Intelligent Data Analysis 23 (2019) 1219–1241
DOI 10.3233/IDA-184255
IOS Press

Probabilistic maximal frequent itemset mining methods over uncertain databases

Haifeng Li, Mo Hai*, Ning Zhang, Jianming Zhu, Yue Wang and Huaihu Cao
School of Information, Central University of Finance and Economics, Beijing 100081, China

Abstract. Uncertain data are data accompanied with probability, which makes frequent itemset mining more challenging. This paper focuses on the problem of mining probabilistic maximal frequent itemsets. We redefine the concept of probabilistic maximal frequent itemset to be consistent with the traditional definition and provide a better view on how to devise pruning strategies. A tree-based index called the probabilistic maximal frequent itemset tree is constructed to maintain the probabilistic frequent itemsets. We proposed a depth-first probabilistic maximal frequent itemset mining algorithm to bottom-up generate the exact results, in which support and expected support are used to estimate the range of probabilistic support, enabling the frequency of an itemset to be inferred with less runtime and memory usage. Also, superset pruning is employed to further reduce mining cost. Nevertheless, certain probabilistic supports have to be computed when the minimum support is low, which may result in highly increased mining speed. This problem is addressed in our approximate probabilistic maximal frequent itemset mining method, which uses the expected support to directly compute the probabilistic support. Theoretical analysis and experimental studies demonstrate that our proposed algorithms have high accuracy, expend less computational time and use less memory, and significantly outperform the *TODIS-MAX* [20] state-of-the-art algorithm.

Keywords: Uncertain database, probabilistic frequent itemset, data mining, probabilistic maximal frequent itemset

1. Introduction

Frequent itemset mining, one of the traditional and important fields in data mining, entails discovering itemsets whose occurrences are larger than a specified threshold from massive data. Many efficient algorithms and methods have been developed in recent years [1]. In such methods, a very important assumption is, however, that the mined transactions are exact, regardless of whether they are static or increasingly updated.

The fact is that, uncertainty often exists in the real world. In a traffic system, a wireless sensor network may miss some high-speed vehicles and generate incomplete data. In a Global Position System (*GPS*), we may locate the positions over only a fuzzy area because of privacy concerns. For two information systems, the same property may have various structures, which will result in errors when we integrate the data. As a result, a probability is given to each value to help users make a better decision. This new feature brings new challenges that cannot be well addressed by employing traditional frequent itemset mining methods.

*Corresponding author: Mo Hai, School of Information, Central University of Finance and Economics, Beijing 100081, China. Tel.: +86 13691154237; E-mail: haimo_hm@163.com.

1.1. Motivation

When mining frequent itemsets over exact databases, it has been presented the frequent itemsets have redundant information. Actually, there have been many itemset compression methods proposed, including the maximal itemset [3], the closed itemset [4] and the non-derivable itemset [5]. If the user does not care about the support of an itemset, but only want to know whether it is frequent, the maximal frequent itemset is the best choice because it is the most efficient method to represent the frequent itemsets. Similarly, if we want to discover frequent itemsets over uncertain databases to obtain the maximal frequent itemsets, not only can we make the mining results easier to use, but we can also reduce the computational cost and memory size. Therefore, in this paper, we investigate how to efficiently discover the maximal frequent itemsets over uncertain databases.

1.2. Challenges and contributions

To address our proposed problem, an intuitive consideration is to enumerate all the probabilistic frequent itemsets and then to filter the probabilistic maximal frequent itemsets. This method, named *pApriori*, was proposed in [20], in which some optimized techniques were introduced. First, it used a divide-and-conquer method to compute the probability of an itemset in $O(nlog^2(n))$ time complexity and $O(n)$ space complexity. Second, the dataset is represented in a vertical manner. Finally, it used the expected support to estimate the frequency of itemsets, which can reduce computational cost when generating infrequent itemsets. It was a costly solution though because it cannot avoid the probability mass function computation of all the frequent itemsets; thus, a new method called *TODIS-MAX* [20] was proposed to improve performance. Besides all the techniques used in *pApriori*, *TODIS-MAX* also presented its own optimizations. It used a top-down method, which generated the itemsets from supersets to subsets, and thus it can efficiently use a pruning strategy when generating the infrequent itemsets. Also, it utilized a method to derive the probability mass function of the subsets from the supersets, which can further reduce the computational cost. To the best of our knowledge, *TODIS-MAX* is the most efficient algorithm that can be used to achieve the probabilistic maximal frequent itemsets.

However, there are some problems to consider in mining: 1. The top-down method may reach a bottleneck when the number of probabilistic frequent 1-items increases, which will result in an exponentially increasing number of probabilistic infrequent itemsets, and this is difficult to handle even if the time complexity is reduced to $O(n)$. 2. The value to decide the probabilistic infrequent itemsets is not tight enough, which means that multilevel itemsets must be recomputed with the divide-and-conquer method. 3. If an itemset is frequent, the computational cost is high because the time complexity is $O(nlog^2 n)$ on the worst condition. Accordingly, the following new problems we plan to address are posed: How do we reduce the number of most exponentially increased itemsets? How do we further decrease the computational cost of the frequent itemsets? Finally, how do we efficiently achieve the maximal frequent itemsets?

In this paper, we focus on these problems and make the following contributions.

1. We focus on the problem of maximal frequent itemset mining over uncertain data, and we redefine the probabilistic maximal frequent itemset to be in line with the traditional definition, and supply a better pruning method.
2. We introduce a compact probabilistic maximal frequent itemset tree data structure to maintain the information of probabilistic frequent itemsets, which in a bottom-up manner, can efficiently organize the mining results for the itemset search.

3. We propose a probabilistic maximal frequent itemset mining algorithm to depth-first discover the probabilistic maximal frequent itemsets. We use pruning strategies to reduce the mining cost. We also propose an approximate probabilistic maximal frequent itemset mining algorithm to more efficiently obtain the probabilistic maximal frequent itemsets with a slight loss of accuracy.
4. We compare our algorithms with the *TODIS-MAX* algorithm [20] on three synthetic datasets and four real-life datasets. Our experimental results demonstrate that our algorithms are more effective and efficient.

The rest of this paper is organized as follows: In Section 2 we summarize the related works. Section 3 presents the preliminaries and then define the problem. Section 4 introduces the data structures, and illustrates our algorithm in detail, Section 5 describes the approximate method. Section 6 evaluates the performance with a theoretical analysis and experimental results. Finally, Section 7 concludes this paper.

2. Related works

The existing uncertain data mining methods can be categorized into two types: those that achieve the expected frequent itemsets [6–17], and those that obtain the probabilistic frequent itemsets [18–30].

2.1. Expected frequent itemset mining

To address the problem of expected frequent itemset mining, three main methods and their data structures are proposed. The first one is the a priori method: Chui first proposed the uncertain frequent itemset mining method, *U-Apriori* [6], which used a data trimming framework to raise the mining efficiency; later, he used a decremented pruning method [7] to improve the performance. The second one is the tree-based method: Leung proposed the *UF-Growth* method with the novel tree structure *UF-tree* [8], which was further improved [10]; also, he presented a more tight upper bound of expected supports and then introduced a more efficient algorithm called *BLIMP-Growth* [11]. The third one is the H-struct-based method: Aggarwal employed H-struct, which is efficient in exact data mining, to perform uncertain frequent itemset mining, and proposed *UHMine* [9].

Besides mining the common results, some algorithms have focused on achieving more valuable and concise results. The constrained frequent itemset mining method *U-FPS* was also proposed in [12]; it uses user constraints to make the mining more efficient. In addition, Calders used sampling to achieve approximate results [13]. Also, Liu pursued research on how to discovery frequent itemsets from univariate uncertain data [31], and Pei focused on mining fuzzy association rules over probabilistic quantitative databases [32].

In addition, based on database mining methods, stream mining algorithms have also been proposed. In [14], the *UF-streaming* algorithm used an immediate mode to get approximate results, and the *SUF-growth* algorithm employed a delayed mode to obtain the exact frequent itemsets; in [15,16], a time-fading model-based algorithm and a landmark model-based mining algorithm were proposed. Moreover, Leung also studied mining uncertain frequent itemsets with the map-reduce framework [17]. These algorithms can get the expected frequent itemsets in real time, but they also ignored some history results.

For this problem, the expected support will be computed with $O(n)$ time complexity and $O(1)$ space complexity for each expected itemset, which is a huge advantage in performance. The expected frequent itemsets, nevertheless, cannot present the whole data probabilistic characteristics, especially when the data do not have enough transactions.

2.2. Probabilistic frequent itemset mining

Probabilistic frequent itemsets can better discover probabilistic features [25], which entail high computing and memory cost in mining. Generally, the methods can also be split two types: the a priori type and the fp-tree type. Zhang first introduced the concept of probabilistic frequent items [18], and employed dynamic programming (DP) to perform mining. This technique was improved upon by Bernecker, who proposed *ProApriori* [19] using the a priori rule for further pruning, as well proposed *ProFPGrowth* [21] with *ProFP-tree* to maintain the itemsets. Sun regarded the probability computation as the convolution of two vectors; thus, he used the divide-and-conquer(DC) method [20] to conduct mining, during which the Fast Fourier Transform (FFT) was used to reduce the computing complexity from $O(n^2)$ to $O(nlog^2n)$, thereby reducing the computing cost.

The probabilistic frequent itemset and the expected frequent itemset were proved to be related in [22, 23] based on a Poisson distribution and in [24] based on a standard normal distribution, and thus two approximate algorithms were proposed.

Recently, Tang began to focus on mining focused on probabilistic frequent closed itemsets and presented the exact algorithm *PFCIM* [26] and the approximate algorithm *A-PFCIM* [27]. Tong introduced the new definition of probabilistic frequent closed itemset when the items in one transaction have the same probability, and proposed the *MPCFI* method. Liu continued to pursue research on mining compressed itemsets, which are called probabilistic representative frequent itemsets; one proposed method, *P-RFP* [29], employed the distance measure to combine the itemsets, and another method, *APM* [30], summarized the frequent itemsets by joining the support probabilities approximately, which enables one to estimate the probability that one itemset represents another. In all these algorithms, the goal, like ours, is to get a representation of the mining results. Furthermore, in this paper, we will use a more effective representation, that is, the probabilistic maximal itemset; and we will find more efficient mining algorithms to get the results.

3. Preliminaries and problem definition

A brief review of the relevant concepts of uncertain database and frequent itemset mining are presented in this section. We then theoretically define the problem addressed in this paper.

3.1. Preliminaries

3.1.1. Uncertain database and possible worlds

Given a set of distinct items $\Gamma = \{i_1, i_2, \cdots, i_n\}$ where $|\Gamma| = n$ denotes the size of Γ, a subset $X \subseteq \Gamma$ is called an itemset. Suppose each item $x_t (0 < t \leqslant |X|)$ in X is associated with an occurrence probability $p(x_t)$. We call X an uncertain itemset, which is denoted as $X = \{x_1, p(x_1); x_2, p(x_2); \cdots; x_{|X|}, p(x_{|X|})\}$. An uncertain transaction UT is an uncertain itemset with an ID. An uncertain database UD is a collection of uncertain transactions $UT_s (0 < s \leqslant |UD|)$. The left table of Fig. 1 shows a simple uncertain database. Using a possible world model, an uncertain database can be converted to multiple exact databases.

