Mining Probabilistic High Utility Itemsets in Uncertain Databases

Yuqing Lan, Shengwei Yi, Dan Yu, and Jie Liang

School of Computer Science and Engineering, Beihang University, China
{lanyuqing, yishengwei, yudan, liangjie}@buaa.edu.cn

Abstract. Recently, with the growing popularity of Internet of Things (IoT) and pervasive computing, a large amount of uncertain data, i.e. RFID data, sensor data, real-time monitering data, etc., has been collected. As one of the most fundamental issues of uncertain data mining, the problem of mining uncertain frequent itemsets has attracted much attention in the database and data mining communities. Although some efficient approaches of mining uncertain frequent itemsets have been proposed, most of them only consider each item in one transaction as a random variable following the binary distribution and ignore the unit profits of items in the real scenarios. In this paper, we focus on the problem of *mining probabilistic high utility itemsets in uncertain databases* (MPHU), in which each item has an unit profit and likely appears multiple times in one transaction. In order to solve the *MPHU* problem, we propose a novel mining framework, called *UUIM*, which not only includes an efficient mining algorithm but also contains an effective pruning technique. Extensive experiments on both real and synthetic datasets verify the effectiveness and efficiency of proposed solutions.

1 Introduction

Data mining approach aims to derive the useful information from the massive data. In many application scenarios, what people generally concerned about are high frequent item sets, because this kind of item sets tend to have greater influence and improvements based on them are often the most effective. It can be said that people already have some effective solution of the uncertain transaction data set mining problem. But can they completely solve the problem of uncertain transaction data set? Let's see an example:

As we all know, from the beginning of human emergence, disease is the main reason which affect human health. According to WHO, in 2011, the number of some infectious disease cases in worldwide are more than 30 million[**?**]. If we statistic and summary the incidence of them, we can get a big data set where data mining approach can be useful. Table 1 shows the incidence of cases, it's a typical uncertain transaction data set which holds 5 transactions.

The decimal number in table represent the incidence of patients . For example, Zhao shows the incidence of apocleisis in two of ten days on average. The incidences of each patients in table is a transaction, each incidence is an item. If we use the traditional data mining approach we will obviously get frequent itemset like A, B, A, B. But these are not what we want, the reason is that even if they appear frequent, their impact on human health is not very high, whereas diseases which do great harm to human such as gastritis

Table 1. Statistic of patient incidences

PatientIncidence	Apocleisis(A)	Drowsiness(B)	Fever(C)	Cold(D)	Gastritis(E)	Appendicitis(F)	Lung cancer(G)
1. Zhao	0.2	0.3	0.05	0.02	0	0	0
2. Qian	0	0.2	0.045	0.015	0	0	0
3. Sun	0.1	0.3	0.055	0.01	0.005	0	0
4. Li	0.2	0.2	0.06	0	0.004	0.002	0
5. Zhou	0.3	0.4	0.04	0.015	0	0	0.001

and lung cancer are not found out. So the answer of the previous question is, frequent data mining approaches in uncertain environment cannot solve the problem entirely.

What should we do? We can get inspiration from the existing work of certain data mining and give the corresponding profit of each item based on it's importance. For example, table 2 shows the profit of each incidence in table 4 based on their harm to human beings. Hence, we can combine frequency and profit to find out useful information.

Table 2. the profit of each incidence

Incidence	Apocleisis(A)	Drowsiness(B)	Fever(C)	Cold(D)	Gastritis(E)	Appendicitis(F)	Lung cancer(G)
profit	1	1	5	10	100	500	1000

Therefore, in this paper, we address the above challenges and make the following contributions:

- To the best of our knowledge, this is the first work to formulate the problem of mining probabilistic high utility itemsets in uncertain databases (MPHU).
- Due to the challenges from utility constraints, we propose a novel mining framework, called *UUH-mine*, which not only includes an efficient mining algorithm but also contains an effective pruning technique.
- We verify the effectiveness and efficiency of the proposed methods through extensive experiments on real and synthetic datasets.

The rest of the paper is organized as follows. Preliminaries and our problem formulation are introduced in Section 2. In Section 3, we present a novel mining framework, called *UUH-mine*. Based on this framework, an efficient mining algorithm and an effective pruning technique is devised. Experimental studies on both real and synthetic data sets are reported in Section 4. In Section 5, we review the existing works and conclude this paper in Section 6.

2 Problem Formulation

In this section, we first introduce some basic concepts and then define the problem of mining probabilistic high utility itemsets in uncertain databases.

Given a finite set of items $I = \{i_1, i_2, ..., i_m\}$. Each item ip has a unit profit $w(i)$. An itemset X is a set of k distinct items, where k is the length of X. A transaction database $D = \{t_1, t_2, , t_n\}$ contains a set of transactions.

