



A Compressed Tree-based Approach for Frequent Item Sets Mining from Uncertain Data

B.Subbhulakshmi^{#1}, C.Hemavathi^{#2}

[#]Department of Computer Science and Engineering
Thiagarajar College of Engineering
Thirupparankundram, Madurai, Tamil Nadu
INDIA

Abstract: Over the past decade, there have been many studies on mining frequent item sets from precise data in which the presence and absence of items in transactions was certainly known. In some applications, the presence and absence of items in transactions are uncertain. The existential probabilities of these items are ranging from 0 to 1. To deal with these situations, a compressed tree based mining algorithm to find frequent patterns from imprecise data is proposed. UF-Tree (Uncertain Frequent Pattern Tree) method was introduced to construct the UF-Tree structure for mining frequent patterns. In UF-Tree structure, the same items with different probabilities are not merged together, thus causing many repeated nodes in the tree. In this work, a new structure called the compressed uncertain frequent-pattern tree (CUFP tree) is developed to efficiently keep the related information in the mining process. In CUFP-tree, the same item with different existential probabilities will be merged in the branch of tree. Performance of CUFP-Mine algorithm on real datasets shows the efficient results than UF-tree and U-Apriori algorithm in terms of execution time and number of node generation.

Keywords: Existential probability, Frequent Patterns, FP-Tree, CUFP-Tree, Data Mining, Uncertain Data.

I. Introduction

Most of the mining approaches are applied on certain database in which the occurrence of an item in transaction is certainly known. Apriori[3] algorithm was first proposed to mine the association rules from certain data. For deriving the frequent item sets efficiently, Han et al. proposed the frequent pattern tree (FP-Tree)[4] without generating candidate item sets. It shows that the better performance than the Apriori-like algorithms. Many of the algorithms based on the tree structure were also proposed to mine frequent patterns[8], [9].

In recent years, the problem of mining uncertain data sets has gained importance because of its numerous applications to a variety of problems [10], [11]. This is because data collection methodologies are often inaccurate and are based on incomplete information. Currently, a number of data mining problems have been studied in the context of uncertain data. In particular, the problems of indexing, clustering and outlier detection[14], [15] have been studied in the context of uncertain data.

Here are some examples of uncertain data arising in real-world applications, motivating the need for generic tools to manage such information. Information Extraction, Deduplication (The process of deciding automatically whether two data records represent the same real-world entity is often approximate), Privacy, Data integration and Various other sources of information such as human readings (e.g., bird species identification), predicting the future (e.g., weather forecasts, Customer purchase behaviours) and data from devices whose precision isn't perfect (e.g., sensors and RFID readers) [2] are inherently uncertain.

A. Uncertain Data Model

Most of uncertain data models use the possible world semantics to represent the uncertain relations among item [5]-[7].

Definition 1 An uncertain item is an item $x \in W$ whose presence in a transaction $t \in T$ is defined by an existential probability $P(x \in t) \in (0, 1)$. A certain item is an item where presence of an item x is either 0 or 1.

Definition 2 An uncertain transaction t is a transaction that contains uncertain items. A transaction database containing uncertain transactions is called an uncertain transaction database (D).

Let X be set of items or an item set consists of $X = \{x_1, x_2, x_3, \dots\}$. An item I in a transaction t may belong to two possible worlds W_1 (existence) and W_2 (non-existence). That is, x belongs to t in W_1 and x does not belong to t in W_2 . The probability for W_1 to be the true world is thus $P(x, t)$, and for W_2 is $1 - P(x, t)$, where $P(x, t)$ represents the existential probability of x in t . The probability of two items x_1 and x_2 to exist at the same time in a transaction t is thus $P(x_1, t) * P(x_2, t)$. In an uncertain database, the expected count of an item or an item set is used. The expected support of set of items X in all the transactions of uncertain database [6],[7] is then calculated as follows,

$$ExpSup = \sum_{i=1}^n (\prod_{x \in X} (P(x, t_i)))$$

Table I shows the simple medical transaction with 2 items, in which patient may be diagnosed to suffer from diseases with different possibilities (tuples), each of which contains set of symptoms (items). Symptoms may

appear in transactions with probabilities ranging from 0 to 1. The probability value may be derived from historical data.

Table I A simple medical dataset

Patient Name	Cough	Cold
Tom	65%	45%
Jerry	70%	60%

It may be seen from Table 1 that the two symptoms are highly correlated and may form a frequent 2-itemset for the disease. Another application of uncertain databases is pattern recognition from the satellite. For example, the features of a satellite image may be extracted to indicate whether a target is present or not. However, due to noise or a limited resolution of the image, the features are unclear and their appearance is represented by probabilities [1].

B. Uncertain Data Mining

In an uncertain database, an item set is considered as a frequent item set if its expected support exceeds the predefined minimum support. Uncertain datasets processing is quite different from that in certain datasets. Besides, when compared to the traditional FP tree for certain items, the degrees of item combinations in each transaction are not easily derived due to the multiplication operation.

An example is given here to show the processing of uncertain database. Consider the uncertain database shown in Table II. It contains 6 transactions and 6 items, denoted as a to f. The number behind an item in a transaction represents the existential probability of the item. For an uncertain database, an item set is frequent if its expected support is larger than or equal to the predefined minimum support threshold [5], [7]. For example, assume that the minimum support threshold is set to 30%.