Definition 1. *(Possible World)* A possible world PW converted from uncertain database UD is an exact database with $|UD|$ transactions, where each transaction T_s is a subset of UT_s; thus, we denoted a possible world as $PW = \{T_1, T_2, \cdots, T_{|UD|}\}$, in which $T_s \subseteq UT_s (0 < s \leqslant |UD|)$.

ID	TRANSACTION
1	{A 0.6} {B 0.7}
2	{A 0.2} {C 0.3}

Ucertain DataBase

PID	POSSIBLE WORLD	PROB.	PID	POSSIBLE WORLD	PROB.
1	{} {}	0.0672	9	{A} {C}	0.0432
2	{A} {}	0.1008	10	{} {AC}	0.0072
3	{} {A}	0.0168	11	{B} {C}	0.0672
4	{B} {}	0.1568	12	{AB} {C}	0.1008
5	{} {C}	0.0288	13	{B} {AC}	0.0168
6	{A} {A}	0.0252	14	{AB} {A}	0.0588
7	{AB} {}	0.2352	16	{A} {AC}	0.0108
8	{A} {B}	0.0392	16	{AB} {AC}	0.0252

Possible Worlds

Fig. 1. Uncertain transaction database vs. possible worlds.

If we assume that the uncertain transactions are independent, a possible world has an occurrence probability $p(PW)$, which can be computed by multiplying the occurrence probability of each item $x \in PW$ if x is in T_s and UT_s, as well as an occurrence probability of each item $\overline{x}$ if x is in UT_s but not T_s, denoted as:

x thuoc Ts, x thuc UTs

$$p(PW) = \prod_{x \in UT_s} \left(\prod_{x \in T_s} p(x) \right) \left(\prod_{x \notin T_s} p(\overline{x}) \right) \tag{1}$$

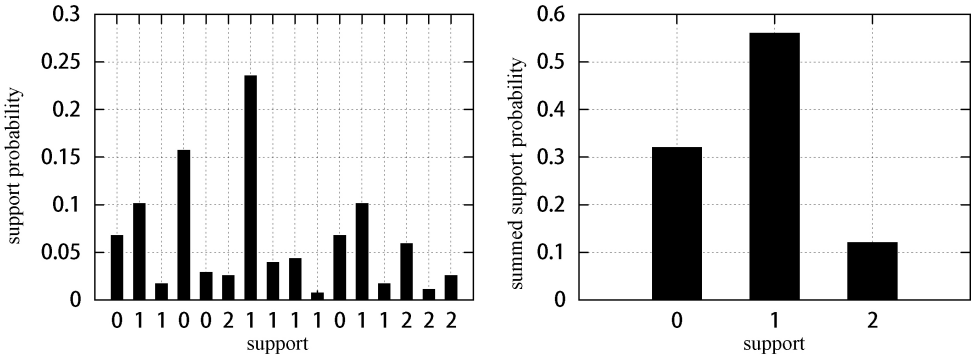
Ts thuoc PW

x ko thuoc Ts , x thuoc UTs

We denote all the possible worlds generated from UD as Ψ. The scale of Ψ, however, is exponentially increased following the original uncertain database size. To describe it in detail, suppose that an uncertain database includes m transactions, each of which covers n_m items. Then Ψ includes $2^{\sum_{i=1}^{m} n_i}$ possible worlds. Consider the uncertain database presented in the left table of Fig. 1. There are two transactions in the database, and each transaction has two items; thus, we materialize them to a possible world model presented in the right table of Fig. 1, which includes $2^{2+2} = 16$ possible worlds. Each possible world has a probability according to Eq. (1). As an example, consider two transactions in the possible world 6, denoted T_1 and T_2, which are both $\{A\}$, then $p(PW_6) = p_{\{A\} \in UT_1 \wedge \{A\} \in T_1}(\{A\}) * p_{\{B\} \in UT_1 \wedge \{B\} \notin T_1}(\overline{\{B\}}) * p_{\{A\} \in UT_2 \wedge \{A\} \in T_2}(\{A\}) * p_{\{C\} \in UT_2 \wedge \{C\} \notin T_2}(\overline{\{C\}}) = 0.6 * 0.3 * 0.2 * 0.7 = 0.0252$. In summing the probabilities of all possible worlds, we find that the summary is 1.

3.1.2. Frequent itemset over uncertain databases

In an exact database D, given a user-specified threshold (also called the minimum support) $\lambda (1 < \lambda < |D|)$, a frequent itemset is an itemset that occurs in at least λ transactions. Also we can say that an itemset X is frequent if the support $\Lambda_D(X)$ is not less than λ. In an uncertain database UD, the frequent itemset can be defined with the possible world model. That is, for an itemset X in each possible world PW generated from UD, we first obtain the occurrence number $\Lambda_{PW}(X)$ with an occurrence probability $p_{PW}(X)$, which in fact, is the probability of PW $p(PW)$; We denote them with a 2-tuple $< \Lambda_{PW}(X), p_{PW}(X) >$, which means that, X has a support $\Lambda_{PW}(X)$ with probability $p_{PW}(X)$. Actually, X in UD has $2^{\sum_{i=1}^{m} n_i}$ such tuples, which are reorganized and represented with a summed probabilistic vector,



(a) Support probabilistic vector (b) Summed support probabilistic vector

Fig. 2. Probabilistic support of itemset $\{A\}$.

denoted as $P_{\Lambda(X)}(X)$. For instance, in Fig. 1, itemset $\{A\}$ has 16 values in a probabilistic vector, which is shown in Fig. 2a; as can be seen, $\{A\}$ does not occur in possible world pw_1, pw_4, pw_5, and pw_{11}, and the corresponding probabilities are 0.0672, 0.1568, 0.0288, and 0.0672. To accumulate the probabilities of the same supports, we can get the summed probabilistic vector, $P_{\Lambda(\{A\})}(\{A\})$, shown in Fig. 2b, that is, $\{A\}$ does not occur with probability 0.32 in an uncertain database. Based on the summed probabilistic vector, two definitions, the expected frequent itemset and the probabilistic frequent itemset, have been proposed.

Definition 2. *(Expected Frequent Itemset [6])* Given an uncertain database UD, an itemset X is an λ-expected frequent itemset if its expected support $\Lambda^E(X)$ is not less than its minimum support λ. Here

$$\Lambda^E(X) = \sum_{PW \in \psi} \Lambda_{PW}(X) p_{PW}(X) \tag{2}$$

The expected frequent itemset is an intuitive consideration. As an example from Fig. 2a, that the expected support of itemset $\{A\}$ is $\Lambda^E(\{A\}) = 0 * 0.32 + 1 * 0.56 + 2 * 0.12 = 0.8$. As can be seen, it can also be computed by accumulating all the probabilities of $\{A\}$ in UD transactions; from Fig. 1, $\Lambda^E(\{A\}) = p_{t1}(\{A\}) + p_{t2}(\{A\}) = 0.6 + 0.2 = 0.8$. Thus, a simplified expected support computation of itemset X is $\Lambda^E(X) = \sum_{t \in UD} p_t(X)$. Earlier research works were mainly based on this definition.

Nevertheless, the definition of expected frequent itemset cannot reflect the uncertainty of an itemset [19]. Consequently, the probabilistic frequent itemset was proposed.

Definition 3. *(Probabilistic Frequent Itemset [19])* Given an uncertain database UD, the minimum support λ, and the minimum probabilistic confidence τ, an itemset X is called an (λ, τ)-probabilistic frequent itemset if the probability of its support not less than λ, denoted $P_{\Lambda(X) \geqslant \lambda}(X)$, is larger than τ. Here

$$P_{\Lambda(X) \geqslant \lambda}(X) = \sum_{PW \in \psi, \Lambda_{PW}(X) \geqslant \lambda} p_{PW}(X) \tag{3}$$

The probabilistic frequent itemset focuses on obtaining the possibility of an itemset occurring in at least λ transactions. For the same example of itemset $\{A\}$, given the minimum support $\lambda = 2$, $P_{\Lambda(\{A\}) \geqslant 2}(\{A\}) = p_{PW6}(\{A\}) + p_{PW14}(\{A\}) + p_{PW15}(\{A\}) + p_{PW16}(\{A\}) = 0.0252 + 0.0588 + 0.0108 + 0.0252 = 0.12$. That is, itemset $\{A\}$ is a "$\geqslant 2$" frequent itemset with 0.12 probability.

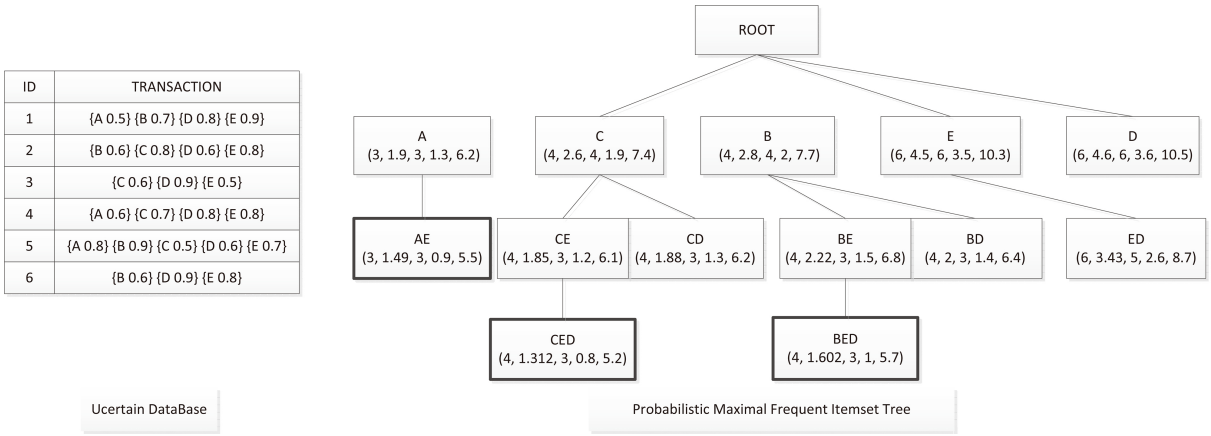


Fig. 3. Probabilistic maximal frequent itemset tree (PMFIT) for $\lambda = 3$, $\tau = 0.1$.

Nevertheless, because Definition 3 is not consistent with the traditional concept of frequent itemset, it is not convenient enough to define the related concepts in exact databases, such as closed and maximal itemsets. Therefore, a new equivalent definition of probabilistic frequent itemset with a different description was proposed in [26].