Definition 1 *The utility of an itemset I in transaction T is denoted as $U(I,D) = \sum_{i \in I} w(i)$, while $I \subseteq T$, it means the sum of each item profit in itemset.*

Definition 2 *The utility of an itemset I in transaction database D is denoted as $U(I,D) = sup(I) \times U(I)$.*

Definition 3 *An itemset is called a high utility itemset if its utility is no less than a user-specified minimum utility threshold which is denoted as minutil. Otherwise, it is called a low utility itemset.*

Thus, utility itemset mining share certain similarities with frequent itemset mining, but the solution is quite different. There is not a theorem like "downward closure"in utility itemset mining. In other words, even if an itemset is low utility, it's superset is not necessarily low utility. Therefore we must find some new strategies to limit the search space. We need some additional definition to solve the problem:

Definition 4 *The utility of a transaction is denoted as $U(T) = \sum_{i \in I} w(i)$, which means the sum of item profits in transaction.*

Definition 5 *The utility of a itemset transaction is denoted as $TU(I) = \sum_{i \subseteq I \wedge T \in D} w(i)$, which means the sum of the utility of transactions which contain the itemset I.*

At this point, we have defined all conceptions about utility itemset mining in certain environment, which laid the foundation for us to solve the utility itemset mining in uncertain environment.

Similar with uncertain frequent itemset mining, uncertain utility itemset mining can be divided into two kinds of semantic model, the expected utility semantic model and the probabilistic utility semantic model. In this paper we will talk about the latter.

For convenience, the probability of item I appears in uncertain transaction PT is denoted as $Pr(i, PT)$, so the probability of itemset I appears in PT is denoted as $Pr(i, PT) = \prod_{i \in I} Pr(i, PT)$.

Definition 6 *The expected utility of itemset I in transaction PT is denoted as $EU(I, PT) = U(I) \times Pr(I, PT)$.*

Definition 7 *The expected utility of itemset I in uncertain transaction database UD is denoted as $EU(I) = \sum_{PT \in UD} EU(I, PT)$.*

Definition 8 *The probabilistic utility of itemset I in uncertain transaction database UD is denoted as $up(I) = \sum_{1 \leq j \leq |PW| \wedge U(I, PW_j) \geq minutil}$, where minutil is the utility threshold.*

Definition 9 *The probabilistic utility itemset in uncertain transaction database is meets $up(I) \geq put$, where put is the probabilistic utility threshold.*

Problem Statement: Mining Probabilistic High Utility Itemsets in Uncertain Databases (MPHU). Given a uncertain transaction database UD, a user-specified minimum utility threshold minutil and a user-specified probabilistic utility threshold put, the problem of mining probabilistic utility itemsets in uncertain databases is to discover from UD all itemsets whose probabilistic utilities are no less than put.

Table 3. Example of uncertain transaction database

Id	Transaction
1	A(0.8), B(0.8), C(0.7), D(0.7)
2	A(0.9), C(0.8), G(0.8)
3	A(0.6), B(0.7), C(0.6), D(0.5), E(0.8), F(0.5)
4	B(0.7), C(0.8), D(0.5)
5	B(0.8), C(0.7), F(0.4)

Table 4. Profit

Item	A	B	C	D	E	F	G
Profit	3	4	1	1	8	2	1

For example, table 3 is an uncertain transaction database, table 4 is the profit of items in table 3. Given utility threshold minutil=5 and probabilistic utility threshold put=0.9, we can test whether A is a probabilistic utility itemset. For A, it's expected utility in transaction 1 is $EU(A, transaction1) = 3 \times 0.8 = 2.4$; it's expected utility is the sum of it's expected utility in each transaction which is $EU(A) = 2.4 + 2.7 + 1.8 = 6.9$, so the probabilistic utility of itemset A is equal to the probability of A appears two or more times in that uncertain transaction database, which is $up(A) = 0.432 + 0.288 + 0.048 + 0.108 = 0.876$. $up(A) < put$, so itemset A is not a probabilistic utility itemset.

3 UUH-mine Framework

In this section, we propose a novel data mining framework UUH-mine to solve the U; then we will describe several optimization approaches such as global bound and local pruning in order to make the algorithm efficient; finally, we will use the pseudo code and examples to describe the algorithm in detail.

3.1 Main Ideas

In this subsection, we first illustrate the main idea of the UUH-mine framework using the table 3 and 4.

First it takes one scan of the transaction database to calculate each 1-itemset's profit and build the header table H shown in Fig. 1 based on the result. Information such as

profit and expected support count of all items in dataset are included in table. Then transactions with the same first item are linked together by the hyper-links into a queue, and the entries in header table H act as the heads of the queues. For example, the entry of item A in the header table H is the head of the A-queue, which links transactions start with A. Then we can use this structure to check whether itemset A's utility is high.

