Mining Probabilistic High Utility Itemsets in Uncertain Databases

1st Author
1st author's affiliation
1st line of address
2nd line of address
Telephone number, incl. country code
1st author's E-mail address

2nd Author
2nd author's affiliation
1st line of address
2nd line of address
Telephone number, incl. country code
2nd E-mail

3rd Author
3rd author's affiliation
1st line of address
2nd line of address
Telephone number, incl. country code
3rd E-mail

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 values 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 a unit value. 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.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Application — Data Mining.

General Terms

Algorithms, Theory

Keywords

Uncertain Databases, Utility Itemsets Mining, Probabilistic Data, Probabilistic High Utility Itemsets

1. INTRODUCTION

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 values of items in the real scenarios. In recommender systems (e.g. music, video) analysis, mining frequent itemsets from an uncertain database refers to discovery of the itemsets which may frequently appear together in the records (transactions). However, the unit values (profits) of items are not considered in the framework of uncertain frequent itemsets mining. Hence, it cannot satisfy the requirement of the user who is interested in discovering the itemsets with high enjoyment values. In view of this, utility mining will emerges as an important topic in data mining for discovering the itemsets with high utility like values (profits) in uncertain databases.

Mining high utility itemsets from the databases refers to finding the itemsets with high utilities. The basic meaning of utility is the interestedness / importance / profitability of items to the users. Before finding high utility itemsets over uncertain databases, the definition of the high utility itemset is the most essential issue. In deterministic data, the utility of items in a transaction database consists of two aspects: (1) the importance of distinct items, which is called external utility, and (2) the importance of the items in the transaction, which is called internal utility. The utility of an itemset is defined as the external utility multiplied by the internal utility. An itemset is called a high utility itemset if its utility is no less than a user-specified threshold. However, different from the deterministic case, the utility of items in an uncertain transaction database only contain the importance of distinct items, which is called external utility in deterministic database. The definition of a high utility itemset over uncertain data has two different semantic explanations: expected support-based high utility itemset and probabilistic high utility itemset. However, the two definitions are different on using the random variable to define high utility itemsets. In the definition of the expected support-based high utility itemset, the expectation of the utility of an itemset is defined as the measurement, called as the expected utility of this itemset. In this definition, an itemset is high utility if and only if the expected utility of such itemset is no less than a specified minimum expected utility threshold, min eutil. In the definition of probabilistic utility itemset, the probability that the utility of an itemset is no less than the threshold is defined as the measurement, called as the high utility probability of an itemset, and an itemset is high utility if and only if the high utility probability of such itemset is larger than a given probabilistic threshold.

CIKM'15, October 19–23, 2015, Melbourne, AU.

Mining high utility itemsets from uncertain databases is an important task which is essential to a wide range of applications such as most of recommender systems analysis, Internet of Things (IoT) and even biomedical applications. The definition of probabilistic high utility itemset includes the complete probability distribution of the utility of an itemset. Although the expectation is known as an important statistic, it cannot show the complete probability distribution. Hence, we mainly discuss mining probabilistic high utility itemsets in this paper.

To sum up, we 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 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 datasets are reported in Section 4. In Section 5, we review the existing works and conclude this paper in Section 6.

2. PROBLEM DEFINITIONS

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 i_p has a unit value v (i_p). 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. The number of transactions containing X in D is a random variable denoted as *sup* (X).

Definition 1. The utility of an itemset X in transaction T is denote as U (X, T):

$$U(X,T) = \sum_{i \in X} v(i)$$

Definition 2. The utility of an itemset X in transaction database D is denoted as U (X, D):

$$U(X,D) = U(X) \times \sup(X)$$

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.

As is well-known, utility itemset mining is not "downward closure", in other words, even if an itemset is low utility, the superset of it also may be a high utility itemset. 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):

$$U(T) = \sum_{i \in T} v(i)$$

Definition 5. The utility of an itemset transaction is denoted as TU (X):

$$TU(X) = \sum_{X \subset T \wedge T \in D} U(T)$$

which means the sum of the utility of transactions which contain the itemset X.