Definition 4. *(Probabilistic Support [26])* Given the minimum probabilistic confidence τ, an uncertain database UD and an itemset X, the probabilistic support of X denoted as $\Lambda_\tau^P(X)$, is the maximal support of itemset X with probabilistic confidence τ, that is,

$$\Lambda_\tau^P(X) = Max\{i | P_{\Lambda(X)\geqslant i} > \tau\} \tag{4}$$

Definition 5. *(Probabilistic Frequent Itemset [26])* Given the minimum support λ, the minimum probabilistic confidence τ and an uncertain database UD, an itemset X is a probabilistic frequent itemset if $\Lambda_\tau^P(X) \geqslant \lambda$.

Property 1. Given two itemsets X and Y, the minimum probabilistic confidence τ and an uncertain database UD, if $X \subseteq Y$, then $\Lambda_\tau^P(X) \geqslant \Lambda_\tau^P(Y)$ [26].

Property 2. Given an itemsets X and an uncertain database UD, for two minimum probabilistic confidence τ_1 and τ_2, if $\tau_1 < \tau_2$, then $\Lambda_{\tau_1}^P(X) \geqslant \Lambda_{\tau_1}^P(X)$.

Proof. Suppose that $\Lambda_{\tau_1}^P(X) < \Lambda_{\tau_1}^P(X)$ when $\tau_1 < \tau_2$. Let $\Lambda_{\tau_1}^P(X) = t_1$ and $\Lambda_{\tau_2}^P(X) = t_2$, then $t_1 < t_2$. According to Definition 4, $P_{\Lambda(X)\geqslant t_1} > \tau_1$, $P_{\Lambda(X)\geqslant t_2} > \tau_2$, and also $P_{\Lambda(X)\geqslant t_1+1} \leqslant \tau_1$, $P_{\Lambda(X)\geqslant t_2+1} \leqslant \tau_2$. Since $t_1 < t_2$, then $t_1 + 1 \leqslant t_2$, and then $P_{\Lambda(X)\geqslant t_1+1} \geqslant P_{\Lambda(X)\geqslant t_2}$ (Lemma 14 in [19]). Consequently, we can get that $\tau_2 < P_{\Lambda(X)\geqslant t_2} \leqslant P_{\Lambda(X)\geqslant t_1+1} \leqslant \tau_1$, which contradicts $\tau_1 < \tau_2$. As a result, if $\tau_1 < \tau_2$, then $\Lambda_{\tau_1}^P(X) \geqslant \Lambda_{\tau_1}^P(X)$. □

Property 2 suggests that the larger the minimum probabilistic confidence is, the smaller is the possibility that an itemset becomes frequent. In the uncertain database of Fig. 3, if $\lambda = 3$, itemset $\{A\}$ is frequent if $\tau = 0.1$, but it is not frequent if $\tau > 0.3$.

3.2. Problem definition

In this paper we employ Definition 5 to define the maximal frequent itemset of uncertain databases, which is called the probabilistic maximal frequent itemset.

Table 1
Summary of notations

Notation	Meaning
UD	Uncertain database
λ	Minimum support
λ_r	Relative minimum support $\left(=\frac{\lambda}{\|UD\|}\right)$
τ	Minimum probabilistic confidence
$\Lambda(X)$	The support of itemset X (the frequency itemset X occurring in UD)
$\Lambda^E(X)$	The expected support of itemset X
$P_{\Lambda(X)}(X)$	The probabilities of itemset X's supports
$P_{\Lambda(X)\geqslant\lambda}(X)$	The probability of itemset X's support not smaller than λ
$\Lambda_\tau^P(X)$	The probabilistic support of itemset X

Definition 6. *(Probabilistic Maximal Frequent Itemset)* Given the minimum support λ, the minimum probabilistic confidence τ, and an uncertain database UD, an itemset X is a probabilistic maximal frequent itemset if it is a probabilistic frequent itemset and is not covered by the other probabilistic frequent itemsets, denoted as $\Lambda_\tau^P(X) \geqslant \lambda \wedge \nexists\{Y|Y \supset X \wedge \Lambda_\tau^P(Y) \geqslant \lambda\}$.

Since Definitions 3 and 5 have been proven to be consistent in [26], Definition 6 is quite reasonable. One can easily see that the support information of a probabilistic frequent itemset $Y \subset X$ can be estimated from the probabilistic maximal frequent itemset X without having to again read from the database. In other words, for a probabilistic maximal frequent itemset X, any itemset Y that $Y \subset X$ satisfy the following statement: The probability of Y's support being no less than λ is larger than τ.

Problem statement: Based on the previous definitions, we summarize the notations in Table 1 and present our problem statement as follows: Given an uncertain database UD, the minimum support λ, and the minimum probabilistic confidence τ, we are required to explore all probabilistic maximal frequent itemsets in UD.

Example 1. For uncertain database UD in Fig. 1, the minimum support $\lambda = 1$ and the minimum probabilistic confidence $\tau = 0.1$, we can get the following results: $\Lambda^P_{\tau=0.1}(\{A\}) = 2$, $\Lambda^P_{\tau=0.1}(\{B\}) = 1$, $\Lambda^P_{\tau=0.1}(\{C\}) = 1$, $\Lambda^P_{\tau=0.1}(\{AB\}) = 1$, $\Lambda^P_{\tau=0.1}(\{AC\}) = 0$. The probabilistic frequent itemsets are $\{A\}$, $\{B\}$, $\{C\}$, and $\{AB\}$, and the probabilistic maximal frequent itemsets are $\{C\}$ and $\{AB\}$. As can be seen, the probabilistic maximal frequent itemsets can reduce storage usage. Also, we can at least infer from the probabilistic maximal frequent itemsets that the probability of itemset $\{A\}$'s support being no less than 1 is larger than 0.1.

4. Probabilistic maximal frequent itemset mining method

In this section, we will first introduce our compressed data structures and an enumeration tree with auxiliary data collections. We then further introduce our pruning strategies, and present the details of our mining algorithm.

4.1. Data structures

4.1.1. Probabilistic mining frequent itemset tree

To accelerate searching and pruning, we design a simple but effective index tree called the *PMFIT* (Probabilistic Maximal Frequent Itemset Tree), in which each node n_X denote an itemset X; n_X is a 6-tuple $< item, sup, esup, psup, lb, ub >$, in which *item* denotes the last item of the current itemset X, *sup*

is the support, *esup* is the expected support, and *psup* is the probabilistic support. lb and ub separately represent the lower and upper bounds of probabilistic support. Except for the root node, each node has a pointer to its parent node. Figure 3 shows the *PMFIT* from the six transactions sampled from our experimental databases. For an example, if n_A denotes itemset $\{A\}$ with support 3, expected support 1.9, and probabilistic support 3, then, as can be seen, 13 nodes are probabilistic frequent itemsets, and only 3 nodes are probabilistic maximal frequent itemsets.

4.1.2. Probability list

We also use an array during implementation to maintain the probability list of an itemset occurring in all transactions. This probabilities list is used to compute the probabilistic support. It is temporary, however, because in our algorithm the probabilistic support need not be computed for each itemset.

4.1.3. Probabilistic maximal frequent itemset collection

Because the final results do not have to maintain the probabilities of each item, we employed a traditional bitmap-based collection to store the probabilistic maximal frequent itemsets, thus reducing memory cost, and increasing the searching speed.

4.2. Probabilistic support computing

Because the summed support probabilistic vector of itemset X in two transactions T_1 and T_2 is the convolution of that in T_1 and that in T_2, the divide-and-conquer method proposed in [20] is also employed in our paper. That is, the uncertain database will be split into two parts to separately compute the summed support probabilistic vector, and this operation will be recursively conducted until the sub-database has only one transaction. The convolution can be computed with a Fast Fourier Transform, which, given the size of the uncertain database n, will efficiently reduce the time complexity from $O(n^2)$ to $O(nlog^2(n))$. With the computed summed support probabilistic vector $\{sp_0, sp_1, \cdots, sp_n\}$, we can get the probabilistic support, which is the maximal t s.t. $\sum_{i=t}^{n}(sp_i)$ larger than minimum probabilistic confidence τ.

4.3. Items reordering

Bayardo [3] stated that ordering items with increasing support can significantly reduce the search space; together with other pruning strategies, it can further reduce the cost of support computing. In this paper, we employed a similar heuristic rule with a slight differences. That is, the items will be ordered by their expected supports rather their supports. This change can be made because of the instinctive observation that two itemsets occurring in the same transactions may have different probabilities and thus have various expected supports. As a simple example in Fig. 3, itemset $\{B\}$ and itemset $\{C\}$ occur in four transactions, which means that their supports are both 4. Nevertheless, the expected supports are 2.8 and 2.6 separately. As a result, we believe that using the expected support to sort the items will make the algorithm more efficient. Actually, we presented in Section 5 that the expected support is more relevant to the probabilistic support; generally, the larger the expected support, the larger the probabilistic support. Note that even though probabilistic support is the best method to be used in sorting the items, we did not use it. This is because computing the probabilistic support consumes much runtime, which, will adversely affect performance when the minimum support is not low enough. In comparison to that, sorting by expected support will reduce computational cost, as mentioned in our experiments, and performance will be a somewhat better when the minimum support is high. In our implementation, we used an array, called the sorted items list, to maintain the sorted items.

4.4. Pruning strategies

We are inspired by the maximal frequent itemset mining methods over regular databases and propose some pruning strategies that can supply tight bounds to infer the probabilistic support or even ignore the computing of probabilistic support and thus improve performance.

4.4.1. Bounds of probabilistic support

For an n-transactions uncertain database, an efficient method to compute the probabilistic support is the divide-and-conquer method; however, the runtime cost and memory usage are still high. When mining the probabilistic maximal frequent itemsets, the probabilistic support is not important for the users, so we try to find a method to infer the frequency of the itemset rather than directly compute the probabilistic support.

Theorem 1. Given an itemset X, the minimum probabilistic confidence τ, and uncertain database UD, if X occurs in transaction T_a with probability p_a, then the probabilistic support $\Lambda_\tau^P(X)$ is monotonically consistent to any of p_a.

Proof. Without loss of generality, we consider $p_t(t = 1 \cdots |UD|)$ as a random selected probability. Because $P_{\Lambda_{UD}(X)\geqslant i} = P_{\Lambda_{UD-T_t}(X)\geqslant i-1} \times p_t + P_{\Lambda_{UD-T_t}(X)\geqslant i} \times (1 - p_t)$, in which $P_{\Lambda_{UD-T_t}(X)\geqslant i-1}$ is larger than $P_{\Lambda_{UD-T_t}(X)\geqslant i}$ (Lemma 14 of [19]), then $P_{\Lambda_{UD}(X)\geqslant i}$ and p_t change consistently. Again, without loss of generality, we suppose p_t increase, then $P_{\Lambda_{UD}(X)\geqslant i}$ increases, and thus an increased i also can keep $P_{\Lambda_{UD}(X)\geqslant i}$ larger than τ. In other words, the probabilistic support $\Lambda_\tau^P(X)$, which is the value of i, increases. □

Theorem 2. For an itemset X in uncertain database UD, given the minimum support λ and the minimum probabilistic confidence τ, then $\Lambda_\tau^P(X) \leqslant \Lambda(X)$.