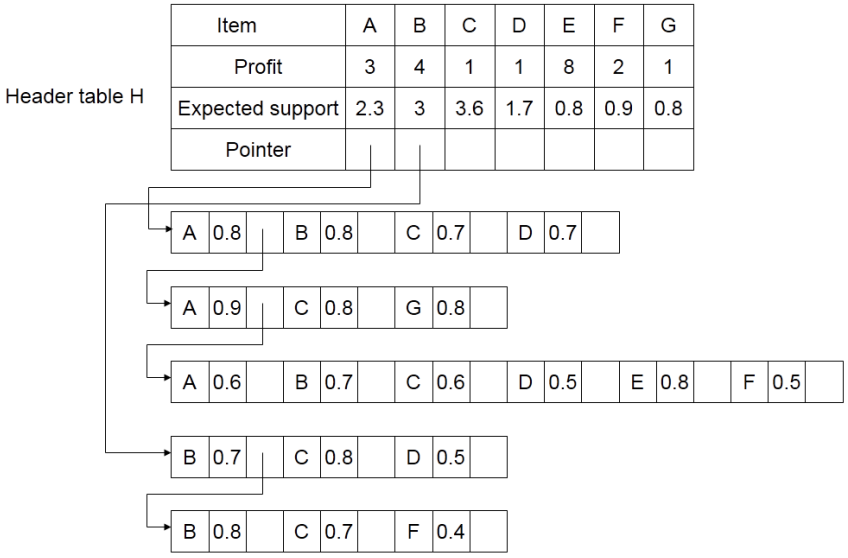

Fig. 1. initial UUH-struct created from uncertain transaction database

We can create a projection database through A-queue(which contains first, second and third transaction in Fig. 2. Then we can calculate profit and excepted support count of all 2-itemsets(such as A, B, A, C) in projection database and create the header table H_A(as shown in Fig. 2) and then link transactions to entries in header table H_A. This structure can be traversed efficiently to find whether A, B is a high utility itemset.

Similarly, the process continues for the a-projected database and create header table H_{AB}, H_{ABC}..... to find high utility itemsets from all itemsets contaning A. After high utility itemsets containing item A are found, the A-projected database, i.e., A-queue, is no longer needed for the remaining mining processes. So we delete item A from database and update the information in header table H(as shown in Fig. 3) to find high utility itemsets containing B.

We use UUH-mine framework to find all high utility itemsets from database through depth first search. However, this UUH-mine framework needs to traverse all 2^m(m is the

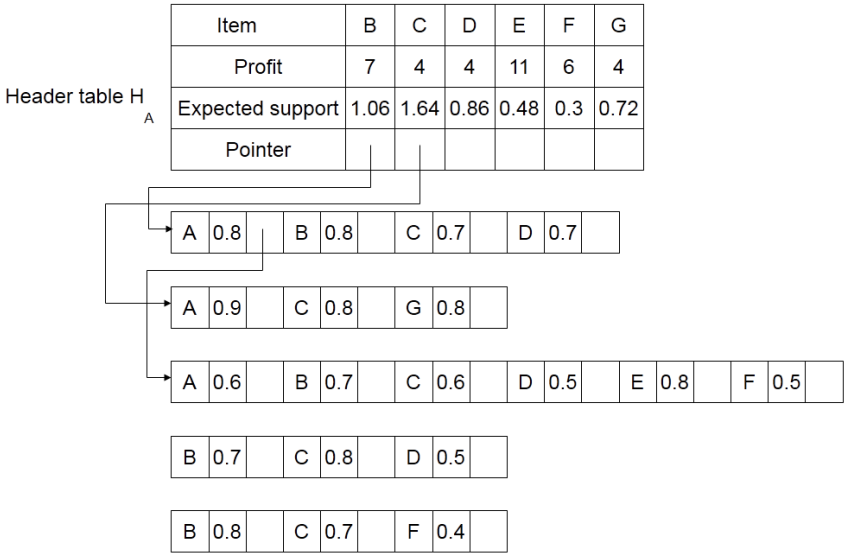

Fig. 2. UUH-struct after creating header table H_A

Table 5. Uncertain Database Adding Maximum Transaction Expected Utility

Index	Transaction content	Maximum transaction expected utility
1	A(0.8), B(0.8), C(0.7), D(0.7)	4.48
2	A(0.9), C(0.8), G(0.8)	2.88
3	A(0.6), B(0.7), C(0.6), D(0.5), E(0.8), F(0.5)	6.72
4	B(0.7), C(0.8), D(0.5)	2.8
5	B(0.8), C(0.7), F(0.4)	3.2

number of different items in database) itemsets to find the result, so we must optimize it.

3.2 Algorithm optimization

In this section, we will introduce the optimization scheme of UUH-mine framework.

Definition 10 *For the given uncertain transaction PT, it's transaction maximum expected utility equals to the max expected utility of itemsets it contains, which is $MU(PT) = maxU(I)|I \subseteq PT$.*

According to Definition 10, we can add a new property to our example as shown in table 5.

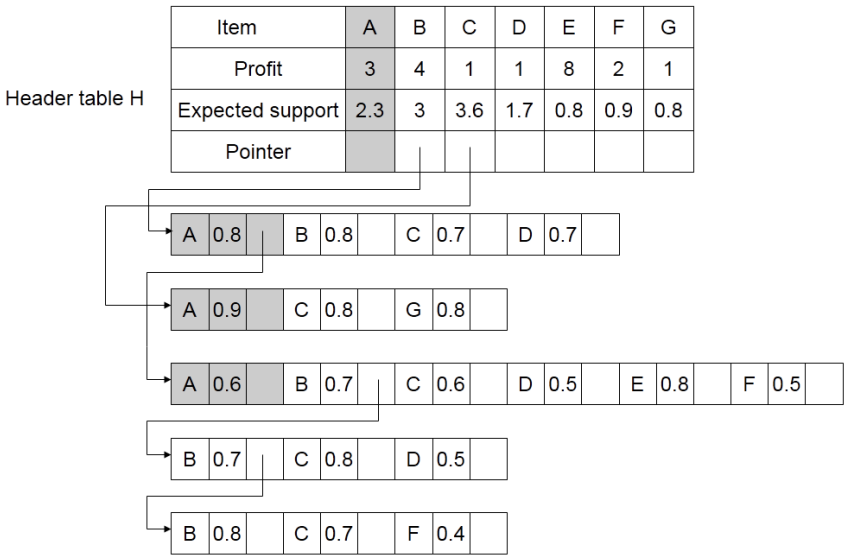

Fig. 3. UUH-struct after removing item A

The maximum expected utility of each transaction is calculated through it's content. For example, in table 5, the maximum expected utility of transaction 1 is 4.48, because the max expected utility of all 16 itemsets contained by transaction 1 is A, B, which is 4.48.

