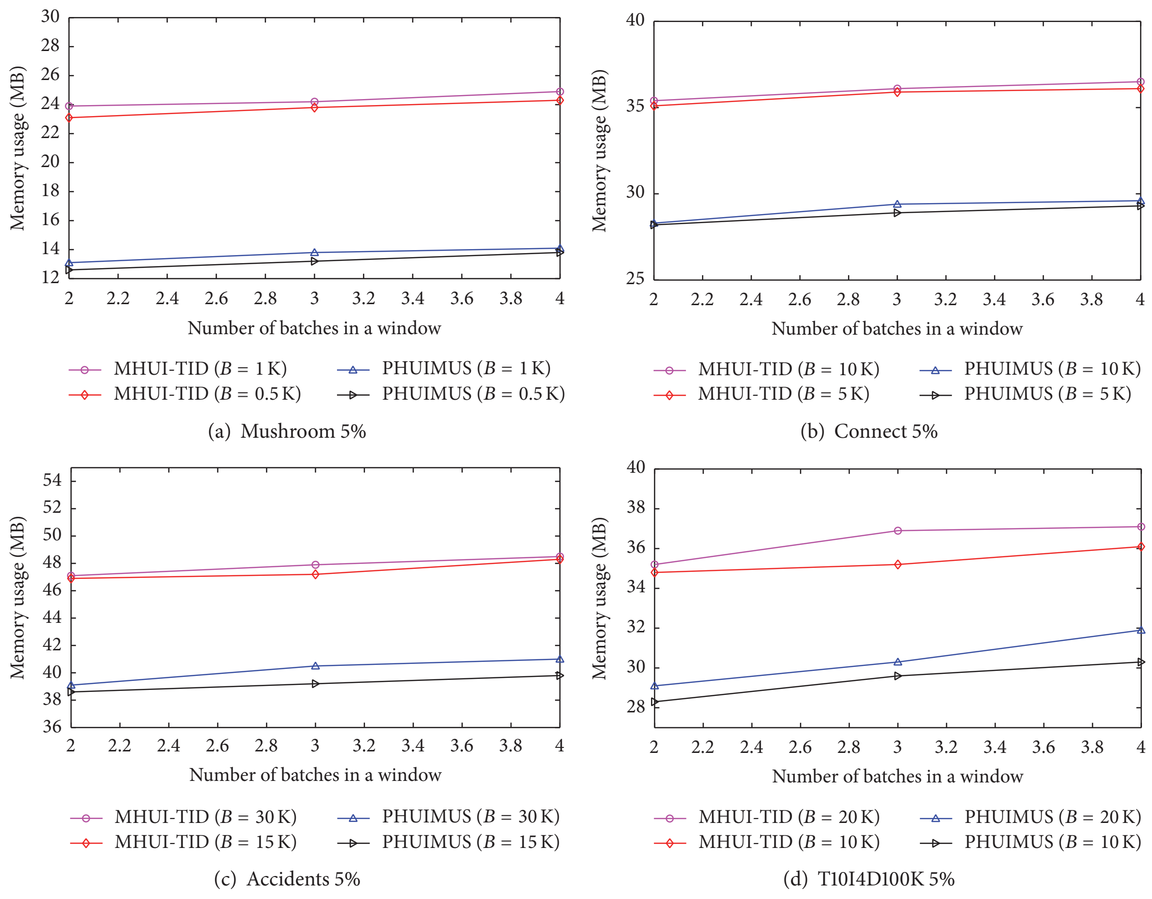


Figure 11: Effect of window size variation on memory consumption.

From Figure 11, it is presented that our proposed algorithm exceeds the existing MHUI-TID algorithm in terms of memory consuming under different window sizes. The main reason for this result is that our proposed TWPUS-tree can represent all the useful information in a compressed form. More importantly, our algorithm can be effective without rescanning the analyzed uncertain data stream with PUS-list to discover PHUIs.

6. Conclusions

In the precise data stream, several algorithms have been proposed to mine HUIs, such as THUI-Mine, MHUI-BT, and MHUI-TID. Moreover, some extended areas of HUIs mining, such as maximal high utility itemsets mining, interactive mining, and top-k high utility itemsets mining, have been studied recently.

In the uncertain databases, itemsets with high utility and high existential probability are useful to users, not itemsets with only one of them. To the best of our knowledge, Lin et al. proposed PHUI-UP based on two-phase model and PHUI-List based on list structure, Lan et al. proposed UHUI-apriori based on Apriori, and these are only algorithms that used to solve HUIs mining problem over uncertain databases.

So according to the above researches, this paper provides an efficient method for HUIs mining over uncertain data stream. New definitions of items utility, itemsets utility, transaction utility, and transaction weighted utility are given. A novel tree structure, TWPUS-tree, list structure, PUS-list, and a new algorithm PHUIMUS are proposed. In detail, TWPUS-tree can maintain a fixed sort order and batch-by-batch information, which is easy to construct and maintain with a sliding window. PUS-list can get exact potential utility of candidate itemsets generated by TWPUS-tree without rescanning the analyzed uncertain data stream. By using TWPUS-tree and PUS-list, PHUIMUS algorithm can capture the recent change of information in an uncertain data stream adaptively. Experiments results show that our algorithm outperforms the existing algorithm in run-time, number of discovered PHUIs, memory usage, and scalability.