Proof. From Theorem 1 we can see that when any probability p_a that itemset X occurs in transaction T_a increases, the probabilistic support $\Lambda_\tau^P(X)$ also increases. When $p_a = 1$, that is, p_a reaches to its upper bound, $\Lambda_\tau^P(X) = \Lambda(X)$. Thus, for any $p_a < 1$, $\Lambda_\tau^P(X) < \Lambda(X)$. □

Theorem 2 shows that the support of an itemset can be regarded as an upper bound of the probabilistic support. That is, for an itemset X, if its support $\Lambda(X)$ is less than the minimum support λ, then $\Lambda_\tau^P(X) < \lambda$, and it will be pruned directly. As an example, for the database in Fig. 3, the support of itemset $\{AB\}$ is 2, which is lower than the minimum support 3, thus it is definitely an infrequent itemset.

Theorem 3. For an itemset X in uncertain database UD, given the minimum probabilistic confidence τ, we can get the lower and upper bounds of the probabilistic support $\Lambda_\tau^P(X)$, denoted by $lb(\Lambda_\tau^P(X))$ and $ub(\Lambda_\tau^P(X))$ as follows.

$$\begin{cases} lb(\Lambda_\tau^P(X)) = \Lambda^E(X) - \sqrt{-2\Lambda^E(X)ln(1-\tau)} \\ ub(\Lambda_\tau^P(X)) = \frac{2\Lambda^E(X) - ln\tau + \sqrt{ln^2\tau - 8\Lambda^E(X)ln\tau}}{2} \end{cases} \tag{5}$$

Proof. For the itemset X, we use ε to denote the expected support $\Lambda^E(X)$, also, we use t to denote the probabilistic support $\Lambda_\tau^P(X)$, which, according to Definition 4, satisfies the following equations.

$$\begin{cases} P_{\Lambda(X)\geqslant t} > \tau \Leftrightarrow P_{\Lambda(X)>t-1} > \tau \\ P_{\Lambda(X)\geqslant t+1} \leqslant \tau \Leftrightarrow P_{\Lambda(X)>t} \leqslant \tau \end{cases} \tag{6}$$

1) If we set $t = (1+\xi)\varepsilon$, i.e., $\xi = \frac{t}{\varepsilon} - 1$, where $\xi \geqslant 0$, that is, $t \geqslant \varepsilon$, then based on the Chernoff Bound, $P_{\Lambda(X)\geqslant t} = P_{\Lambda(X)\geqslant(1+\xi)\varepsilon} \leqslant e^{-\frac{\xi^2\varepsilon}{2+\xi}}$; based on the first inequality of Eq. (6), we can get $\tau < e^{-\frac{\xi^2\varepsilon}{2+\xi}} = e^{-\frac{(t-\varepsilon)^2}{t+\varepsilon}}$; that is to say, when $t \geqslant \varepsilon$, $\frac{2\varepsilon - ln\tau - \sqrt{ln^2\tau - 8\varepsilon ln\tau}}{2} < t < \frac{2\varepsilon - ln\tau + \sqrt{ln^2\tau - 8\varepsilon ln\tau}}{2}$. Since $\frac{2\varepsilon - ln\tau - \sqrt{ln^2\tau - 8\varepsilon ln\tau}}{2} \leqslant \varepsilon$, we can obtain the following inequality.

$$t < \frac{2\varepsilon - ln\tau + \sqrt{ln^2\tau - 8\varepsilon ln\tau}}{2} \quad if \quad t \geqslant \varepsilon \tag{7}$$

2) If we set $t = (1-\xi')\varepsilon$, i.e., $\xi' = 1 - \frac{t}{\varepsilon}$, where $\xi' \geqslant 0$, that is, $t \leqslant \varepsilon$, then based on the Chernoff Bound, $P_{\Lambda(X)>t} = P_{\Lambda(X)>(1-\xi')\varepsilon} > 1 - e^{-\frac{\xi'^2\varepsilon}{2}}$; based on the second inequality of Eq. (6), we can get $\tau > 1 - e^{-\frac{\xi'^2\varepsilon}{2}}$, i.e., $-\sqrt{-\frac{2ln(1-\tau)}{\varepsilon}} < \xi' < \sqrt{-\frac{2ln(1-\tau)}{\varepsilon}}$, then when $t \leqslant \varepsilon$, $\varepsilon - \sqrt{-2\varepsilon ln(1-\tau)} < t < \varepsilon + \sqrt{-2\varepsilon ln(1-\tau)}$. Since $\varepsilon \leqslant \varepsilon + \sqrt{-2\varepsilon ln(1-\tau)}$, we can get the following inequality.

$$t > \varepsilon - \sqrt{-2\varepsilon ln(1-\tau)} \quad if \quad t \leqslant \varepsilon \tag{8}$$

From Eqs (7) and (8), we can conclude that no matter t is larger or less than the ε, it is definitely within the range of $\left(\varepsilon - \sqrt{-2\varepsilon ln(1-\tau)}, \frac{2\varepsilon - ln\tau + \sqrt{ln^2\tau - 8\varepsilon ln\tau}}{2}\right)$. □

Theorem 3 provides us with two pruning strategies. For an itemset X, if the upper bound $\frac{2\varepsilon - ln\tau + \sqrt{ln^2\tau - 8\varepsilon ln\tau}}{2}$ is not larger than the minimum support λ, then X is a probabilistic infrequent itemset. Also, if the lower bound $\varepsilon - \sqrt{-2\varepsilon ln(1-\tau)} \geqslant \lambda$, then X is definitely a probabilistic frequent itemset. We can see that, for an uncertain database of size n, if the minimum support λ is not within this range, we can successfully hit the target, that is, there is only a possibility $\frac{2\sqrt{-2\varepsilon ln(1-\tau)} - ln\tau + \sqrt{ln^2\tau - 8\varepsilon ln\tau}}{2n}$ of failing to infer the frequency of the itemset. Clearly the possibility decides the expected support and the data size; i.e., the lower the expected support, or the larger the data size, the more efficient is our deriving method.

Example 2. We used the uncertain database in Fig. 3 as an example. If we set the minimum support $\lambda = 1$ and the minimum probabilistic confidence $\tau = 0.1$, then for itemset $\{A\}$, the lower bound is 1.3, which is larger than 1, and then itemset $\{A\}$ is a frequent itemset. Further, if we set the minimum support $\lambda = 5$ and the minimum probabilistic confidence $\tau = 0.1$, then for itemset $\{AB\}$, the upper bound is 4.7, which is less than 5, and thus itemset $\{AB\}$ is an infrequent itemset.

Note that we can use the support $\Lambda(X)$ and the $ub(\Lambda_\tau^P(X))$ computed with Eq. (5) as the upper bound. The former can be obtained without any parameters, and thus it is fixed; the latter need to be computed with a given minimum probabilistic confidence, so it is more flexible. Generally, the lower the probabilities that an itemset have in transactions, the lower the value of $ub(\Lambda_\tau^P(X))$, and the larger the possibility that we use it as the final upper bound. In our implementation we just select the minimal value of them to be the final upper bound.

4.4.2. Superset pruning

According to our definition, an itemset being probabilistic maximal frequent must satisfy two conditions: 1. The probabilistic support is not less than a threshold; 2. It is not covered by any other probabilistic frequent itemset. Both computation are needed, so the computation with the lower computing cost should be conducted first. As mentioned above, computing the probabilistic support requires $O(nlog^2(n))$ time complexity in which n is the database size, even though, with our derived method, the time complexity is $O(n)$. However, to scan the existing probabilistic maximal frequent itemsets, whose

size is assumed to be m, requires at most $O(m)$ time complexity. Generally, m is much less than n. Consequently, for a new generated itemset, we will first decide whether it is covered by a super itemset, then, if not, we compute the bounds or the value of probabilistic support. This strategy can be extended for further pruning. That is, for a probabilistic maximal frequent itemset $X = \{x_1 x_2 \cdots x_n\}$, if they are the last n items in the sorted items list, then items $x_2, \cdots, x_n$ can be pruned directly for further computation. We can see from Fig. 3 that because $\{CDE\}$ is an itemset in which the items are the last three in the sorted items list, then $\{D\}$ and $\{E\}$ will be removed, and the computation of all their descendants can be pruned.

4.5. Algorithm description

In this paper, we propose a depth-first probabilistic maximal frequent itemset mining (PMFIM) algorithm to build the bottom-up organized tree; that is, the subsets will be computed first, and then the supersets will be generated if their subsets are all frequent. We discover the itemsets in this manner because the probabilistic frequent itemset also has the apriori property as presented in Property 1. The algorithm can be implemented in five steps. Algorithm 1 shows the pseudo code of Steps 3–5.

Step 1: We get all the distinct items and sort them in an incremental order according to their expected supports before we build the *PMFIT*. In addition, items with support or upper bound lower than the minimum support will be initially pruned.

Step 2: The PMFIT is initialized with only one root node, which represent the null itemset.

Step 3: For a parent node, we begin to generate the child node, and we compute to decide whether it is frequent. First, if the child node is covered by one of the maximal frequent itemsets, it is not a maximal but a frequent itemset. If all the items in it are the last ones in the sorted

Algorithm 1 PMFIM Algorithm

Require: n_I: node of *PMFIT* denote itemset I; UD: Uncertain Database; *PMFIC*: Probabilistic Maximal Frequent Itemset Collection; λ: minimum support; τ: minimum probabilistic confidence;

```
1: get the itemsets J (|J| = |I|) order larger than I;
2: for each itemset J do
3:    if J is probabilistic maximal frequent itemset and J is the last items in the sorted items list then
4:       break;
5:    generate the child node n_{I∪J} of n_I;
6:    if I ∪ J ∈ PMFIC then
7:       CALL PMFIM (n_{I∪J}, UD, PMFIC, λ, τ);
8:       continue;
9:    compute Λ^E(I ∪ J) and Λ(I ∪ J);
10:   compute lb(Λ^P_τ(I ∪ J)) and ub(Λ^P_τ(I ∪ J));
11:   if Min (Λ(I ∪ J), ub(Λ^P_τ(I ∪ J)) ⩽ λ) then
12:      delete n_{I∪J};
13:      continue;
14:   if lb(Λ^P_τ(I ∪ J)) ⩾ λ then
15:      CALL PMFIM (n_{I∪J}, UD, PMFIC, λ, τ);
16:   else
17:      compute Λ^P_τ(I ∪ J);
18:      if Λ^P_τ(I ∪ J) ⩾ λ then
19:         CALL PMFIM (n_{I∪J}, UD, PMFIC, λ, τ);
20:      else
21:         delete n_{I∪J};
22: if n_I has no children and I is not in PMFIC then
23:    add I in PMFIC;
```