In this example, the biggest transaction contains only 6 items, so we can calculate maximum expected utility of every transaction through exhaustive method. However, in practical problems, a transaction may contains many items,so the exhaustive method is not efficient. Here we will introduce a fast way:

Given a transaction PT which length(the number of items it contains) is L, items in PT is $i_1, i_2, , i_L$, probability of each item is $p_1, p_2, , p_L$. We can give the sub problem $S_{I,j}$(i represent a itemset), which means the maximum expected utility in set which contains itemsets derived from itemset I and the last j items in PT. Obviously we have $EU(PT) = S_{\phi,L}$ and $S_{I,0} = EU(I, PT)$ as well as recursive relation: $S_{I,j} = max(S_{I \cup i_{L-j+1}, j-1}, S_{I,j-1})$ (3.2). We can get $EU(PT)$ from that recursive relation, but it still need to consider all 2^L itemsets. So we can use this theorem to optimize it:

Theorem 1 *If there exist itemset I_1 and $I_2 = I_1 \cup i_j$ (I_2 represents a superitemset which has one more item than I_1), and meet $EU(I_1, PT)$ $EU(I_2, PT)$, then expected utilities of I_2 and all superitemsets of I_2 cannot be the maximum expected utility of PT.*

Rationale 1 *The expected utility of I_2 obviously cannot be the maximum expected utility of PT, so we will prove that is also true for I_2's superitemsets $I_3 = I_2 \cup I^{'}$. For one of I_2's superitemset , we can construct a new itemset $I_4 = I_1 \cup I^{'}$, then we have*

$$\begin{aligned} EU(I_3, PT) &= U(I_3) \times Pr(I_3, PT) \\ &= (U(I_2) + U(I^{'})) \times Pr(I_2, PT) \times Pr(I^{'}, PT) \\ &= (EU(I_2, PT) + U(I^{'}) \times Pr(I_2, PT)) \times Pr(I^{'}, PT) \end{aligned} \tag{1}$$

$$\begin{aligned} EU(I_4, PT) &= U(I_4) \times Pr(I_4, PT) \\ &= (U(I_1) + U(I^{'})) \times Pr(I_1, PT) \times Pr(I^{'}, PT) \\ &= (EU(I_1, PT) + U(I^{'}) \times Pr(I_1, PT)) \times Pr(I^{'}, PT) \end{aligned} \tag{2}$$

because $EU(I_1, PT) > EU(I_2, PT)$ and $Pr(I_1, PT) \geq Pr(I_2, PT)$, so $EU(I_3, PT) < EU(I_4, PT)$, so the expected utility of I_3 cannot be the maximum expected utility of PT.

We can use Theorem. 1 to optimize Eq. 3:

$$S_{i,j} = \begin{cases} max\{S_{I \cup \{i_{L-j+1}\}, j-1}, S_{I,j-1}\} & (EU(I, PT) < EU(I \cup \{i_{L-j+1}\}, PT)) \\ S_{I,j-1} & (EU(I, PT) \geq EU(I \cup \{i_{L-j+1}\}, PT)) \end{cases} \tag{3}$$

Using Eq. 3 can greatly speed up the calculation of transaction maximum expected utility.

Definition 11 *For itemset I and uncertain transaction database UD, the transaction maximum expected utility of itemset I is $MU(I) \sum_{I \subseteq PT \wedge PT \in UD} MU(PT)$. Which means the sum of transaction maximum expected utility of transactions containing itemset I.*

We can derive Lemma 1 through definitions above:

Lemma 1 *The transaction maximum expected utilities of I's superitemsets are not more than I's. For $I \subseteq I^{'}$, there always have $MU(I) \leq MU(I^{'})$.*

According to Chernoff bound[**?**], random variables $X_1, X_2, , X_n$ are results of n times independent Poisson experiments(X_j can only be 0 or 1). Given $X = \sum_{j=1}^{n} X_j$ and$\mu = E[X]$, for any positive read number δ, there exists the following inequality:

$$Pr(X \leq (1+\delta)\mu) < \{\frac{\mathrm{e}^{\delta}}{(1+\delta)^{1+\delta}}\}^{\mu} (\delta > 0) \tag{4}$$

In the problem of mining probabilistic high utility itemsets, for one itemset I, it's appear in one uncertain transaction PT can be seen as a independent Possion experiment, and the real support of I, I.e. the $sup(I)$ is the sum of many Possion experiments,