For convenience, the probability of item i appears in uncertain transaction T is denoted as P (i, T), so the probability of itemset X appears in T is denoted as P (X, T) :

$$P(X,T) = \prod_{i \in X} P(i,T)$$

Table 1. An Uncertain Database

TID	Transaction
T_1	(A, 0.8) (B, 0.8) (C, 0.7) (D, 0.7)
T_2	(B, 0.9) (C, 0.8) (G, 0.8)
T_3	(A, 0.6) (B, 0.7) (C, 0.6) (D, 0.5) (E, 0.8) (F, 0.5)
T_4	(B, 0.7) (C, 0.8) (D, 0.5)
T_5	(B, 0.8) (C, 0.7) (F, 0.4)

Table 2. Value Table

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

Table 3. The Probability Distribution of sup (A)

sup (A)	0	1	2	3
Probability	0.008	0.116	0.444	0.432

Definition 6. The expected utility of itemset X in transaction T is denoted as EU (X, T):

$$EU(X,T) = U(X) \times P(X,T)$$

Definition 7. The expected utility of itemset X in uncertain transaction database D is denoted as EU (I):

$$EU(X) = \sum_{T \in D} EU(X,T)$$

For example, the expected utility of {A} in transaction 1 is EU ({A}, transaction1) = 3 * 0.8 = 2.4, the expected utility of {A} in database is EU ({A}, D) = 2.4 + 2.7 + 1.8 = 6.9.

Definition 8. The probabilistic utility of itemset X in uncertain transaction database D is denoted as *up* (X):

$$up(X) = \sum_{U(X) \times \sup(X) \geq minutil} P(\sup(X))$$

Definition 9. An itemset X is called a probabilistic utility itemset in uncertain transaction database if up(X) $\geqslant$ put, put is the probabilistic utility threshold.

For example, given utility threshold minutil = 5 and probabilistic utility threshold put = 0.9, v (A) = 3, up ({A}) = 0.8*0.6*0.9 + (1-0.8)*0.6*0.9 + 0.8*(1-0.6)*0.9 + 0.8*0.6*(1-0.9) = 0.876. Due to up ({A}) < put, itemset {A} is not a probabilistic utility itemset.

We are now able to specify the Mining Probabilistic High Utility Itemsets (MPHU) problem as follows; given an uncertain transaction database T, a minimum utility threshold minutil and a utility probability threshold put.

3. AGLORITHMS OF PROBABILISTIC HIGH UTILITY ITEMSET MINING

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.

3.1 Framework

In this subsection, we first illustrate an extended H-mine algorithms. We call it UUH-Struct.

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 (value) 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.

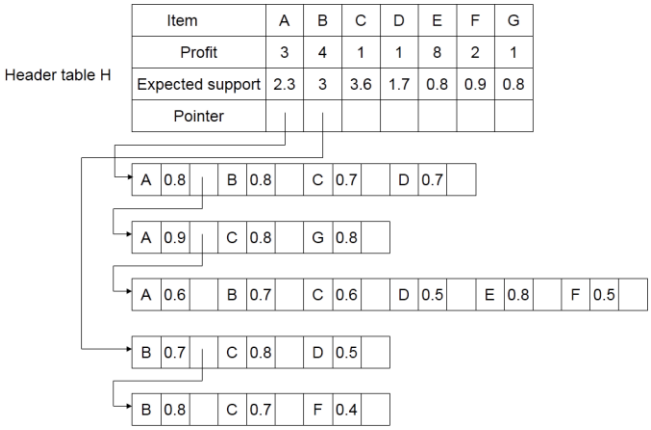


Figure 1. initial UUH-Struct created from uncertain transaction database

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

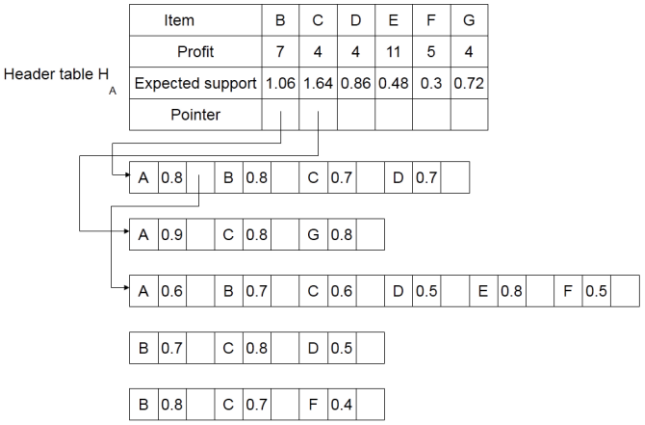


Figure 2. UUH-Struct after creating header table H_A

3.2 Optimization Strategies

Definition 10. For the given uncertain transaction T, its transaction maximum expected utility equals to the max expected utility of itemsets it contains, denoted as MU (T):

$$MU(T) = \max\{EU(X) | X \subset T\}$$

For example, the maximum expected utility of transaction 1 is 4.48, and the maximum expected utility of all 16 itemsets contained by transaction 1 is {A, B}, refer to Table 4.