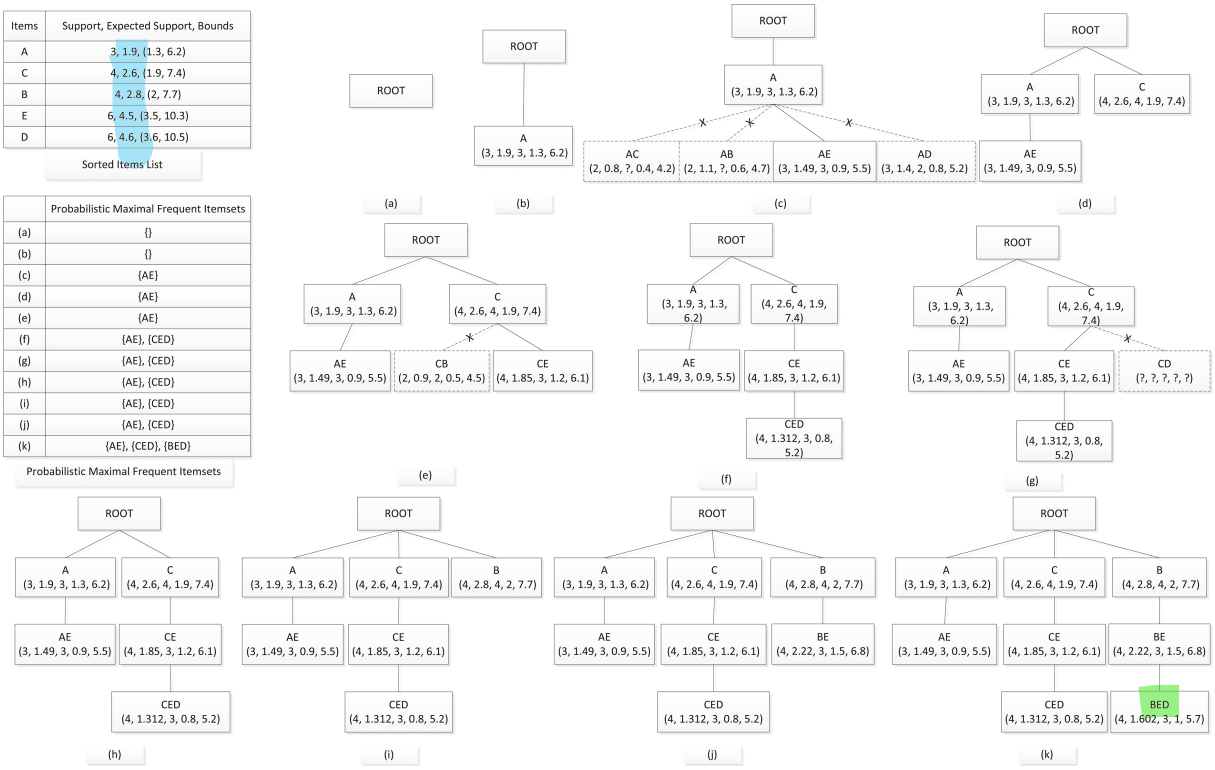


Fig. 4. Building the PMFIT for $\lambda = 3$, $\tau = 0.1$.

items list, we can determine that all the right nodes are not the final results, and the loop will be ended immediately. Second, after the support, the expected support and the support bounds are computed, if the upper bound is not larger than the minimum support, then it is an infrequent itemset, and if the lower bound is not less than the minimum support, then it is a frequent itemset. Finally, if we can not determine the frequency by using the previous value, we need to compute the probabilistic support and compare it to the minimum support.

Step 4: If a child node is frequent, we will recall Step 3 for it; otherwise, it will be pruned.

Step 5: If a node has no children and is not in the final results, it is a probabilistic maximal frequent itemset. We can add it to the probabilistic maximal frequent itemset collection. Because of our depth-first mining manner, there is no possibility of removing the subset from the collection.

Example 3. We again use the uncertain database in Fig. 3 as an example. Let the minimum support be $\lambda = 3$ and the minimum probabilistic confidence be $\tau = 0.1$. Figure 4 shows the process of building the *PMFIT*, in which each node has five values with the meaning as presented in Section 4.1.1. The details are as follows.

1) The sorted items list has five items ordered by the expected support incrementally. All of them have "$\geqslant 3$" supports.
2) Itemset $\{A\}$ is generated as a child node of the root. Because the lower bound is less than 3 and the upper bound is larger than 3, we need to compute the probabilistic support, which is 3; then $\{A\}$ is frequent.

3) Itemset $\{AC\}$, $\{AB\}$, $\{AE\}$, and $\{AD\}$ are generated as the children nodes of $\{A\}$. $\{AC\}$ and $\{AB\}$ have "< 3" supports, and $\{AD\}$ has < 3 probabilistic support; thus, only $\{AE\}$ is frequent. Because $\{AE\}$ has no children, it is a probabilistic maximal frequent itemset.
4) Itemset $\{C\}$, $\{B\}$, $\{CE\}$, and $\{CED\}$ are generated sequentially. $\{B\}$ is pruned because the probabilistic support is < 3, and $\{CED\}$ is the probabilistic maximal frequent itemset. Because $\{CD\}$ is covered by $\{CED\}$, it will be directly pruned.
5) Itemset $\{B\}$, $\{BE\}$, and $\{BED\}$ are also generated, and $\{BED\}$ is the probabilistic maximal frequent itemset, which is also the last three items of the sorted items list. Then all the left nodes will be pruned directly.

Note that in this example we never use the pruning of support bounds, which does not mean that it is not useful, only just that the data size is too small. In our experiments, we find that, when the data size is large, our pruning from support bounds is efficient.

5. Approximate probabilistic maximal frequent itemset mining method

Even though we employ pruning strategies to reduce the runtime cost, there is still the possibility that we have to compute the probabilistic supports in the *PMFIM* algorithm. In this section, we propose an efficient way to approximately estimate the probabilistic support from the expected support and the variance [24].

For an itemset X, whose occurrence probability p_i in a transaction T_i is supposed to a single coin toss, in an uncertain database UD, the possibilities of the supports of X, $P_{\Lambda(X)}(X)$, follows a binomial distribution with an expectation $\Lambda^E(X) = \sum_{i=1}^{|UD|}(p_i)$ and a variance $\Lambda^V(X) = \sum_{i=1}^{|UD|}(p_i \times (1 - p_i))$. When the size of UD increases, they approximately converge to the standard normal distribution. The probability of $\Lambda(X) = t$ can be computed by the standard normal distribution computation of t. If we use Φ to denote the cumulative distribution function of the standard normal distribution, then we can get the following equation.

$$P_{\Lambda(X)\geqslant t}(X) = 1 - \Phi\left(\frac{t - \Lambda^E(X)}{\sqrt{\Lambda^V(X)}}\right) \tag{9}$$

Theorem 4. Given an itemset X, an uncertain database UD, a minimum support λ, a minimum probabilistic confidence τ, the probabilistic support can be computed by the following equation.

$$\Lambda_\tau^P(X) \approx \left\lfloor \Phi^{-1}(1-\tau)\sqrt{\Lambda^V(X)} + \Lambda^E(X) \right\rfloor \tag{10}$$

Proof. From our Definition 6, $\Lambda_\tau^P(X)$ is the maximal t from Eq. (9) to satisfy $P_{\Lambda(X)\geqslant t}(X) > \tau$, that is, $1 - \Phi\left(\frac{t-E(\Lambda(X))}{\sqrt{V(\Lambda(X))}}\right) > \tau$; thus, $t < \Phi^{-1}(1-\tau)\sqrt{V(\Lambda(X))} + E(\Lambda(X))$, the maximal integer of t is $\lfloor \Phi^{-1}(1-\tau)\sqrt{V(\Lambda(X))} + E(\Lambda(X)) \rfloor$. □

Based on our computing method of the probabilistic support, if the approximate probabilistic support is not less than the minimum support, then the itemset is a probabilistic frequent itemset, with which we can reduce the time complexity from $O(nlog^2(n))$ to $O(n)$ and reduce the space complexity from $O(n)$ to $O(1)$. Note that our approximate method is based on the assumption that the occurrence of an itemset has a standard normal distribution. Also, in comparison to the derivation in Theorem 3, in computing the approximate probabilistic support, we need to obtain the variance.

Consequently, we employ the same framework in Algorithm 1 with a slight difference to implement the approximate probabilistic maximal frequent itemset mining (*APMFIM*) method. On the one hand, we sort the 1-items by their approximate probabilistic support rather than the expected support, which can further reduce the search space. On the other hand, we will prune the itemsets directly according to the approximate probabilistic support.

6. Experiments

We conducted experiments to evaluate the performance of the *PMFIM* and *APMFIM* methods. The state-of-the-art top-down algorithm *TODIS-MAX* [20], which has been presented in [20] as much more efficient than *pApriori*, was used as the evaluation method. The dataset size, the minimum support and the minimum probabilistic confidence are the main elements that may affect uncertain data mining. Therefore, these properties were used to compare the algorithms in runtime and memory cost.

6.1. Running environment and datasets

All the algorithms were implemented with C++, compiled with Visual Studio 2010 running on Microsoft Windows 7 and performed on a PC with a 2.90 GHZ Intel Core i7-3520M processor and 8 GB of main memory.

Because there are no public real-world uncertain datasets, we used the datasets from [25], in which a probability of Gaussian distribution is assigned to each item. This technique is widely accepted in uncertain data mining research. We used three synthetic datasets and four real-life datasets to generate the uncertain datasets. Datasets *T25I15D320K*, *T25I15D320K*, and *T40I10D100K* are generated with the IBM synthetic data generator, in which *T25I15D320K* is to used to evaluate the scalability of the algorithms. The *KOSARAK* dataset contains the click-stream data of a Hungarian online news portal. The *ACCIDENTS* dataset is a report of multiple accidents. The *GAZELLE* dataset contains the clickstream data from e-commerce, and the *CONNECT4* dataset contains all legal eight-ply positions in the game of Connect Four in which neither player has won yet, and in which the next move is not forced. The detailed data characteristics are given in Table 2.

In a dataset, given the distinct item number μ and the average transaction length ν, an item will occur once in at most $\frac{\mu}{\nu}$ transactions on average. Thus, we can demonstrate the approximate correlation among transactions with $\frac{\mu}{\nu}$. For smaller values of $\frac{\mu}{\nu}$, the correlation of the transactions is high. As shown in the last column of Table 2, data in *CONNECT4* have the highest correlation; that is, *CONNECT4* is the densest of the six datasets; in contrast, *KOSARAK* is the sparsest dataset.

We also presented the mean and variance of each dataset. As can be seen, *T25I15D320K*, *T25I15D320K*, and *GAZELLE* datasets have high means and low variances, *T40I10D100K* and *CONNECT4* datasets have high means and high variances, the *ACCIDENTS* dataset has a low mean and a high variance, and the *KOSARAK* dataset has a low mean and a low variance.