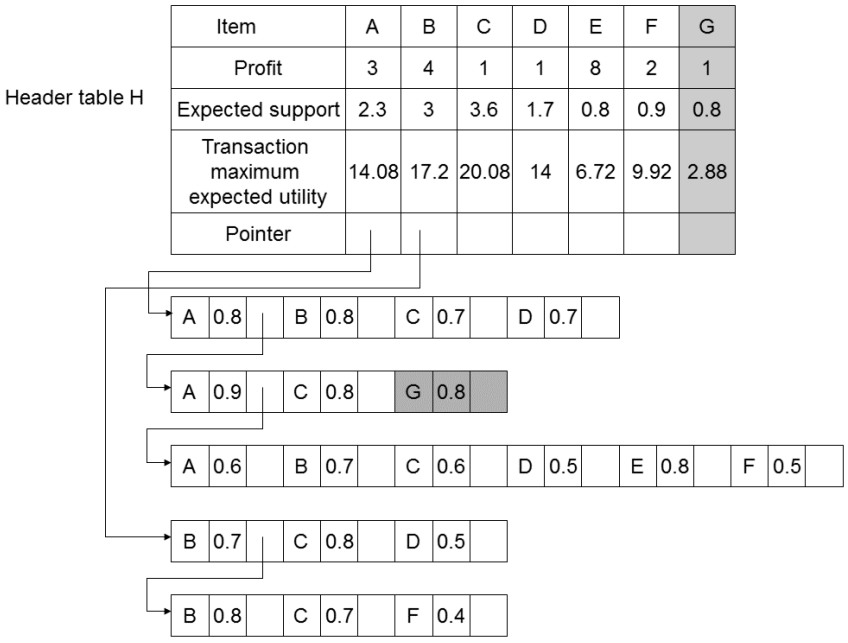

Fig. 4. initial UUH-min data structure after optimization

so the expect of that variable is the expected support count of I. So the utility probability of I we need to calculate is:

$$Pr(sup(I) \times U(I) \geq minutil) = Pr(sup(I) \geq \frac{minutil}{U(I)}) \tag{5}$$

Combine Eq. 4 and Eq. 5, when $esupp(I) < \frac{minutil}{U(I)}$, we can let $(1+\delta)esup(I) = \frac{minutil}{U(I)}U(I)$, so we can get $\delta = \frac{minutil}{U(I)esup(I)} - 1 = \frac{minutil}{EU(I)} - 1$. Feed it into Eq. 4 can get:

$$Pr(up(I) \geq minutil) < (\frac{\mathrm{e}^{\delta}}{(1+\delta)^{1+\delta}})^{esup(I)} \tag{6}$$

while $\delta = \frac{minutil}{EU(I)} - 1$.The right side of inequality 6 is a decreasing function of δ, so when δ decreases, the original inequality still holds. Hence we can get the following Lemma:

Lemma 2 *For itemset I, given uncertain transaction database UD, utility threshold $minutil$ and probabilistic utility threshold put, if $MU(I) < minutil$ and $(\frac{\mathrm{e}^{\delta}}{(1+\delta)^{1+\delta}})^{esup(I)} \leq put$ (where $\delta = \frac{minutil}{MU(I)} - 1$), then itemset I and all of it's superitemsets cannot be utility itemsets.*

Lemma 2 is the key point to solve the "uncertain database utility itemset minin" problem, it can greatly reduce the search space so that the algorithm can be efficient. We can use it to optimize the UUH-mine framework mentioned in last section, as shown

in Fig. 4. Because the transaction maximum expected utility of itemset G is 2.88, cannot pass the check in Lemma 2, so G and all of it's superitemsets cannot be utility superitemsets and we don't need to check them.

Of course, we can use the same method to optimize the other header tables generated by projection databases(such as H_A, H_{AB}, etc).

3.3 UUIM algorithm

In this section, we will introduce UUIM(Uncertain Utility Itemsets Mining) algorithm in detail. First, we will give the overall outline of this algorithm, which divides the mining process into two phases and briefly describe them. Then we will explain that two phases in detail.

Algorithm 1: UUIM Algorithm

Input: uncertain transaction database UD, utility threshold $minutil$, probabilistic utility threshold put
Output: set of utility itemset UIS

```
1 UIS;
2 HInitializeHeader(UD, minutil, put);
3 Recursion(H, UD, minutil, put, UIS);
4 return UIS
```

Algorithm 1(UUIM) is the algorithm framework of mining utility itemsets. This algorithm is purposed to mine all utility itemsets UIS from the given uncertain transaction database UD through the given utility threshold minutil and probabilistic utility threshold put. Line 1 is used to initialize the result set UIS; line 2 creates the initial header table H of the UUH-mine framework through function InitializeHeader; line 3 use the key function Recursion search each items in depth first way; the last line return the calculation results which are all utility itemsets. According to this we can know that the main part of this algorithm consists of two parts, one for creating header table which is explained above in detail; the other is the recursive function Recursion which is used to traverse all probabilistic utility itemsets.

Next, we will introduce how the key function Recursion works. In algorithm 2(Recursion), line 1 traverses each itemset in header table; line 2 and 3 check whether the new itemset is utility itemset, if true it will be added to result; line 4 checks whether that new itemset can pass the Eq. (3.6), if passed a header table will be created in line 5 and traversed in line 6.

In summary, this section explains the probabilistic utility itemset mining algorithm, UUIM. First we introduce a new data structure UUH-struct, which lay the foundation of the algorithm's efficiency; then we describe several algorithm optimization methods such as global bound and local pruning in order to speed up the algorithm.

