An algorithm for mining frequent itemsets on uncertain dataset

Si Tian[1], Shui Wang[1*], Yang Liu[2], Le Wang[1]

[1]*School of Information Engineering, Ningbo Dahongying University, Ningbo, Zhejiang, China, 315175*

[2]*School of Innovation Experiment, Dalian University of Technology, Liaoning, China, 116024*

Received 1 March 2014, www.cmnt.lv

Abstract

Mining frequent itemsets from uncertain transaction dataset is a research topic in data mining. Some algorithms are based on FP-Growth, but they construct the tree structure in a manner that cannot be as compact as the original FP-Tree, so the tree is easily developed to huge size and this hinders their performance. In this paper, we propose a new tree structure called IT-Tree (Itemset Tail-node Tree) to efficiently maintain probability information of itemsets in tail-nodes; we also propose a corresponding algorithm IT-Mine to mine frequent itemsets from IT-Tree without additional dataset scans. We evaluate our approach on real sparse and dense datasets with different minimum support numbers that can produce non-null frequent k-itemsets ($k \geq 2$); the results show that IT-Mine outperforms other algorithms in terms of execution time, especially for large dataset or small minimum expected support number.

Keywords: frequent itemset, frequent pattern, uncertain transaction dataset, data mining

1 Introduction

Uncertain transaction dataset describes the existential probability of each item in a transactional process. Many applications create uncertain datasets; for example, as indicated by [1,2] and [3], measurement errors of RFID, GPS and other sensors are part of the major sources of uncertain data, because the sensor readings are constantly fluctuating and can hardly be precise, such as the location of an object provided by RFID or GPS. Another kind of uncertainty comes from statistical laws; for example, in medical field, the illness or disease diagnosed for a patient cannot be completely determined by one or more symptoms; in market analysis, customer purchase behaviours, computed from basket data for predicting what a customer will buy in the future, are also statistical probabilities [4, 5]. With the development of applications using uncertain datasets, the issue of data mining over uncertain dataset has become a hot topic in data mining in recent years [4-12].

Table 1 shows an example of an uncertain transaction dataset. Each transaction in Table 1 represents that a customer buy a certain item with a probability. The decimal value associated with an item is called the existential probability of the item. For instance, the first transaction T_1 in Table 1 shows that the customer *A* might purchase products "a", "b", "c" and "d" with 60%, 20%, 30% and 50% chances in the future, respectively. The probability values in Table 1 may be obtained from the analysis of the customers' browsing online-shop history: if customer *A* visited an online-shop ten times in a certain period of time, out of which "a" product was clicked six times, then it might be established that customer *A* has a 60% probability to buy "a" in the future. The probability values may also be obtained from data mining results on the supermarket basket data.

TABLE 1 An example of uncertain transaction dataset

TID	Customer	Transaction itemset
T_1	*A*	(a: 0.6), (b: 0.2), (c: 0.3), (d,0.5)
T_2	*B*	(a: 0.7), (e: 0.25), (d,0.8)
T_3	*C*	(a: 0.3), (c: 0.8) ,(d,0.4)
T_4	*D*	(c: 0.7), (e: 0.2), (d,0.3)
T_5	*E*	(a: 0.5), (b: 0.3), (e: 0.3)

Mining frequent itemsets from uncertain transaction dataset is to discover those itemsets whose sum of probability values or occurring probability exceeds the user specified threshold. Because of its probablistic nature, frequent itemsets mining on uncertain transaction dataset is different from that on precise dataset, which has already been well defined and studied [13, 14], and many algorithms have been proposed, such as Apriori [13], FP-Growth [14], MAFIA [15], COFI [16], Pincer-search [17], CHARM [18], Index-BitTableFI [19] etc. Researchers usually extrapolate the existing algorithms on precise data to get their new algorithms on uncertain data. The papers [5, 7, 10-12] proposed algorithms based on Apriori. These algorithms bear the same bottleneck as Apriori: the generating & processing of the candidate itemsets. And with the increasing of the number of long transactions and the decreasing of the minimum expected support, their performance deteriorates rapidly. The algorithms proposed in papers [6,8,9] are based on FP-Growth; in building their UF-Tree, two items with the same name but different existential probabilities are considered as different nodes; this approach leads to excessive memory requirement to maintain the tree.

[*] ***Corresponding author's*** e-mail: seawan@163.com

Our approach is also inspired by FP-Growth, and tries to amend the defects of the above tree-based algorithms: we propose a more efficient tree structure, named IT-Tree (Itemset Tail-node Tree), to maintain the probability information of transaction itemsets, and give an algorithm, named IT-Mine, to mine frequent itemsets from the IT-Tree. An IT-Tree is created by two scans of the dataset, and IT-Mine mines frequent itemsets from the IT-Tree without additional scan of the dataset, and without generating candidate itemsets.

1.1 CONTRIBUTIONS OF THIS PAPER

1) A tree structure named IT-Tree is proposed for maintaining transaction itemsets of an uncertain dataset.
2) An algorithm named IT-Mine is proposed for discovering frequent itemsets from uncertain dataset based on IT-Tree.
3) Both real and synthetic datasets are used in our experiments to evaluate the performance of the proposed algorithm with the state-of-the-art algorithm MBP.

1.2 THE CONSTRUCTION OF THIS PAPER

The rest of this paper is organized as follows: Section 2 is the description of the problem and definitions; Section 3 is related works; Section 4 is our proposed algorithm IT-Mine; Section 5 is the experimental results; and Section 6 is the conclusion and discussion.