6.2. Effect of data size

We first evaluated the scalability of these three algorithms; that is, we performed the algorithms with respect to different database sizes, which are shown in Fig. 5. The *T25I15D320K* dataset was used as the evaluation dataset; it was separately divided into the first 20,000, 40,000, 80,000, 160,000 and 320,000 transactions to run the algorithms. The relative minimum support and the minimum probabilistic

Table 2
Uncertain DataSet characteristics

Uncertain DataSet	nr. of trans.	avg. trans. length	min. trans. length	max. trans. length	nr. of items	mean	variance	trans. corr.
T25I15D320K	320 002	26	1	67	994	0.87	0.27	38
T25I15D100K	100 000	26	4	67	1000	0.88	0.27	38
T40I10D100K	100 000	39	4	77	1000	0.79	0.61	25
KOSARAK	990 002	8	1	2498	41 270	0.5	0.28	5159
ACCIDENTS	340 183	33	18	51	468	0.5	0.58	14
CONNECT4	67 557	43	43	43	129	0.78	0.65	3
GAZELLE	59 602	3	2	268	497	0.94	0.08	166

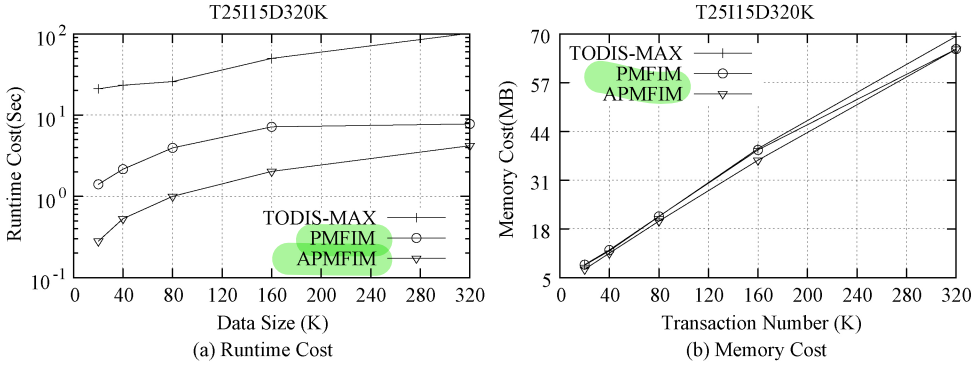


Fig. 5. Effect of data size for $\lambda_r = 0.1$, $\tau = 0.9$.

confidence were set to fixed values of 0.1 and 0.9, respectively. Note that, even though the relative minimum support is fixed, the minimum support will change because the data sizes are different. As shown in Fig. 5a, when the dataset became larger, the runtime cost of the three algorithms also increased, but the *PMFIM* and *APMFIM* algorithms were much more stable when the dataset became larger. Also, in Fig. 5b, we can see that, with the increase in the number of transactions, the memory cost of the algorithms increased linearly. This is reasonable because for more transactions, the expected support and the probabilistic support require more computation and memory usage; also, more probabilistic frequent itemsets will be maintained.

6.3. Effect of minimum probabilistic confidence

It has been proved in Property 2 that a larger minimum probabilistic confidence may result in a lower possibility that an itemset becomes frequent, which will reduce the computational cost and the memory usage. We evaluated the runtime and memory cost of our algorithms for different minimum probabilistic confidence (from 10^{-5} to 10^{-1}) when the relative minimum support was set to 0.9.

In Figs 6 and 7, we separately present the runtime cost and the memory usage over six datasets. We find that, when the minimum probabilistic confidence increases, both the runtime and the memory cost of the three algorithms remain almost unchanged. This demonstrates that the minimum probabilistic confidence has little effect on the performance of these three algorithms. This is because the summed support probabilistic vector is highly sparse when the dataset size is large, such that most of the minimum probabilistic confidence can slightly change the value of the support, and thus it can almost not turn the itemset from frequent to infrequent, or from infrequent to frequent.

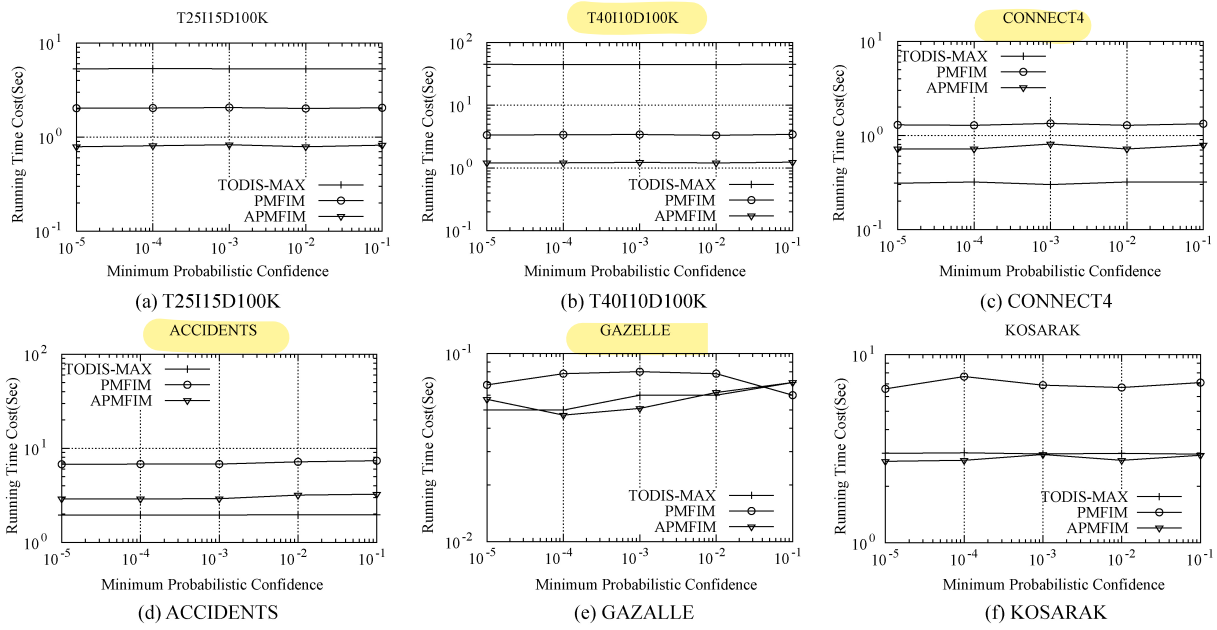


Fig. 6. Runtime cost vs. minimum probabilistic confidence for $\lambda_r = 0.9$.

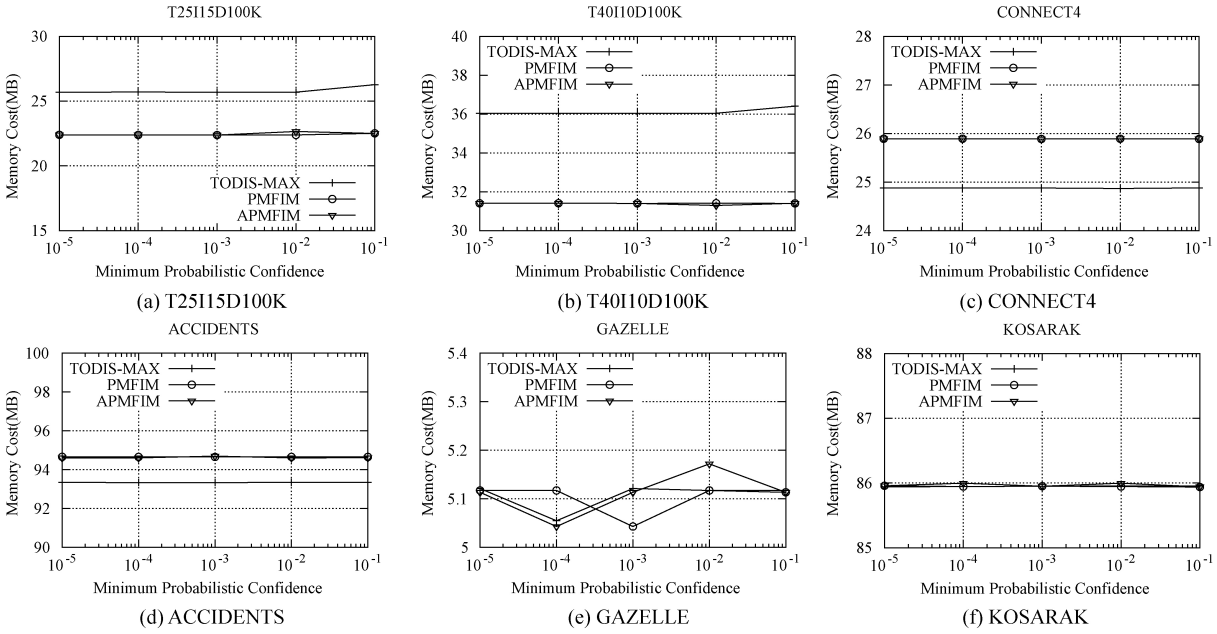


Fig. 7. Memory cost vs. minimum probabilistic confidence for $\lambda_r = 0.9$.

6.4. *Effect of minimum support*

Theoretically, the minimum support is a very important parameter that will impact performance. When the minimum support becomes larger, certain probabilistic frequent itemsets will become infrequent,

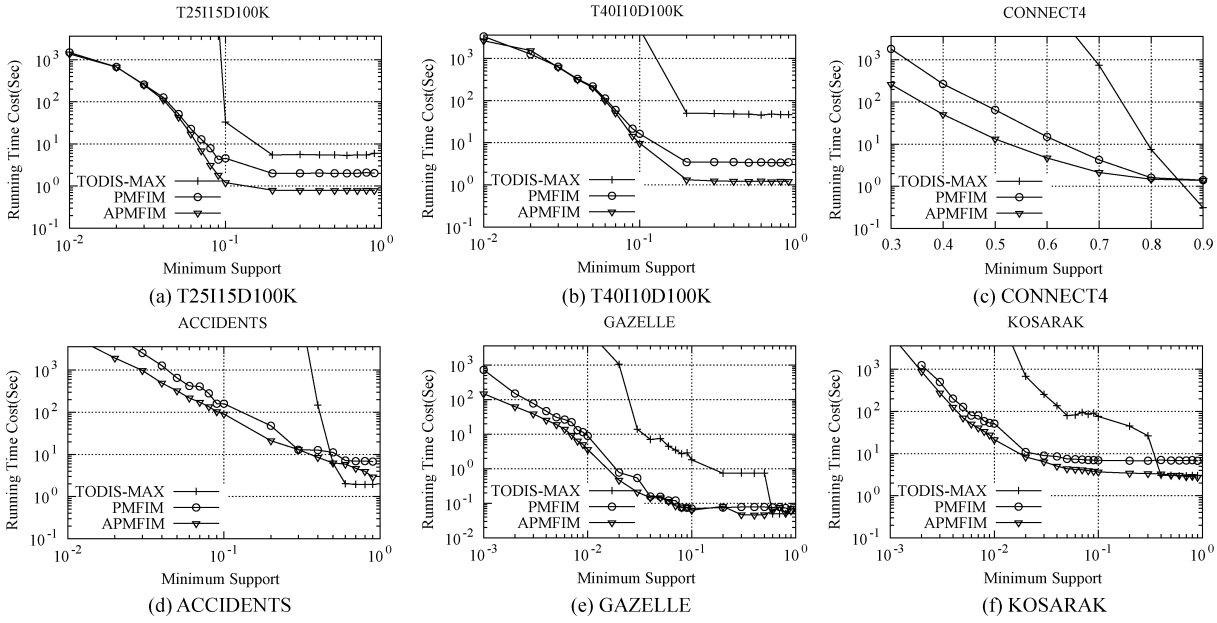


Fig. 8. Running time cost vs. minimum support for $\tau = 0.9$.

thereby reducing the runtime and memory cost. We set a fixed minimum probabilistic confidence, that is, $\tau = 0.9$, and compared the performance of the three algorithms when the minimum support was changed.