Algorithm 2: Recursion Algorithm

Input: header table H_I, uncertain transaction database UD, utility threshold $minutil$, probabilistic utility threshold put, current set of utility itemset UIS

Output: updated set of utility itemset UIS

```
for each i in H_I do
    if I ∪ i is utility itemset do then
        UIS = UIS ∪ I ∪ i
    if ChernoffCheck() = true do then
        create new header table H_{I∪i};
        Recursion(H, UD, minutil, put, UIS);
End;
```

4 Experimental evaluation

Experiments are performed to test the effectiveness of our UUIM algorithm in obtaining the high utility transactions. First we will introduce the software platform, hardware platform, datasets, variable setting rules and the experiment; then the effectiveness of UUIM will be considered through it's running time, especially with Chernoff pruning strategy; next the memory cost of UUIM will be compared and finally, we will conclusion this section.

4.1 Introduction to experiment platform

All experiments were performed on a Intel(R) Core(TM) i5 M450 2.40GHz PC machine with 4GB main memory, running Microsoft Windows 8.1. UUIM was implemented by us using C++ in Visual Studio 2012.

In order to ensure the fairness of experiment, according to the experimental evaluation method of probabilistic database research field in previous, we will assign probability to transactions in real certain transaction database to get an uncertain transaction database. The way we used for assigning is assign a probability value to one transaction, the value obey the normal distribution given expected and variance. We use the classical transaction database Mushroom which comes from Audobon Society Field Guide and contains 8124 transactions and 120 different items.

Given experiment platform, we will further explain the evaluation scheme. The evaluation scheme consists of two parts, the algorithm efficiency evaluation and the memory cost evaluation. During the evaluation progress, we mainly compare the efficiency of UUIM algorithm and UUIM-NoCh algorithm. The UUIM-NoCh algorithm didn't use the Chernoff pruning strategy. According this comparison we can verify the validity of that strategy.

Besides, the main variables used in UUIM are utility threshold minutil and probabilistic utility threshold put, so the following two experiments will show the influence of these two variables to UUIM algorithm and Chernoff bound pruning strategy. The following are the experiment results.

4.2 Algorithm efficiency evaluation

Based on the experiment scheme in last section, this section mainly evaluate the algorithm efficiency. We will evaluate from three angles: change of utility threshold, change of probabilistic utility threshold and whether use Chernoff bound pruning strategy. In order to hold the principle of single variable, the default parameter setting is: default utility threshold minutil = 10000, default probabilistic utility threshold put = 0.6. When one of them is changing, the other keep the default value. Moreover, the range of the utility threshold minutil is 5000 to 15000 while the range of the probabilistic utility threshold put is 0.5 to 0.9. According these big ranges, we can clearly see the characteristics of UUIM algorithm and UUIM-NoCh algorithm thus the presented experiment results can be more objective.

Fig. 5 shows the comparison of UUIM algorithm and UUIM-NoCh algorithm in different utility threshold. Clearly, in the same parameter setting the efficiency of UUIM algorithm is about two times more than the efficiency of UUIM-NoCh algorithm. And the difference becomes lager as the utility threshold goes lower. This shows that the Chernoff bound pruning strategy can efficiently reduce the search space, greatly decrease the number of itemsets need to be calculated, thus greatly increase the algorithm efficiency.

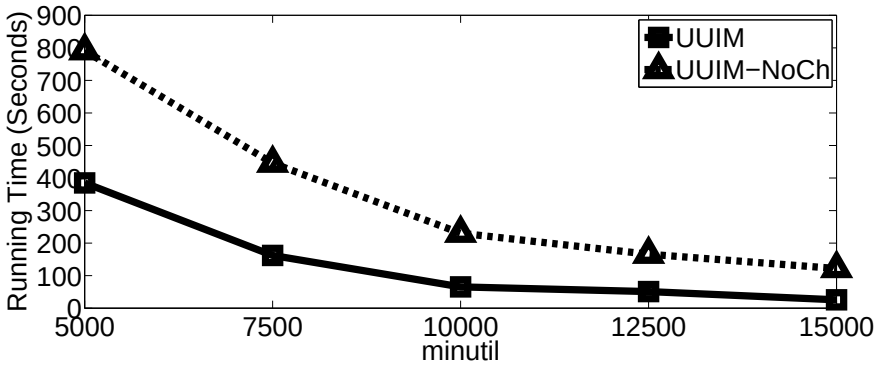

Fig. 5. Running Time vs. Minimum Utility Threshold

Then we test the influence of the change of probabilistic utility threshold to UUIM algorithm and UUIM-NoCh algorithm. Fig. 6 shows that while the probabilistic utility goes higher, the running time of UUIM-NoCh left largely unchanged, but the running time of UUIM algorithm is decreasing. Thus shows that while the probabilistic utility threshold gets higher, the effect of Chernoff bound pruning strategy will be better, but it does not influence UUIM-NoCh algorithm.