2 Problem definitions

An uncertain transaction dataset $D=\{T_1,T_2,\ldots,T_n\}$ contains n transaction itemsets and m distinct items i.e. $I=\{i_1,i_2,\ldots,i_m\}$, and each transaction itemset T_d ($1\le d\le n$) is represented as $\{i_1:p_1, i_2:p_2,\ldots, i_v:p_v\}$ or $\{\{i_1,i_2,\ldots,i_v\},\{p_1,p_2,\ldots,p_v\}\}$, where $\{i_1,i_2,\ldots,i_v\}$ is a subset of I, and p_u is the probability of the item i_u ($1\le u\le v$) in transaction itemset T_d. An itemset $X=\{i_1,i_2,\ldots,i_k\}$ is called a k-itemset, and k is the length of the itemset X.

We adopt definitions similar to those presented in the previous works [5, 11, 13].

Definition 1: According to the paper [13], the *support number* (*sn*) of an itemset X is the number of transaction itemsets containing X.

Definition 2: The probability of an item i_r in transaction itemset T_d is denoted as $p(i_r,T_d)$, and is defined by

$$p(i_r,T_d)=p_r. \tag{1}$$

For example, in Table 1, $p(\{a\},T_1)=0.6$, $p(\{d\},T_1)=0.5$.

Definition 3: The probability of an itemset X in a transaction itemset T_d is denoted as $p(X,T_d)$, and is defined by

$$p(X,T_d)=\prod_{i_r\in X, X\subset T_d} p(i_r,T_d). \tag{2}$$

For example, in Table 1, $p(\{a,d\},T_1)=0.6\times0.5=0.3$, $p(\{a, d\},T_2)=0.7\times0.8=0.56$, $p(\{a, d\},T_3)=0.3\times0.4=0.12$.

Definition 4: The *expected support number* (*exp*) of an itemset X in an uncertain transaction dataset is denoted as $E(X)$, and is defined by

$$E(X)=\sum_{T_d\in D} P(X,T_d). \tag{3}$$

For example, in Table 1, $E(\{a, d\})=p(\{a, d\},T_1)+p(\{a, d\},T_2)+p(\{a, d\},T_3)=0.3+0.56+0.12=0.98$.

Definition 5: According to the probability theory, the occurring probability of the itemset X occurring in k mutually independent transaction itemsets ($0\le k\le|D|$) is denoted as $P_k(X)$, and is defined by

$$P_k(X)=\sum_{S\subseteq D,|S|=k}\left(\prod_{T_d\in S} P(X,T_d)\cdot\prod_{T_d\in D-S}(1-P(X,T_d))\right). \tag{4}$$

For example, in Table 1,

$$\begin{aligned}P_2(\{a,d\})=&p(\{a,d\},T_1)\times p(\{a,d\},T_2)\times\\&(1-p(\{a,d\},T_3))+p(\{a,d\},T_1)\times p(\{a,d\},T_3)\times\\&(1-p(\{a,d\},T_2))+p(\{a,d\},T_2)\times p(\{a,d\},T_3)\times\\&(1-p(\{a,d\},T_1))=0.21072;\end{aligned}$$

$$P_3(\{a,d\})=p(\{a,d\},T_1)\times p(\{a,d\},T_2)\times p(\{a,d\},T_3)=0.02016.$$

Definition 6: According to the probability theory, the occurring probability of the itemset X occurring in more than k mutually independent transaction itemsets ($0\le k\le|D|$) is denoted as $P_{\ge k}(X)$, and is defined by

$$P_{\ge k}(X)=\sum_{S\subseteq D,|S|\ge k}\left(\prod_{T_d\in S} P(X,T_d)\cdot\prod_{T_d\in D-S}(1-P(X,T_d))\right). \tag{5}$$

For example, in Table 1,

$$P_{\ge2}(\{a,d\})=P_2(\{a,d\})+P_3(\{a,d\})+P_{\ge4}(\{a,d\})=0.21072+0.02016+0=0.23088.$$

Note $P_{\ge4}(\{a,d\})$ is 0 because there is only 3 transaction itemsets containing the itemset $\{a,d\}$.

3 Related work

The approaches of finding frequent itemsets from precise dataset can be classified into two categories: the level-wise approach and the pattern-growth approach. The algorithms Apriori [13] and FP-Growth [14] are representative ones for mining frequent itemsets from precise transaction dataset, and they are representative ones for the level-wise approach and pattern-growth approach, respectively. Apriori is to iteratively generate frequent (k+1)-itemsets using frequent k-itemsets ($k\ge1$):

1) a (k+1)-itemset X is a candidate itemset if its k sub k-itemsets are frequent itemsets;

2) the support number of X is calculated by one scan of dataset if X is a candidate itemset;

3) the itemset X is frequent if its support number is not less than the specified minimum support number.

One advantage is that it has a high time performance when the dataset is sparse, does not contain many long transaction itemsets, and the minimum support number is not small. Its main shortcoming is that it requires multiple scans of dataset, and generating candidate itemsets. Pattern-growth also employs the iteration approach:

1) it finds the set of frequent 1-itemsets (the set is denoted as F) under the condition of a k-itemset X ($k \geq 1$),

2) any itemset $X \cup f$ ($f \in F$) is a frequent (k+1)-itemset. It maintains all transaction itemsets on a tree by one scan of dataset, and generates a conditional sub-tree for each frequent itemset X. Thus, it will find all frequent itemsets under the condition of X by scanning this conditional sub-tree, instead of scanning the whole dataset.