Table 4. Uncertain Database with Maximum Transaction Expected Utility

TID	Transaction	MU
T_1	(A, 0.8) (B, 0.8) (C, 0.7) (D, 0.7)	4.48
T_2	(B, 0.9) (C, 0.8) (G, 0.8)	2.88
T_3	(A, 0.6) (B, 0.7) (C, 0.6) (D, 0.5) (E, 0.8) (F, 0.5)	6.72
T_4	(B, 0.7) (C, 0.8) (D, 0.5)	2.8
T_5	(B, 0.8) (C, 0.7) (F, 0.4)	3.2

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 T which length (the number of items it contains) is L, items in T is i_1, i_2…, i_L, probability of each item is p_1, p_2 …, p_L. We can give the sub problem $S_{X,j}$ (X represent a itemset), which means the maximum expected utility in set which contains itemsets derived from itemset X and the last j items in T . Obviously, we have EU (T) = $S_{\emptyset,L}$ and $S_{I,0}$ = EU (X, T) as well as recursive relation:

$$S_{X,j} = \max\{S_{X\cup\{i_{L-j+1}\},j-1}, S_{X,j-1}\}$$

We can get EU (T) from that recursive relation, but it still need to consider all 2^L itemsets. So we can optimize it by the theorem:

Theorem 1. If there exist itemset X_1 and $X_2 = X_1 \cup \{i_j\}$ (X_2 represents a super-itemset which has one more item than X_1), and EU (X_1, T) > EU (X_2, T), then expected utility of X_2 and all super-itemsets of X_2 cannot be the maximum expected utility of T .

Rationale 1. The expected utility of X_2 obviously cannot be the maximum expected utility of T , so we will prove that is also true for X_2's super-itemsets $X_3 = X_2 \cup X'$. For one of X2's super itemset, we can construct a new itemset $X_4 = X_1 \cup X'$, then we have:

$$EU(X_3, T) = U(X_3) \times P(X_3, T)$$
$$= (U(X_2) + U(X')) \times P(X_2, T) \times P(X', T)$$
$$= (EU(X_2, T) + U(X') \times P(X_2, T)) \times P(X', T)$$
$$EU(X_4, T) = U(X_4) \times P(X_4, T)$$
$$= (U(X_1) + U(X')) \times P(X_1, T) \times P(X', T)$$
$$= (EU(X_1, T) + U(X') \times P(X_1, T)) \times P(X', T)$$

Due to $EU(X_1, T) > EU(X_2, T)$ and $P(X_1, T) \geqslant P(X_2, T)$, we can know $EU(X_3, T) < EU(X_4, T)$, namely, the expected utility of X_3 cannot be the maximum expected utility of T.

$$S_{X,j} = \begin{cases} \max\{S_{X\cup\{i_{L-j+1}\},j-1}, S_{X,j-1}\} & (EU(X,T) < EU(X \cup \{i_{L-j+1}\},T)) \\ S_{X,j-1} & (EU(X,T) \geq EU(X \cup \{i_{L-j+1}\},T)) \end{cases}$$

Thus, we can greatly speed up the calculation of transaction maximum expected utility.

Definition 11. For itemset X and uncertain transaction database D, the transaction maximum expected utility of itemset X is MU:

$$MU(X) = \sum_{X \subset T \cap T \in D} MU(T)$$

Which means the sum of transaction maximum expected utility of transactions containing itemset X.

We can derive Lemma 1 through definitions above:

Lemma 1. The transaction maximum expected utilities of X's super-itemsets are not more than the transaction maximum expected utilities of X. For $X \subseteq X'$, always $MU(X') \leqslant MU(X)$.