4.3 Memory cost evaluation

Based on the experiment scheme, this section mainly evaluate the memory cost of the algorithm. During this process, we will mainly compare the peak memory cost. As mentioned in section 3, UUH-struct in UUIM algorithm need many header tables to store

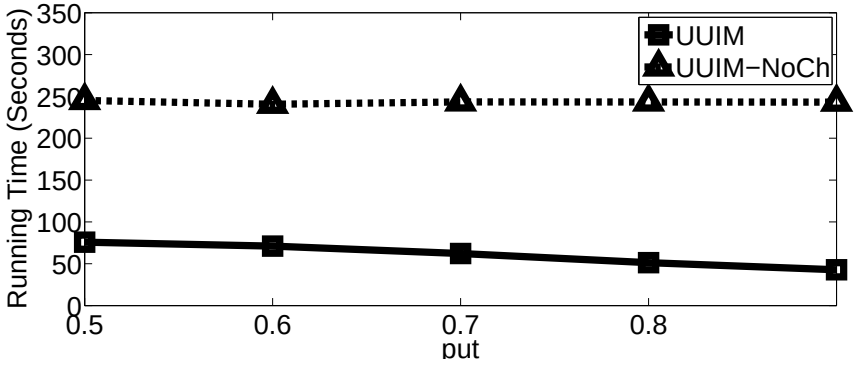

Fig. 6. Running Time vs. Probabilistic Uility Threshold

the transaction maximum utility of each itemset in order to use the pruning strategy. So we will focus on whether we can control the increase extant of memory cost.

Similar to last experiment, the default utility threshold is minutil=10000 and the default probabilistic utility threshold is put=0.6. The range of the utility threshold is 5000 to 15000 while the range of the probabilistic utility threshold is 0.5 to 0.9.

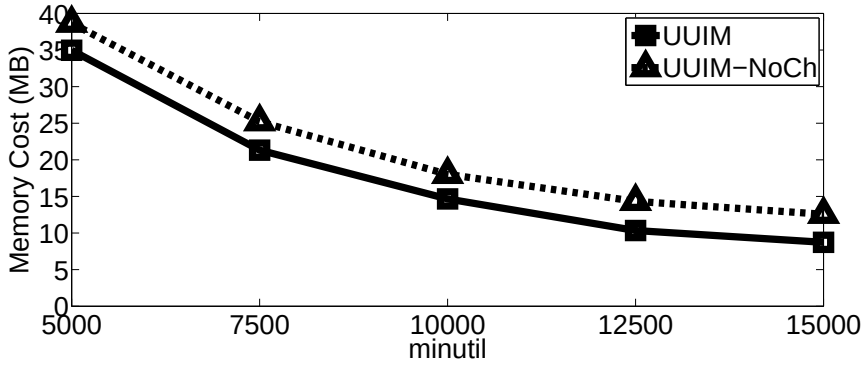

Fig. 7. Memory Cost vs. Minimum Uility Threshold

Fig. 7 shows the comparison of UUIM and UUIM-NoCh on Mushroom dataset. We can see that while the utility threshold goes higher, the memory cost of these two algorithms decreases, and the difference between UUIM and UUIM-NoCh is very small, hence we can conclude that the operation of adding transaction maximum utility to header table doesn't have obvious effect.

Fig. 8 shows the memory cost of UUIM and UUIM-NoCh in different probabilistic utility threshold. It shows that probabilistic utility threshold has almost no effect on memory cost. Besides, there is no difference between the memory cost of two algorithms.

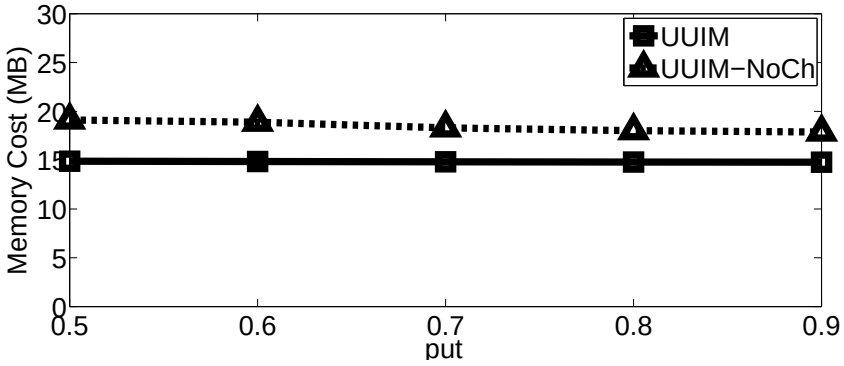

Fig. 8. Memory Cost vs. Probabilistic Uility Threshold

Therefore, although Chernoff bound pruning strategy needs more space, it's influence on total memory cost is small, so the strategy almost has no effect on the memory cost of the algorithm.

4.4 Experiments conclusion

In algorithm efficiency evaluation experiment, we can clearly see that the efficiency of UUIM algorithm is much more than UUIM-NoCh, and the difference get bigger while the utility threshold and the probabilistic utility threshold goes higher.

In Memory cost evaluation experiment, we can see that two algorithms need same memory in different utility threshold and probabilistic utility threshold. Thus the Chernoff bound pruning strategy doesn't increase the memory cost.

In summary, our UUIM algorithm is obviously better than base algorithm.

5 Related Work

In this section, we review the related work in two categories, mining frequent itemsets in deterministic and uncertain data.

5.1 Deterministic High Utility Itemset Mining

Since Rakesh Agrawal first proposed the concept of mining frequent itemsets (or called mining large itemset) [2], many efficient algorithms about mining frequent itemsets have been designed, such as FP-growth [7], Eclat [16], and so on.