An important difference between precise and uncertain transaction dataset is that each transaction itemset of the former only contains items, and that of the latter contains items and their existential probabilities. Thus, the existing algorithms of mining frequent itemsets from precise dataset cannot be used directly on uncertain transaction dataset. Recently, some algorithms have been proposed for mining frequent itemsets from uncertain transaction dataset.

U-Apriori [7] was proposed in 2007 to find frequent itemsets from uncertain transaction dataset, and it was a level-wise approach. The difference of U-Apriori and Apriori is that U-Apriori calculates the sum of probability of a candidate itemset/item X in all transaction itemsets while Apriori calculates the number of transaction itemsets containing X when they scan a dataset to judge whether the candidate itemset/item X is frequent. U-Apriori and Apriori have the same advantage and disadvantage. The time and memory performance may be worse with the increasing of the number of long transaction itemsets and the decreasing of the minimum expected support number.

In 2007, Leung et al. [8] propose a tree-based algorithm, named UF-Growth, for mining frequent itemsets from uncertain transaction dataset using the same definition of frequent itemsets as in paper [7]. UF-Growth is based on FP-Growth, and is a pattern-growth approach. UF-Growth also constructs a UF-Tree using the given uncertain transaction dataset. But, if two items have the same item name but different existential probabilities, they are considered as different items, and they cannot share the same node when they are added to a tree. For example, two sorted itemsets {a:0.90, b:0.70, c:0.73} and {a:0.95, b:0.85, c:0.70}, they will not share the same node "a" because the probabilities of item "a" in two itemsets are different. After the first UF-Tree is built, UF-Growth retrieves the frequent itemsets from the UF-Tree or sub UF-Trees recursively as the method of FP-Growth. However, the UF-Growth algorithm requires a lot of memory to store tree nodes and a large amount of computational time to process tree nodes.

In 2008, Leung et al. [9] proposed two improvements to boost the time and memory performance of UF-Growth. The first improved algorithm uses the idea of the co-occurrence frequent itemset tree [16] to avoid creating of sub UF-Trees. The second improved algorithm considers that the items, which have the same ***k***-digit value after the decimal point, have the same probability. For example, for two sorted itemsets {a:0.90, b:0.70, c:0.73} and {a:0.95, b:0.85, c:0.70}, both probabilities of item "a" are 0.9 and they will share the node "a". When they are inserted to a UF-Tree if k is set as 1; both probabilities of item "a" are 0.90 and 0.95 respectively and they will not share the node "a" if k is set as 2. This improved algorithm has a better performance than its original algorithm UF-Growth. However, the improved algorithm still does not build a UFP-Tree as compact as the original FP-Tree [14], and it may loss some frequent itemsets.

In 2009, Aggarwal et al. [6] proposed the two algorithms UH-mine and UFP-Growth respectively. These two algorithms employ the pattern-growth approach. Aggarwal also performed a comparison on three frequent itemsets mining algorithms U-Apriori, UH-Mine and UFP-Growth, and concluded that U-Apriori outperforms the other two algorithms.

In the papers [6-9], the proposed algorithms are based on the expected support number of an itemset. An itemset, whose *expected support number* is not less than the *user specified minimum expected support number*, is called as a frequent itemset.

In 2008, Zhang et al. [12] proposed an approximate algorithm for mining frequent itemsets from uncertain transaction dataset based on Definition 6, which defines an itemset to be a frequent itemset if its occurring probability is not less than the *user specified minimum probability*.

In 2009, Bernecker et al. [5] developed a dynamic-programming-based algorithm based on Definition 6, to mine frequent itemsets from uncertain transaction dataset, which employs the level-wise approach. This algorithm inherits the advantage and disadvantage of Apriori.

In 2010, Sun et al. [10] proposed algorithms p-Apriori and TODIS to mine frequent itemsets from uncertain transaction dataset. The algorithm p-Apriori is based on Apriori, and applies a divide-and-conquer approach in calculating the occurring probability of an itemset. The difference between TODIS and p-Apriori is that TODIS works in a top-down manner and p-Apriori works in a bottom-up manner. However calculating the occurring probability of an itemset requires a large amount computation, the time performance of the algorithms proposed in the papers [5, 10, 12] is low.

The problem of getting the occurring probability of an itemset is changed to the problem of getting the expected support number of an itemset in the paper [11], because a Poisson binomial distribution can be well approximated by a Poisson distribution [20] and calculating the occurring probability of an itemset requires a large amount computation. The paper [11] also proposed an algorithm MPB based on U-Apriori. MPB can fast identify frequent and non-frequent itemsets from candidate itemsets without scanning the whole dataset; it can achieve a better performance than U-Apriori in terms of time and space. In

this paper, we propose a mining algorithm based on the expected support number of an itemset.

4 Our proposed method

Our proposed algorithm IT-Mine mainly includes two procedures: firstly, create an IT-Tree or sub IT-Tree; secondly, mine frequent itemsets from the IT-Tree or sub IT-Tree. We descript the structure of IT-Tree in Section 4.1, give an example of the construction of IT-Tree in Section 4.2, and describe the process of mining frequent itemsets from the tree with an example in Section 4.3.

4.1 STRUCTURE OF IT-TREE

Definition 7: Let X be a sorted k-itemset $\{i_1, i_2, i_3, \ldots, i_k\}$, where i_k is the *tail-item*. When the itemset X is inserted into a tree T in this order, the node N on the tree that represents this *tail-item* is defined as a *tail-node* for itemset X. The itemset X is called *tail-node-itemset* for node N.