According to Chernoff bound, suppose X_1, X_2..., X_n be independent random variables taking values in {0, 1}. Let X denote their sum and let $\mu = E[X]$ denote the sum's expected value. For any $\delta > 0$ it holds that

$$Pr(X > (1+\delta)\mu) < \left(\frac{e^{\delta}}{(1+\delta)^{(1+\delta)}}\right)^{\mu} (\delta > 0)$$

In the problem of mining probabilistic high utility itemsets, for one itemset X, it's appear in one uncertain transaction T can be seen as an independent Possion experiment, and the real support of X, i.e. the sup(X) is the sum of many Possion experiments, so the expect of that variable is the expected support count of X. The utility probability of X is:

$$P(sup(X) \times U(X) \geq minutil) = P(sup(X) \geq \frac{minutil}{U(I)})$$

When $esup(X) < minutil / U(X)$, we can let $(1+\delta)\, esup(X) = minutil / U(X)$, so we can get

$$\delta = \frac{minutil}{U(I)esup(I)} - 1 = \frac{minutil}{EU(I)} - 1$$

$$P\left(sup(X) \geq \frac{minutil}{U(X)}\right) < \left(\frac{e^{\delta}}{(1+\delta)^{(1+\delta)}}\right)^{esup(X)}$$

While $\delta = minutil / EU(X) - 1$. The right side of inequality 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 X, given uncertain transaction database D, utility threshold minutil and probabilistic utility threshold put, if $MU(I) < minutil$ and $\left(\frac{e^{\delta}}{(1+\delta)^{(1+\delta)}}\right)^{esup(X)} < put$, then itemset X and all of its super-itemsets cannot be utility.

Lemma 2 is key point to solve the "mining probabilistic high utility itemsets" problem in uncertain database, 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. 3. Because the transaction maximum expected utility of itemset G is 2.88, cannot pass the check in Lemma 2, so G and all of its' super itemsets cannot be utility super itemsets 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).

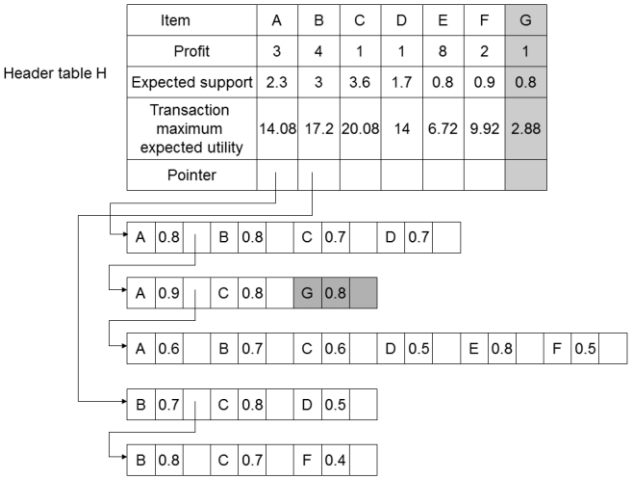


Figure 3. initial UUH-mine data structure after optimization

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 D, utility threshold minutil, probabilistic utility threshold put

Output: set of utility itemset *UIS*

1 UIS ← ∅ ;

2 HlnitializeHeader(D, minutil, put);

3 Recurision(H,D, 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 D 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., 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_X, uncertain transaction database D, utility threshold minutil, probabilistic utility threshold put, current set of utility itemset UIS

Output: updated set of utility itemsets *UIS*

1. **for each** *i* **in** H_I **do**
2. **if** $X \cup \{i\}$ is utility itemset do then
3. **UIS = UIS** $\cup (X \cup \{i\})$
4. **If** ChernoffCheck($X \cup \{i\}$) = true do **then**
5. create new header table $H_{X \cup \{i\}}$
6. Recursion(H, D, minutil, put, UIS)
7. End;

4. EXPERIMENTAL EVALUATION

In this section, we evaluate the performance of our UUIM algorithm and compare it with UUIM-NoCh algorithm. The experiments were performed on a 2.40 GHz Intel (R) Core (TM) i5 M450 Processor with 4GB main memory, and running on Windows 8.1. The algorithms are implemented in C++ language. Both synthetic and real datasets are used to evaluate the performance of the algorithms.

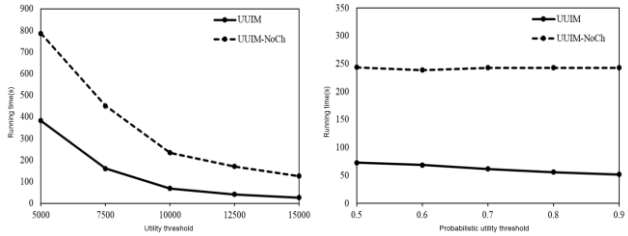
4.1 Databases

We use the classical transaction database Mushroom which comes from Audobon Society Field Guide and contains 8124 transaction and 120 different items.

4.2 Algorithms Efficiency Evaluation

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. 4(a) 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.