Especially, utility itemset mining, also generally called utility pattern mining, was first introduced in [15]. Although the UMining algorithm was proposed by [15], it cannot extract the complete set of them. A transaction-weighted downward closure property was introduced in [9], in which a two-phase algorithm was proposed and performed faster than UMining. Moreover, IHUP [3] maintains the high utility patterns in an incremental environment; since it avoids multiple scans of the database, its efficiency is far better than [9]. Recently, UP-Growth [12, 14] also uses a tree structure, UP-Tree,

to mine high utility itemsets. Compared to IHUP, UP-Growth is more efficient, since it further reduces the number of promising patterns which cannot be pruned in IHUP.

5.2 Uncertain Frequent Itemset Mining

Another set of related researches with our work is the issues of mining frequent itemsets over uncertain data. Different from the certain case, the definition of a frequent itemset in uncertain data has two types of probabilistic semantics: *expected support-based frequent itemset* [1, 6] and *probabilistic frequent itemset* [4]. In the definition of the expected support-based frequent itemset, the expectation of the support of an itemset is defined as the measurement, called as the expected support of this itemset [1, 5, 6, 8]. In the definition of probabilistic frequent itemset [4, 10, 13], the probability that an itemset appears at least the minimum support (*min_sup*) times is defined as the measurement, called as the frequent probability of an itemset. Recently, [11] using an experimental method shows the aforementioned two definitions of uncertain frequent itemsets is actually equivalent while the uncertain databases are very large.

Although there are related researches of mining frequent itemsets over uncertain data, all of them are built over the assumption that each item has no unit profit and only appear once in each transaction. In other words, none of existing work cannot address the problem of mining probabilistic high utility itemsets over uncertain data.

6 Conclusion

In this paper, we formulate a new type of problem of mining uncertain frequent itemsets, called *mining probabilistic high utility itemsets in uncertain databases* (MPHU), where each item has an unit profit and likely appears multiple times in one transaction. In order to solve the *MPHU* problem, we propose a novel mining framework, called *UUIM*, which not only includes an efficient mining algorithm but also contains an effective pruning technique. Extensive experiments on both real and synthetic datasets verify the effectiveness and efficiency of proposed solutions.

References

1. Aggarwal, C.C., Li, Y., Wang, J., Wang, J.: Frequent pattern mining with uncertain data. In: KDD. pp. 29–38 (2009)
2. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules in large databases. In: VLDB. pp. 487–499 (1994)
3. Ahmed, C.F., Tanbeer, S.K., Jeong, B., Lee, Y.: Efficient tree structures for high utility pattern mining in incremental databases. IEEE Trans. Knowl. Data Eng. 21(12), 1708–1721 (2009)
4. Bernecker, T., Kriegel, H.P., Renz, M., Verhein, F., Züfle, A.: Probabilistic frequent itemset mining in uncertain databases. In: KDD. pp. 119–128 (2009)
5. Chui, C.K., Kao, B.: A decremental approach for mining frequent itemsets from uncertain data. In: PAKDD. pp. 64–75 (2008)
6. Chui, C.K., Kao, B., Hung, E.: Mining frequent itemsets from uncertain data. In: PAKDD. pp. 47–58 (2007)

7. Han, J., Pei, J., Yin, Y.: Mining frequent patterns without candidate generation. In: SIGMOD. pp. 1–12 (2000)
8. Leung, C.K.S., Mateo, M.A.F., Brajczuk, D.A.: A tree-based approach for frequent pattern mining from uncertain data. In: PAKDD. pp. 653–661 (2008)
9. Liu, Y., Liao, W., Choudhary, A.N.: A two-phase algorithm for fast discovery of high utility itemsets. In: Advances in Knowledge Discovery and Data Mining, 9th Pacific-Asia Conference, PAKDD 2005, Hanoi, Vietnam, May 18-20, 2005, Proceedings. pp. 689–695 (2005)
10. Sun, L., Cheng, R., Cheung, D.W., Cheng, J.: Mining uncertain data with probabilistic guarantees. In: KDD. pp. 273–282 (2010)
11. Tong, Y., Chen, L., Cheng, Y., Yu, P.S.: Mining frequent itemsets over uncertain databases. PVLDB 5(11), 1650–1661 (2012)
12. Tseng, V.S., Wu, C.W., Shie, B.E., Yu, P.S.: Up-growth: an efficient algorithm for high utility itemset mining. In: KDD. pp. 253–262 (2010)
13. Wang, L., Cheng, R., Lee, S.D., Cheung, D.W.L.: Accelerating probabilistic frequent itemset mining: a model-based approach. In: CIKM. pp. 429–438 (2010)
14. Wu, C., Shie, B., Tseng, V.S., Yu, P.S.: Mining top-k high utility itemsets. In: The 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '12, Beijing, China, August 12-16, 2012. pp. 78–86 (2012)
15. Yao, H., Hamilton, H.J., Butz, C.J.: A foundational approach to mining itemset utilities from databases. In: Proceedings of the Fourth SIAM International Conference on Data Mining, Lake Buena Vista, Florida, USA, April 22-24, 2004. pp. 482–486 (2004)
16. Zaki, M.J.: Scalable algorithms for association mining. IEEE Trans. Knowl. Data Eng. 12(3), 372–390 (2000)