Definition 8: Let an itemset X containing itemset Y be added to a conditional sub-tree T of the itemset Y. On the tree T, the *base probability* of an itemset X is denoted as $BP(X,Y)$, and is defined by

$$BP(Y, X) = P(Y, X). \quad (6)$$

The structure of IT-tree is illustrated in Figure 1. There are two types of nodes on the IT-tree: one is normal node, as shown in Figure 1a, where N records the item name of each node, *Parent* records the parent node, *Children list* records all children nodes; and the other is tail-node, as shown in Figure 1b, where S records support number, *BP* is *base probability* value of a transaction itemsets, and *ItemsP* is an array which records probability values of all items in corresponding *tail-node-itemset*. We use a "list" to maintain *BP* and *ItemsP values* because there may be many itemsets sharing the same tail-node.

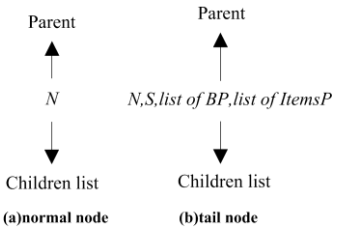

FIGURE 1 Structure of nodes on IT-Tree

4.2 CONSTRUCTION OF AN IT-TREE

The construction of an IT-Tree needs two scans of dataset. In the first scan, create a header table to maintain the support number and expected support of each item; delete those items whose expected support number is less than the user specified *minimum expected support number* from the header table; arrange remaining items of the header table in descending order of support numbers. In the second scan, all transaction itemsets are inserted into an IT-tree. The process is as follows:

1) Delete items that are not in the header table from the transaction itemset;

2) Sort remaining items in the transaction itemset according to the order of the header table;

3) Insert the modified transaction itemset into an IT-tree, and store support number and probability value of each item in each modified transaction itemset to the tail-node of the itemset.

To facilitate tree traversals, *links* in the header table are also maintained (not shown in the Figure 2 for simplicity).

For example, consider the transaction dataset in Table 1. Here the user specified *minimum expected support number* is set to 0.8.

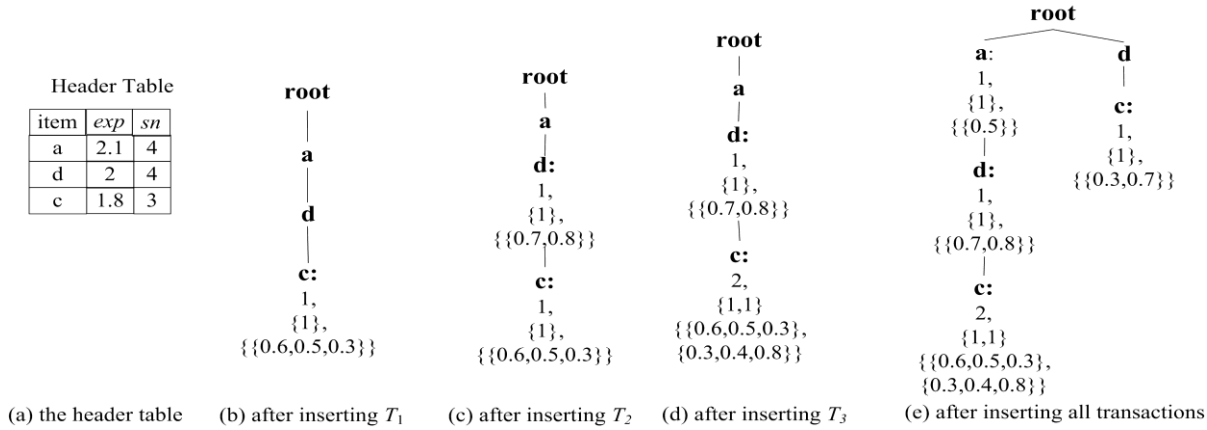

FIGURE 2 Construction of an IT-Tree

Figure 2a is a header table created in the first scan of dataset. Initially, the IT-Tree is created with a root R. When the transaction itemset T_1 is retrieved, the algorithm IT-Mine firstly deletes the item "b"; and then arranges the itemset in the order as the header table in Figure 2a; last, inserts the sorted itemset into the IT-tree. The resulting IT-Tree is shown in Figure 2b. On tail-node "c", "1" represents the support number of this transaction itemset, "{1}" represents the *base probability*, "{0.6,0.5,0.3}"

represents the probability values of items "a", "d" and "c" respectively.

Now insert the transaction itemset T_2 into the IT-Tree. Since the path "root-a-d" can be shared, the algorithm IT-Mine changes the node "d" to a tail-node for the transaction itemsets T_2. The resulting IT-Tree is shown in Figure 2c.

Afterwards, insert the transaction itemsets T_3 into the IT-Tree. Since the path "root-a-d-c" can be shared, only the probability information of this itemsets needs to be stored into tail node "c". The resulting IT-Tree is shown in Figure 2d. On the tail node "c", "{1,1}" represents the base probability values of two transaction itemsets, "{{0.6,0.5,0.3},{0.3,0.4,0.8}}" represents the probability values of items "a", "d" and "c" in two transaction itemsets respectively.

Figure 2e is the result after adding all transaction itemsets in Table 1 are added to the tree; this is the first IT-Tree, and we also call it the *global* IT-Tree. Since the conditional *base-itemset* of the first IT-Tree is null, the base probability of any tail node on the first IT-Tree is set to 1.