6.4.1. Runtime cost evaluation

We compared the runtime cost in Fig. 8 of the three algorithms being conducted over 6 datasets. Because of time constraints, we only conducted and presented the running results within one hour.

As can be seen, the runtime cost of all three algorithms decreased linearly with a decreasing of the minimum support, which is consistent with our theoretical analysis. We also compared our algorithms to *TODIS-MAX*. When the minimum support was low, the runtime cost of the *TODIS-MAX* method was a little higher than those of the *PMFIM* and *APMFIM* methods over most of the datasets except for *CONNECT4* and *ACCIDENTS*, the two densest datasets. This is due to the advantage of the top-down mining manner in *TODIS-MAX*; that is, faster pruning will be employed if the supersets are not frequent, which is more useful in terms of performance with dense datasets. However, when the minimum support becomes much higher, the performance of *TODIS-MAX* significantly deteriorates, which can also be clearly noticed over denser datasets. As shown in the figure, over *CONNECT4*, the densest dataset, *PMFIM* and *APMFIM* algorithms performed faster than *TODIS-MAX* by a factor of nearly 1000 when the relative minimum support was 0.7; over *KOSARAK*, the sparsest dataset, a factor of 100 increase in performance can also be achieved when the relative minimum support was 0.05. Also, this outperformance increased when the minimum support became lower. This is also because of the different mining fashions: *TODIS-MAX* employed the top-down method, which, when the minimum support is low, will use more frequent items to generate infrequent itemsets, whose number will exponentially increase along with the number of frequent items. Even though *TODIS-MAX* employed the expected support to prune most of the probability mass function computations, the remaining computations were still too numerous to calculate. In comparison, our two algorithms were bottom-up methods, with which

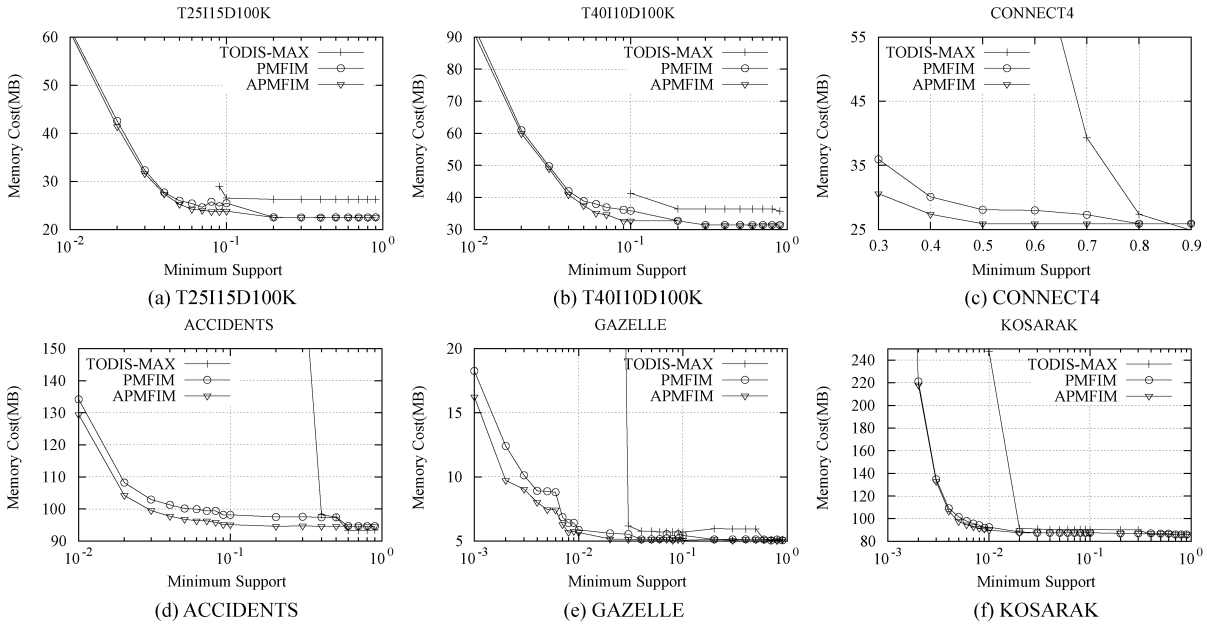


Fig. 9. Memory cost vs. minimum support for $\tau = 0.9$.

only frequent itemsets were generated, so, with the help of superset pruning, the number of computations will not increase sharply when the minimum support became lower.

In addition, we compared our two algorithms and noticed that when the minimum support was high, both algorithms had similar runtime cost, but the *APMFIM* algorithm was more efficient than the *PMFIM* algorithm when the minimum support turned lower; as shown in Fig. 8c, *APMFIM* performed nearly ten times faster than *PMFIM* when the relative minimum support changed to 0.3. This is because, in most cases, the *PMFIM* algorithm used the computed upper and lower bounds to infer whether an itemset was frequent, which had the same time complexity as that of the *APMFIM* algorithm. Unless the minimum support was much lower, this computation can replace almost all the probabilistic support computation. Two exceptions were the performance over *T25I15D100K* and *T40I10D100K* datasets. We find this to be reasonable because the *APMFIM* and *PMFIM* algorithms arrange items in different order; that is, in the *APMFIM* algorithm, the items were sorted by their approximate probabilistic support but, in the *PMFIM* algorithm, the items were sorted by their expected support. This difference may result in slightly varied performances even though the time complexities are similar.

Additionally, to compare the runtime under the same minimum support, we can easily find that when the transaction correlation is lower, which means that the dataset is denser, more runtime cost is needed. This conclusion is reasonable because a dense dataset will generate more frequent itemsets, which may consume more computing resources. As a result, we conclude that the transaction correlation is also an important factor that can affect the performance of the algorithms.

6.4.2. Memory cost evaluation

We also compared the running memory cost of the three algorithms. As shown in Fig. 9, similar to the runtime cost, the memory usages will decrease as the minimum support decreases. In addition, we can see that in most cases *TODIS-MAX* used more memory than our two algorithms. Because our algorithm employed the divide-and-conquer method proposed in *TODIS-MAX*, which had a space complexity of

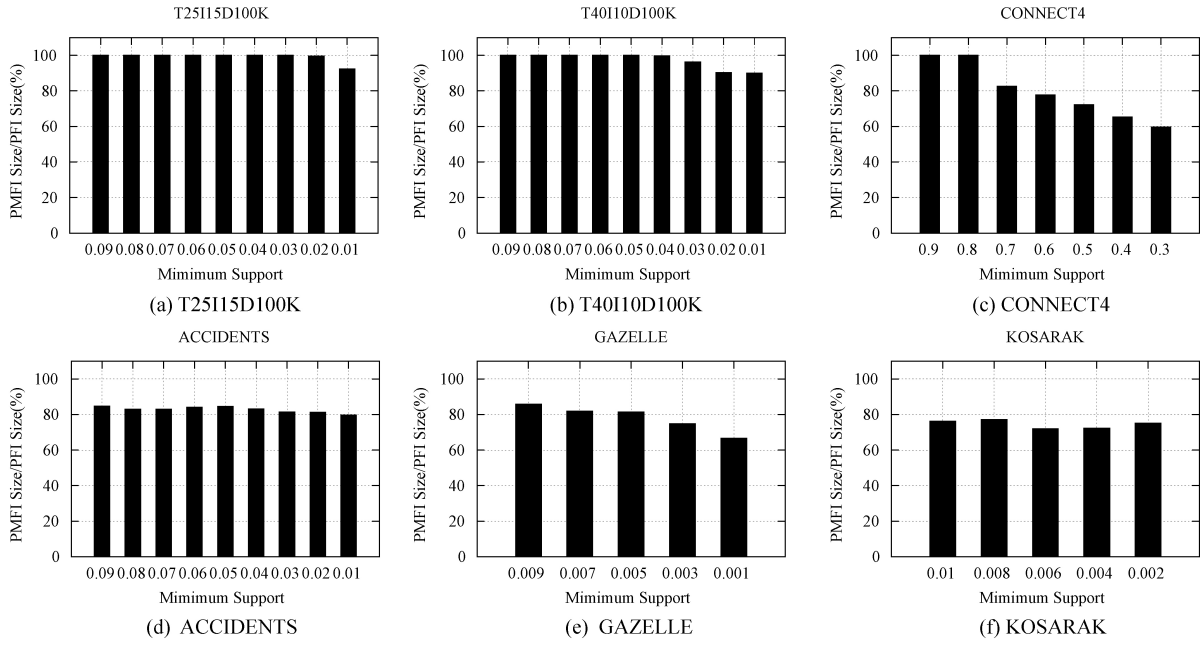


Fig. 10. Prob. maximal frequent itemsets size/prob. frequent itemsets size for $\tau = 0.9$.

$O(n)$, the running memory costs of *TODIS-MAX* and *PMFIM* algorithms were similar when the minimum support was high. Nevertheless, when the minimum support became lower, the memory usage of *TODIS-MAX* increased exponentially. Again, this is because of the massively generated infrequent itemsets. In addition, the lattice used in *TODIS-MAX* also leads to more memory usage. Note that our two algorithms *PMFIM* and *APMFIM* exhibited similar memory costs independent of the minimum support, and the *APMFIM* algorithm always used less memory than the *PMFIM* algorithm. This is because the *APMFIM* algorithm completely pruned the divide-and-conquer method; thus, it can absolutely reduced the space complexity to compute the probabilistic support of an itemset to $O(1)$.

6.4.3. Itemsets size analysis

To further shown the effectiveness of probabilistic maximal frequent itemsets, we presented the ratio between the probabilistic maximal frequent itemsets size and the probabilistic frequent itemsets size in Fig. 10. Over the densest dataset *Connect4*, the probabilistic maximal frequent itemsets were much decreased when the minimum support was low; also, an average of 25% reduction was maintained over the sparsest dataset *KOSARAK*. This can reduce both the runtime cost and memory usage, and enabling users to make easier decisions.

6.5. Precision and recall

Our algorithm *APMFIM* is an approximate algorithm that may result in both false positives and false negatives; that is, it may select the wrong itemsets or will miss certain real results because the value of probabilistic support is not accurate. We used *precision* and *recall* to represent the approximation of the mining results of our algorithm, which is defined as follows: For an actual collection Θ and a computed one Θ', the *precision* of Θ' is $\frac{\Theta \cap \Theta'}{\Theta'}$, and the *recall* of Θ' is $\frac{\Theta \cap \Theta'}{\Theta}$. A higher *precision* and a larger *recall* mean that Θ and Θ' are closer; when $\Theta = \Theta'$, both precision and recall equal 1.