(a)Minimum Utility (b) Probabilistic Utility

Figure 4. Running Time vs. Threshold

Then we test the influence of the change of probabilistic utility threshold to UUIM algorithm and UUIM-NoCh algorithm. Fig. 4(b) 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.

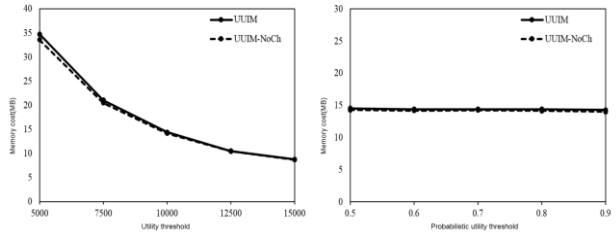
4.3 Memory Cost Evaluation

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. UUH-struct in UUIM algorithm need many header tables to store 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. 5(a) 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. 5(b) 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.



(a)Minimum Utility (b) Probabilistic Utility

Figure 4. Memory Cost vs. Threshold

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

5. ACKNOWLEDGMENTS

Our thanks to ACM SIGCHI for allowing us to modify templates they had developed.

6. REFERENCES

[1] Aggarwal, C. C., Li, Y., Wang, J., & Wang, J. (2009, June). Frequent pattern mining with uncertain data. In Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 29-38). ACM.

[2] Agrawal, R., & Srikant, R. (1994, September). Fast algorithms for mining association rules. In Proc. 20th int. conf. very large data bases, VLDB (Vol. 1215, pp. 487-499).

[3] Ahmed, C. F., Tanbeer, S. K., Jeong, B. S., & Lee, Y. K. (2009). Efficient tree structures for high utility pattern mining in incremental databases.Knowledge and Data Engineering, IEEE Transactions on, 21(12), 1708-1721.

[4] Bernecker, T., Kriegel, H. P., Renz, M., Verhein, F., & Zuefle, A. (2009, June). Probabilistic frequent itemset mining in uncertain databases. InProceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 119-128). ACM.

[5] Chui, C. K., & Kao, B. (2008). A decremental approach for mining frequent itemsets from uncertain data. In Advances in Knowledge Discovery and Data Mining (pp. 64-75). Springer Berlin Heidelberg.

[6] Chui, C. K., Kao, B., & Hung, E. (2007). Mining frequent itemsets from uncertain data. In Advances in knowledge discovery and data mining (pp. 47-58). Springer Berlin Heidelberg.

[7] Han, J., Pei, J., & Yin, Y. (2000, May). Mining frequent patterns without candidate generation. In ACM SIGMOD Record (Vol. 29, No. 2, pp. 1-12). ACM.

[8] Leung, C. K. S., Mateo, M. A. F., & Brajczuk, D. A. (2008). A tree-based approach for frequent pattern mining from uncertain data. In Advances in Knowledge Discovery and Data Mining (pp. 653-661). Springer Berlin Heidelberg.

[9] Liu, Y., Liao, W. K., & Choudhary, A. (2005). A two-phase algorithm for fast discovery of high utility itemsets. In Advances in Knowledge Discovery and Data Mining (pp. 689-695). Springer Berlin Heidelberg.

[10] Sun, L., Cheng, R., Cheung, D. W., & Cheng, J. (2010, July). Mining uncertain data with probabilistic guarantees. In Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining(pp. 273-282). ACM.

[11] Tong, Y., Chen, L., Cheng, Y., & Yu, P. S. (2012). Mining frequent itemsets over uncertain databases. Proceedings of the VLDB Endowment, 5(11), 1650-1661.

[12] Tseng, V. S., Wu, C. W., Shie, B. E., & Yu, P. S. (2010, July). UP-Growth: an efficient algorithm for high utility itemset mining. In Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 253-262). ACM.

[13] Wang, L., Cheng, R., Lee, S. D., & Cheung, D. (2010, October). Accelerating probabilistic frequent itemset mining: a model-based approach. InProceedings of the 19th ACM international conference on Information and knowledge management (pp. 429-438). ACM.

[14] Wu, C. W., Shie, B. E., Tseng, V. S., & Yu, P. S. (2012, August). Mining top-K high utility itemsets. In Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 78-86). ACM.

[15] Yao, H., Hamilton, H. J., & Butz, C. J. (2004, April). A Foundational Approach to Mining Itemset Utilities from Databases. In SDM (Vol. 4, pp. 215-221).

[16] Zaki, M. J. (2000). Scalable algorithms for association mining. Knowledge and Data Engineering, IEEE Transactions on, 12(3), 372-390.