4.3 MINING FREQUENT ITEMSETS FROM THE TREE

Start processing from the last item (denoted as *Z*) in the header table (denoted as *HT*):

Step 1. Add item *Z* to the current *base-itemset* (which is initialized as null). Each new *base-itemset* is a frequent itemset.

Step 2. Let *Z.links* contain *k* nodes whose item name is *Z*; we denote these *k* nodes as $N_1,N_2,\ldots,N_k$ respectively; because item *Z* is in the last of the header table, all these *k* nodes are *tail nodes*, i.e., each of these nodes contains *probability information* (*S*, list of *BP*, list of *ItemsP*).

Substep 2.1. Since each one of these *k* nodes contains the probability of each item, create a sub header table through scanning these *k* branches whose paths are from these *k* nodes to the root.

Substep 2.2. Go to Step 3 if the sub header table is null.

Substep 2.3. Create a sub IT-tree for the current *base-itemset* {*Z*} according to the sub header table and these *k* branches. The probability values of items in the current *base-itemset* are stored to the corresponding tail-nodes.

Substep 2.4. Perform a recursive mining process on this sub IT-Tree.

Step 3. Remove the item *Z* from the *base-itemset*.

Step 4. For each of the *k* nodes (which we denote as N_i, $1\le i\le k$), modify its *tail-information*:

Substep 4.1. Delete item *Z*'s probability from list of *BP* of these *k* nodes.

Substep 4.2. Move the *probability information* (*S*, list of *BP*, list of *ItemsP*) on each node to the parent of each node.

Step 5. Process the next item in the header table *HT* using the same method.

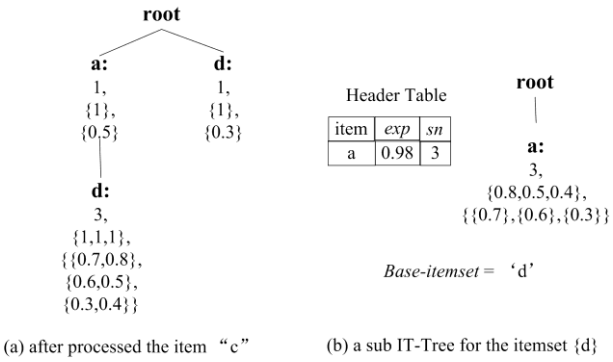

FIGURE 3 Mining the frequent itemsets

The following is the detailed explanation of the mining process illustrated in Figure 2 and Figure 3.

In Figure 2a, item "c" is the last item in the header table. Firstly, add item "c" to the *base-itemset*, and get a frequent itemset {c}; then create a header table for the current *base-itemset* by scanning the branches "root-a-d-c" and "root-d-c". For branch "root-a-d-c", under the condition of the current *base-itemset* {c}, the probability values of items "a" and "d" are calculated as 0.42 (=0.3*0.6+0.8*0.3) and 0.47 (=0.3*0.5+0.8*0.4); for branch "root-d-c", under the condition of the current *base-itemset* {c}, the probability value of item "d" is calculated as 0.21 (=0.7*0.3). The expected support numbers of items "a" and "d", 0.42 and 0.68 (=0.47+0.21), are less than the user specified *minimum expected support number* 0.8. So remove item "c" from the base-itemset, and remove the probability of each node "c" from *ItemsP* of each node, then pass the modified probability information to the parent node of this node. The resulting IT-Tree is shown in Figure 3a.

Afterwards, begin processing the item "d" in the header table in Figure 2a.

Add item "d" to the current *base-itemset*, we get a frequent itemset {d}; then create a header table for the current *base-itemset* {d} by scanning the branch "root-a-d" in Figure 3a; as the result, under the condition of the current *base-itemset* {d}, the expected support number of item "a" is calculated as 0.98 (= 0.5 * 0.6 + 0.4 * 0.3 + 0.8 * 0.7). Since this expected support number is not less than *minimum expected support number* 0.8, create a conditional sub IT-Tree and a sub header table for the current *base-itemset* {d}, as shown in Figure 3b. On tail-node "a" of Figure 3b, "{0.8,0.5,0.4}" represents the base probability values of 3 transaction itemsets containing the itemset {da}, "{{0.7},{0.6},{0.3}}" represents the probability value of the item "a" in 3 transaction itemsets respectively.

Now, the item "a" in header table in Figure 3b is processed.

Add item "a" to the current *base-itemset* {d}, and get a frequent itemset {da}. After the sub IT-tree in Figure 3b is processed, return to process previous tree in Figure 3a.

Go on processing the next item "a" of the header table in Figure 2a.

After finish processing item "a", we get a new frequent itemset {a}.

Lastly, we find 4 frequent itemsets from the dataset in Table 1: {c}, {d}, {da} and {a}.

5 Experimental results

In this section, we compare the performance of the proposed algorithm IT-Mine, the level-wise algorithm MBP and the pattern-growth algorithm UF-Growth using five datasets.

TABLE 2 Dataset characteristics

Dataset	|D|	|I|	ML	DS (%)	Type
retail	88162	16470	10.3	0.06	sparse
T20I6D100K	100000	980	20	2.03	sparse
mushroom	8124	119	23	19.33%	dense
connect	67557	129	43	33.33	dense
T20I6D200K	200000	980	20	2.03	sparse

Table 2 shows the characteristics of 5 datasets used in our experiments. "|D|" represents the number of transactions; "|I|" represents the number of distinct items; "ML" represents the mean length of all transaction itemsets; "DS" represents the degree of sparse or dense. The datasets *retail*, *mushroom* and *connect* are real-world and obtained from FIMI Repository [21]. The datasets *T20I6D100K* and *T20I6D200K* came from the IBM Data Generator [13**]**. Because the original datasets do not provide probability values for each item, to use these dataset as uncertain transaction dataset, we assign a randomly generated existential probability from range (0, 1] to each item of each transaction itemset. The runnable programs and testing datasets used in our experiments can be downloaded from the following address: http://code.google.com/p/it-tree/downloads/list.