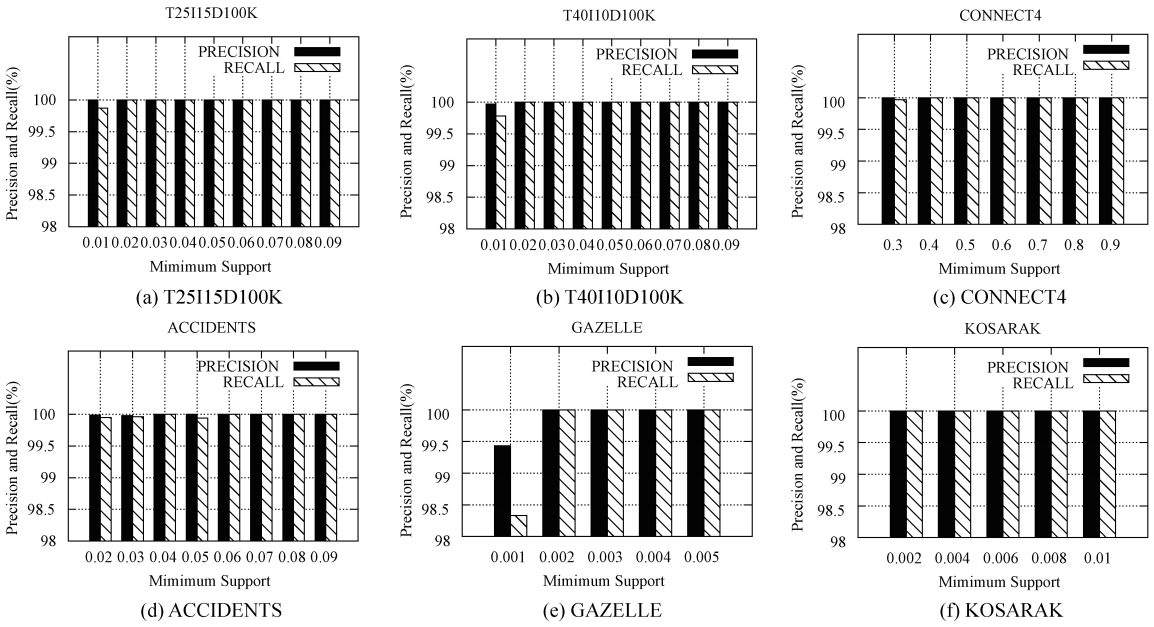


Fig. 11. Precision and recall of *APMFIM* for $\tau = 0.9$.

Figure 11 showed the *precision* and *recall* of the *APMFIM* algorithm over six datasets under different minimum supports. As can be seen, unless the minimum support is much lower, our algorithm can achieve 100% precision and 100% recall. Moreover, even though under low minimum support, our algorithm can also achieve $\geqslant$ 99.9% precision and $\geqslant$ 99.8% recall. The only exception was over the *GAZELLE* dataset when the relative minimum support was 0.001, but still, the precision and recall were $\geqslant$ 98%. These results verified that our algorithm was very effective and accurate.

6.6. Experimental results

From our experimental results, we can conclude the following: First, because computing the probabilistic support requires $O(nlog^2n)$ time complexity and $O(n)$ space complexity, our algorithms and the *TODIS-MAX* method have linear increasing runtime cost and memory usage with respect to data size. Second, our algorithms and the *TODIS-MAX* algorithm are insensitive to the minimum probabilistic confidence. Third, the three algorithms are sensitive to the minimum support; that is, the runtime cost, memory usage, and number of itemsets all increase with the decrease in minimum support. Finally, our approximate mining algorithm, *APMFIM*, is most efficient with a slight loss of accuracy. With the help of pruning strategies, our exact mining algorithm *PMFIM* also achieves significantly better performance than *TODIS-MAX*, the state-of-the-art algorithm, especially when the minimum support is low.

7. Conclusions

In this paper we studied the behavior of maximal frequent itemset mining over uncertain databases. We redefined the probabilistic maximal frequent itemset to make it more reasonable and more conducive to pruning. Based on this, the extended enumeration tree *PMFIT* was introduced as an efficient index

to maintain the probabilistic frequent itemsets. A bottom-up algorithm *PMFIM*, mining in a depth-first manner, was proposed, in which we used the support and the expected support to estimate whether an itemset was probabilistic frequent, which greatly reduced the probabilistic support computing. In addition, a superset pruning method was employed to further improve mining performance. Furthermore, we used the expected support to directly derive the value of probabilistic support, and then we developed an approximate method *APMFIM*, which yielded mining efficiency and lower memory usage. Our extensive experimental studies showed that our *PMFIM* algorithm significantly outperformed the *TODIS-MAX* method. Additionally, the *APMFIM* algorithm was more efficient in terms of computation and memory usage than the *PMFIM* algorithm and yielded higher accuracy.

Acknowledgments

This research is supported by the National Natural Science Foundation of China (61100112, 61309030), Beijing Higher Education Young Elite Teacher Project (YETP0987), National Key R&D Program of China under Grant (2017YFB1400701), Discipline Construction Foundation of Central University of Finance and Economics, CUFE Young Teacher Foundation (QJJ1706).

References

[1] J. Han, H. Cheng, D. Xin and X. Yan, Frequent pattern mining: current status and future directions, *in Data Mining and Knowledge Discovery* **17** (2007), 55–86.
[2] C.C. Aggarwal and P.S. Yu, A survey of uncertain data algorithms and applications, *in Transaction of Knowledge and Data Mining* **21**(5) (2009), 609–623.
[3] R.J. Bayardo, Efficiently Mining Long Patterns from Databases, in Proceedings of SIGMOD, 1998.
[4] N. Pasquier, Y. Bastide, R. Taouil and L. Lakhal, Discovering frequent closed itemsets for association rules, in Proceedings of ICDT, 1999.
[5] T. Calders and B. Goethals, Mining all non-derivable frequent itemsets, in Proceedings of PKDD, 2002.
[6] C.K. Chui, B. Kao and E. Hung, Mining Frequent Itemsets from Uncertain Data, in Proceedings of PAKDD, 2007.
[7] C.K. Chui and B. Kao, A Decremental Approach for Mining Frequent Itemsets from Uncertain Data, in Proceedings of PAKDD, 2008.
[8] C.K.S. Leung, M.A.F. Mateo and D.A. Brajczuk, A tree-based approach for frequent pattern mining from uncertain data, in Advances in Knowledge Discovery and Data Mining, 2008.
[9] C.C. Aggarwal, Y. Li, J. Wang and J. Wang, Frequent Pattern Mining with Uncertain Data, in Proceedings of KDD, 2009.
[10] C.K.S. Leung and S.K. Tanbeer, Fast Tree-based Mining of Frequent Itemsets from Uncertain Data, in Proceedings of DASFAA, 2012.
[11] C.K.S. Leung and R.K. MacKinnon, BLIMP: A Compact Tree Structure for Uncertain Frequent Pattern Mining, in DaWak, 2014.
[12] C.K.S. Leung and D.A. Brajczuk, Efficient algorithms for the mining of constrained frequent patterns from uncertain data, *in SIGKDD Explorer* **11**(2) (2009), 123–130.
[13] T. Calders, C. Garboni and B. Goethals, Efficient Pattern Mining of Uncertain Data with Sampling, in Proceedings of PAKDD, 2010.
[14] C.K.S. Leung and B. Hao, Mining of Frequent Itemsets from Streams of Uncertain Data, in Proceedings of ICDE, 2009.
[15] C.K.S. Leung and F. Jiang, Frequent Pattern Mining from Time-fading Streams of Uncertain Data, in Proceedings of DaWaK, 2011.
[16] C.K.S. Leung, A. Cuzzocrea and F. Jang, Discovering frequent patterns from uncertain data streams with time-fading and landmark models, *in Transactions on Large-Scale and Knowledge-Centered Systems* **7790** (2013), 174–196.
[17] C.K.S. Leung and Y. Hayduk, Mining frequent patterns from uncertain data with MapReduce for big data analytics, *in Database Systems for Advanced Applications* **7825** (2013), 440–455.
[18] Q. Zhang, F. Li and K. Yi, Finding Frequent Items in Probabilistic Data, in Proceedings of SIGMOD, 2008.
[19] T. Bernecker, H.P. Kriegel, M. Renz, F. Verhein and A. Zuefle, Probabilistic Frequent Itemset Mining in Uncertain Databases, in Proceedings of SIGKDD, 2009.

[20] L. Sun, R. Cheng, D.W. Cheung and J. Cheng, Mining Uncertain Data with Probabilistic Guarantees, in Proceedings of KDD, 2010.
[21] T. Bernecker, H.P. Kriegel, M. Renz, F. Verhein and A. Zuefle, Probabilistic frequent Pattern Growth for Itemset Mining in Uncertain Databases, in Proceedings of SSDM, 2012.
[22] L. Wang, R. Cheng, S.D. Lee and D. Cheung, Accelerating Probabilistic Frequent Itemset Mining: a Model-based Approach, in Proceedings of CIKM, 2010.
[23] L. Wang, D. Cheung, R. Cheng, S.D. Lee and X.S. Yang, Efficient mining of frequent item sets on large uncertain databases, *in Transaction of Knowledge and Data Mining* **24**(12) (2012), 2170–2183.
[24] T. Calders, C. Garboni and B. Goethals. Approximation of Frequentness Probability of Itemsets in Uncertain Data, in Proceedings of ICDM, 2010.
[25] Y. Tong, L. Chen, Y. Cheng and P.S. Yu, Mining Frequent Itemsets over Uncertain Databases, in Proceedings of VLDB, 2012.
[26] P. Tang and E.A. Peterson, Mining Probabilistic Frequent Closed itemsets in Uncertain Databases, in Proceedings of ACMSE, 2011.
[27] E.A. Peterson and P. Tang, Fast approximation of probabilistic frequent closed itemsets, in Proceedings of ACMSE, 2012.
[28] Y. Tong, L. Chen and B. Ding, Discovering Threshold-based Frequent Closed Itemsets over Probabilistic Data, in Proceedings of ICDE, 2012.
[29] C. Liu, L. Chen and C. Zhang, Mining Probabilistic Representative Frequent Patterns From Uncertain Data, in Proceedings of SDM, 2013.
[30] C. Liu, L. Chen and C. Zhang, Summarizing Probabilistic Frequent Patterns: A Fast Approach, in Proceedings of KDD, 2013.
[31] Y.H. Liu, Mining frequent patterns from univariate uncertain data, *in DKE* **71**(Issue 1) (2012), 47–68.
[32] B. Pei, S. Zhao, H. Chen, X. Zhou and D. Chen, FARP: mining fuzzy association rules from a probabilistic quantitative database, *in Information Sciences* **237** (2013), 242–260.