To the best of my knowledge, this is the first algorithm about finding HUIs over uncertain data stream. By combining a pattern growth approach with a list based approach, the proposed algorithm can significantly reduce the number of candidate itemsets as well as the overall run-time. What is more, by keeping the recent information very efficiently in the TWPUS-tree and PUS-list, the algorithm also saves a lot of memory space. More works can be done in improving efficiency of discovering HUIs over uncertain data stream in the near future.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] R. Agrawal, T. Imielinski, and A. Swami, "Database mining: a performance perspective," *IEEE Transactions on Knowledge and Data Engineering*, vol. 5, no. 6, pp. 914–925, 1993.

[2] J. Han, J. Pei, Y. Yin, and R. Mao, "Mining frequent patterns without candidate generation: a frequent-pattern tree approach," *Data Mining and Knowledge Discovery*, vol. 8, no. 1, pp. 53–87, 2004.

[3] R. Agrawal, T. Imielinski, and A. Swami, "Mining association rules between sets of items in large database," in *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pp. 207–216, Washington, DC, USA, 1993.

[4] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules," in *Proceedings of the 20th International Conference on Very Large Data Bases*, pp. 487–499, Santiago, Chile, 1994.

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

[6] R. Chan, Q. Yang, and Y.-D. Shen, "Mining high utility itemsets," in *Proceedings of the 3rd IEEE International Conference on Data Mining (ICDM '03)*, pp. 19–26, Melbourne, Fla, USA, November 2003.

[7] Y. Liu, W. K. Liao, and 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*, pp. 689–695, Hanoi, Vietnam, May 2005.

[8] C. F. Ahmed, S. K. Tanbeer, B.-S. Jeong, and Y.-K. Lee, "Efficient tree structures for high utility pattern mining in incremental databases," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 12, pp. 1708–1721, 2009.

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

[10] V. S. Tseng, B.-E. Shie, C.-W. Wu, and P. S. Yu, "Efficient algorithms for mining high utility itemsets from transactional databases," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 8, pp. 1772–1786, 2013.

[11] M. Liu and J. Qu, "Mining high utility itemsets without candidate generation," in *Proceedings of the 21st ACM International Conference on Information and Knowledge Management (CIKM 12)*, pp. 55–64, November 2012.

[12] J. C.-W. Lin, W. Gan, T.-P. Hong, and V. S. Tseng, "Efficient algorithms for mining up-to-date high-utility patterns," *Advanced Engineering Informatics*, vol. 29, no. 3, pp. 648–661, 2015.

[13] V. S. Tseng, C. J. Chu, and T. Liang, "Efficient mining of temporal high utility itemsets from data streams," in *Proceedings of the ACM KDD Workshop on Utility-Based Data Mining Workshop*, pp. 18–27, Chicago, Ill, USA, 2006.

[14] H.-F. Li, H.-Y. Huang, Y.-C. Chen, Y.-J. Liu, and S.-Y. Lee, "Fast and memory efficient mining of high utility itemsets in data streams," in *Proceedings of the 8th IEEE International Conference on Data Mining (ICDM '08)*, pp. 881–886, IEEE, Pisa, Italy, December 2008.

[15] H.-F. Li, H.-Y. Huang, and S.-Y. Lee, "Fast and memory efficient mining of high-utility itemsets from data streams: with and without negative item profits," *Knowledge and Information Systems*, vol. 28, no. 3, pp. 495–522, 2011.

[16] B.-E. Shie, P. S. Yu, and V. S. Tseng, "Efficient algorithms for mining maximal high utility itemsets from data streams with different models," *Expert Systems with Applications*, vol. 39, no. 17, pp. 12947–12960, 2012.

[17] C. F. Ahmed, S. K. Tanbeer, B.-S. Jeong, and H.-J. Choi, "Interactive mining of high utility patterns over data streams," *Expert Systems with Applications*, vol. 39, no. 15, pp. 11979–11991, 2012.

[18] M. Zihayat and A. An, "Mining top-k high utility patterns over data streams," *Information Sciences*, vol. 285, pp. 138–161, 2014.

[19] 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," *Knowledge-Based Systems*, vol. 96, pp. 171–187, 2016.

[20] Y. Q. Lan, Y. Wang, Y. Wang, S. W. Yi, and D. Yu, "Mining high utility itemsets over uncertain databases," in *Proceedings of the 7th International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC '15)*, pp. 235–238, IEEE, Xi'an, China, September 2015.

[21] S. Kannimuthu and K. Premalatha, "Discovery of high utility itemsets using genetic algorithm with ranked mutation," *Applied Artificial Intelligence*, vol. 28, no. 4, pp. 337–359, 2014.

[22] J. C.-W. Lin, L. Yang, P. Fournier-Viger, T.-P. Hong, and M. Voznak, "A binary PSO approach to mine high-utility itemsets," *Soft Computing*, vol. 3, pp. 1107–1135, 2016.

[23] C. K. Chui, B. Kao, and E. Hung, "Mining frequent itemsets from uncertain data," in *Proceedings of the Pacific-Asia Conference Advances in Knowledge Discovery and Data Mining*, pp. 47–58, Nanjing, China, 2007.