The configuration of the testing platform is as follows: Windows 7 operating system, 3G memory, Intel(R) Core(TM) i5-2300 CPU @ 2.80 GHz; Java heap size is 1024M.

The minimum expected support number is set from 130 down to 40 with the decreasing of 10 on *retail*, from 160 down to 70 with the decreasing of 10 on *T20I6D100K*, from 550 down to 100 with the decreasing of 50 on *mushroom*, and is set from 9000 down to 7200 with the decreasing of 200 on *connect*. The more the number of frequent itemsets is, the smaller the minimum expected support number is set. Under the upper bound of the minimum expected number on our experiments, there are at least frequent 2-itemsets on those datasets. As shown in Figures 4-7, they show the distribution of frequent itemsets (FIs) under different minimum expected support number.

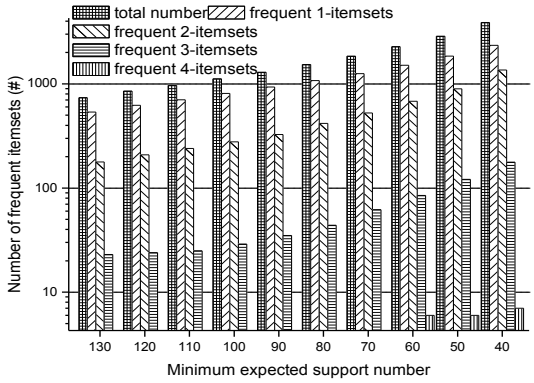

FIGURE 4 Distribution of FIs of retail

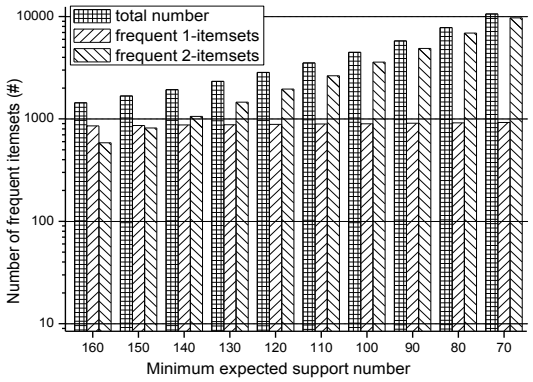

FIGURE 5 Distribution of FIs of *T20I6D100K*

TABLE 3 Number of candidate itemsets under varied minimum expected support number

retail		T20I6D100K		mushroom		connect	
Min_exp	*Candidate itemsets (#)*	*Min_exp*	*Candidate itemsets (#)*	*Min_exp*	*Candidate itemsets (#)*	*Min_exp*	*Candidate itemsets (#)*
130	143524	160	367899	550	2002	9000	7758
120	193308	150	375253	500	2403	8800	7829
110	246261	140	384624	450	2777	8600	8078
100	327889	130	392468	400	3533	8400	8524
90	435082	120	404535	350	4589	8200	9496
80	576580	110	417895	300	5692	8000	10761
70	786105	100	436599	250	7860	7800	11986
60	1142949	90	467162	200	12275	7600	13524
50	1713089	80	516153	150	19595	7400	15263
40	2756787	70	596506	100	40061	7200	17221

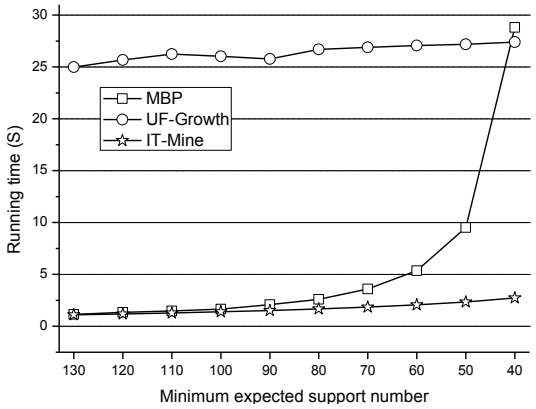

FIGURE 8 Running time on retail

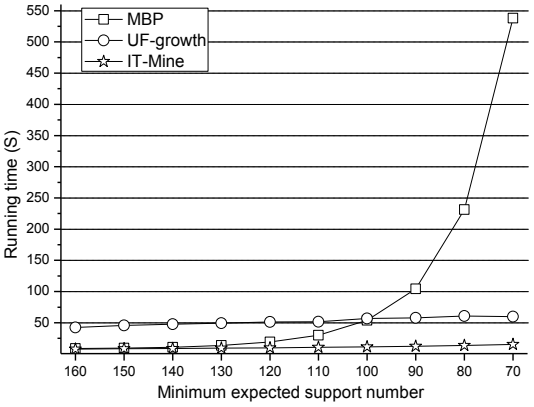

FIGURE 9 Running time on T20I6D100K

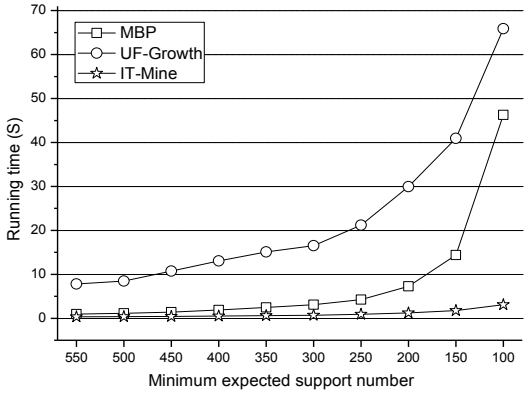

FIGURE 10 Ru3nning time on mushroom

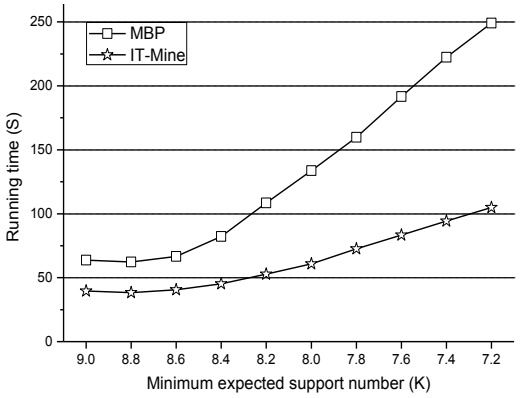

FIGURE 11 Running time on connect

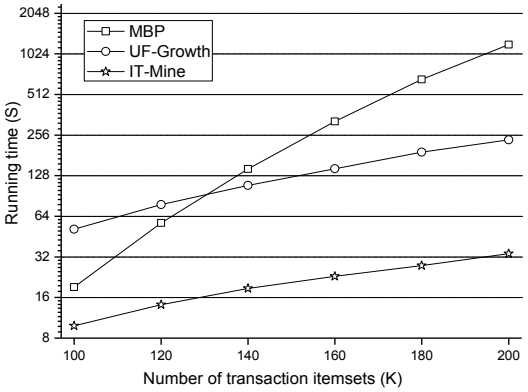

FIGURE 12 Scalability for the algorithms on the sparse dataset

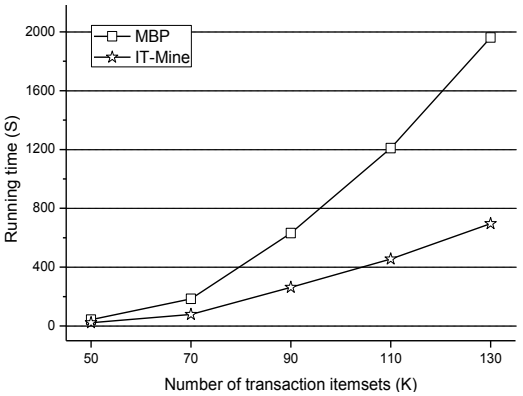

FIGURE 13 Scalability for the algorithms on the dense dataset

Figures 8-11 show the running time of 3 algorithms on four testing datasets under different minimum expected support number. On the dense dataset *connect*, UF-Growth is out of memory although the minimum expected support number is set 9000 because the dataset connect is very dense and the length of each transaction itemset is 43. The time performance of UF-Growth is low because UF-Growth generates too many tree nodes and requires too much time to process these tree nodes. The number of candidate itemsets generated by MBP increases with the decreasing of the minimum expected support number, as shown in Table 3. For example, the number of candidate itemsets is up to 2756787 from 143524 when the minimum expected support number is down to 40 from 130 on *retail*, thus the running time of MBP increases quickly with the decreasing of the minimum expected support number. Figures 8-11 shows that IT-Mine has a high time performance on those four dataset under different minimum expected support number.

In the following experiments, we evaluate the scalability of IT-Mine, UF-Growth and MBP on sparse datasets using the synthetic sparse dataset T20I6D200K; we vary the size of the dense dataset *connect* to evaluate the scalability of IT-Mine, UF-Growth and MBP on dense

datasets; The minimum expected support number is set 120 and 8000 on the sparse dataset and dense dataset respectively.

MBP requires generating candidates and identifying frequent itemsets from the candidates by scanning the dataset, thus its time performance is dependent on the numbers of candidates, the size of dataset and the length of transaction itemsets. As shown in Figure 12 and Figure 13, the time performance of MBP slows down drastically with the increasing of number of transaction itemsets. Both IT-Mine and UF-Growth have a stable time performance on the sparse dataset. On the dense dataset, because the length of each transaction itemset is long and the dataset is very dense, UF-Growth is out of memory in our experiment of dense dataset and the time performance of MBP also slows down with the increasing of the size of dataset. Figure 12 and Figure 13 show that the strong scalability of IT-Mine is obvious.

6 Conclusion and discussion

In this paper, we propose a novel tree structure, named IT-Tree, to represent transaction itemsets of an uncertain transaction dataset. An important feature of IT-Tree is that it maintains probability information of each transaction itemset into a tail-node of the transaction itemset. This way the IT-Tree is as compact as the original FP-Tree, while it does not lose information with respect to the distinct probability values for transaction itemsets. We also give an algorithm named IT-Mine to mine frequent itemsets from IT-Tree without additional scans of dataset.

Experiments were performed on both real sparse and dense dataset, and IT-Mine outperforms the algorithm MBP and UF-Growth in terms of running time; the performance of IT-Mine is also quite stable with the changing of the minimum expected support number.

There are two models in mining frequent itemsets from uncertain transaction dataset: one is based on the *expected support number* of an itemset; the other is based on *occurring probability* of an itemset. IT-Mine is developed on *expected support number*, but it can also be applied to the latter model, because after getting probability information from the tree, the data structure itself is irrelevant with the calculation of the *expected support number* or the *occurring probability* of an itemset.

Although IT-Mine has achieved better performance on testing dataset, construction of global IT-Tree and conditional IT-Trees consumes much time. We hope to improve the speed of IT-Tree construction in future work.

Acknowledgements

This work is partially supported by National Natural Science Foundation of P. R. China (Grant No. 61173163, 51105052, 2013A610115), Ningbo Natural Science Foundation (2013A610115, 2014A610073, 2014A610020), and General Scientific Research Fund of Zhejiang Provincial Education Department (Y201432717).

References

[1] Sistla A Prasad 2003 Querying the uncertain position of moving objects *Proceedings of Seminar Temporal Databases: Research and Practice Dagstuhl Germany*
[2] Khoussainova N, Balazinska M, Suciu D 2006 Towards correcting input data errors probabilistically using integrity constraints *MobiDE 2006: 5th ACM International Workshop on Data Engineering for Wireless and Mobile Access Chicago IL United states*
[3] Hung C, Peng W 2010 Model-driven traffic data acquisition in vehicular sensor networks *39th International Conference on Parallel Processing, ICPP San Diego CA United states*
[4] Aggarwal C C, Yu P S 2009 *Knowledge and Data Engineering IEEE Transactions* **21**(5) 609-23
[5] Bernecker T 2009 Probabilistic frequent itemset mining in uncertain databases *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining Paris France*
[6] Aggarwal C C 2009 Frequent pattern mining with uncertain data *15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining KDD Paris France*
[7] Chui C, Kao B, Hung E 2007 Mining frequent itemsets from uncertain data *11th Pacific-Asia Conference on Knowledge Discovery and Data Mining PAKDD Nanjing China*
[8] Leung C K, Carmichael C L, Hao B 2007 Efficient mining of frequent patterns from uncertain data *17th IEEE International Conference on Data Mining Workshops, ICDM Workshops Omaha NE United states*
[9] Leung CK, Mateo M A F, Brajczuk D A 2008 A tree-based approach for frequent pattern mining from uncertain data *12th Pacific-Asia Conference on Knowledge Discovery and Data Mining PAKDD Osaka Japan*
[10] Sun L 2010 Mining uncertain data with probabilistic guarantees *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining Washington DC United states*
[11] Wang L, Cheung, D W-L, Cheng R, Sau D L, Yang X S 2012 *IEEE Transactions on Knowledge and Data Engineering* **24**(12) 2170-83
[12] Zhang Q, Li F, Yi K 2008 Finding frequent items in probabilistic data *Proceedings of the ACM SIGMOD International Conference on Management of Data Vancouver BC Canada*
[13] Agrawal R, Srikant R 1994 Fast algorithms for mining association rules in large databases *Proceedings of the 20th International Conference on Very Large Data Bases Santiago Chile*
[14] Han J, Pei J, Yin Y 2000 Mining frequent patterns without candidate generation *ACM SIGMOD - International Conference on Management of Data Dallas TX United states*
[15] Burdick D, Calimlim M, Flannick, J, Gehrke J, Yiu T 2005 *IEEE Transactions on Knowledge and Data Engineering* **17**(11) 1490-504
[16] El-hajj M, Zaïane O R 2003 COFI-tree mining: a new approach to pattern growth with reduced candidacy generation *IEEE. International Conference on Frequent Itemset Mining Implementations*
[17] Lin D I, Kedem Z M 1998 Pincer search: A new algorithm for discovering the maximum frequent set 1377 105-105
[18] [18] Zaki MJ, Hsiao C 2002 CHARM: An efficient algorithm for closed itemset mining *ACM SIGMOD - International Conference on Management of Data Dallas TX United states*
[19] Song W, Yang B, Xu Z 2008 Index-BitTableFI: An improved algorithm for mining frequent itemsets *Knowledge-Based Systems* **21**(6) 507-13
[20] Cam L L 1960 An approximation theorem for the Poisson binomial distribution *In Pacific Journal of Mathematics* **4**(10) 1181-97
[21] Goethals B 2011 Frequent itemset mining dataset repository http://fimi.cs.helsinki.fi/data/ Accessed 11

Authors

Si Tian, born on July 20, 1967, Xiangcheng Henan, China

Current position, grades: professor in Department of Scientific Research at Ningbo Dahongying University.
University studies: Ningbo Dahongying University.
Scientific interest: data mining and image processing.
Publications: 2 SCI, 7 EI.

Shui Wang, born on November 17, 1967, Nanyang Henan, China

Current position, grades: professor in the School of Information Engineering, Ningbo Dahongying University, China.
University studies: Lanzhou University.
Scientific interest: data mining and software engineering.
Publications number or main: 9 EI, 2 books.

Yang Liu, born on February 16, 1993, Huaian Jiangsu, China

Current position, grades: master in computer technology and application towards Ningbo Dahongying University, China.
University studies: M.S. degree in Dalian University of Technology.
Scientific interest: data mining
Publications: none

Le Wang, born on August 15, 1978, Nanyang Henan, China

Current position, grades: associate professor in the School of Information Engineering, Ningbo Dahongying University, China.
University studies: PhD degree in Computer Application from Dalian University of Technology, China, in 2013.
Scientific interest: frequent pattern mining, high utility pattern mining, data streams and big data.
Publications:2 SCI, 10 EI.