Table II An Example for Uncertain Database

Transactions	Contents
t ₁	a(0.931), b(0.5), c(0.82), d (0.5)
t ₂	a(0.7),c(0.75), d(0.82)
t ₃	a(0.8), c(0.81),b(0.012)
t ₄	c(0.72), e(0.81), f(0.8)
t ₅	a(0.8), f(0.98), e(0.84)
t ₆	e(0.821), d(0.83), f(0.95)

The minimum count for an item set in Table II to be frequent is 6×0.3 which is 1.8. Let us take item b as an example to illustrate the idea. The expSup of the item 'b' is calculated as $0.5 + 0.012$, which is 0.512 and is smaller than the minimum count. Thus the item 'b' is not a frequent item in the uncertain database. For an item set with two or more items, the multiplication operation is used to calculate its expected support of item sets in an uncertain database. For example, the expected support for the 2-item set (a,c) is calculated as $(0.931 \times 0.82) + (0.7 \times 0.75) + (0.8 \times 0.81)$, which is 1.93642. Since the value is larger than the minimum count, the 2-item set (a,c) is thus considered as a frequent item set.

II. Related work

In this section, the existing algorithms for mining frequent patterns from uncertain database are briefly discussed.

A. U-Apriori

In order to solve the frequent pattern mining problem in uncertain data, the U-Apriori[5] algorithm was proposed which essentially mimics the Apriori[3] algorithm, except that it performs the counting by computing the expected support of the different item sets. The expected support of a set of items in a transaction is obtained by simply multiplying the probabilities of the different items in the transaction. This approach can be made further scalable by using the concept of data trimming. In the data trimming approach, those items with very low existential probability are pruned from the data. U-Apriori suffers from the following problems: (i) Inherited from the Apriori algorithm, U-Apriori does not scale well when handling large amounts of data because it also follows a level wise generate-and test framework. (ii) If the existential probabilities of most items within a pattern X are small, increments for each transaction can be insignificantly small. Consequently, many candidates would not be recognized as infrequent until most transactions were processed.

B. uFP-Tree

UF-Growth[7] algorithm is proposed for reducing the number of scans that are required to generate candidates at each step like apriori algorithm. At the same time, the number of nodes generated in each step is more in uncertain data, since each item is associated with different existential probability ranging from 0 to 1. The algorithm consists of two operations: (1) The construction of UF-tree (2) The mining of frequent patterns from

UF-tree. The UF-tree require a large amount of memory in order to increase the chance of path sharing, the data needs to be discretized and expected support of each tree node is rounded up to k decimal places.

Construction of uFP-Tree Algorithm

In the process of constructing the UF tree, the frequent 1-itemsets with their expected support larger than or equal to the minimum count were first found. Each transaction in the uncertain database was then updated to keep only the frequent 1-items. The construction process of the UF tree was a little like building an FP-tree except that the same items with different existential probabilities was put in different nodes. Only the same items with the same existential probabilities in transactions were merged together in the tree. As an example, the uncertain database in Table II is used to show how to construct the UF tree. Assume that, minimum support threshold is set to 30%. The frequent 1-items with their expected support are found as {(a: 3.231), (c: 3.1), (d: 2.15), (e: 2.471), (f: 2.73)} in are shown in Figure 1. Each of which has its expected support is larger than or equal to the minimum count 1.8. Each transaction in the uncertain database is then updated to remove the items which are not frequent. After that, the updated transactions are used to construct the UF tree tuple by tuple. Consider the first and the second transactions as examples to illustrate the process. Figure 1 shows UF-Tree construction process after updating first the two transactions as (a: 0.931, c: 0.82, d: 0.5) and (a: 0.7, c: 0.75, d: 0.82) respectively.

Figure 1 UF-Tree Construction after two transactions are processed

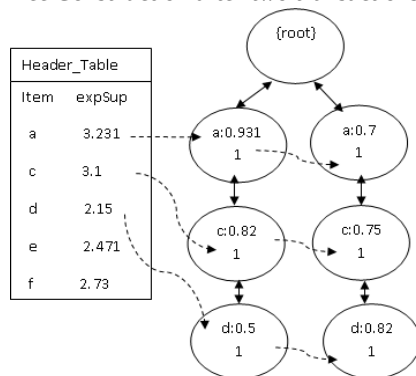
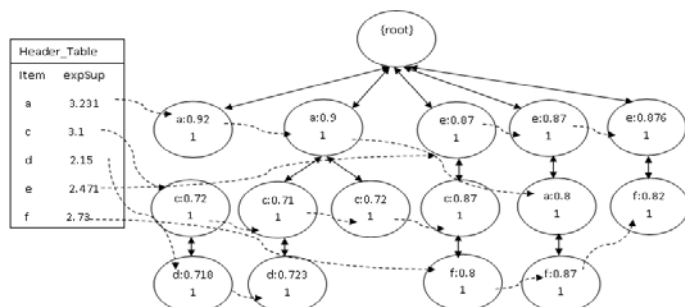


Figure 2 shows the resultant UF-Tree for the example uncertain database in Table II afte updating all transactions

Figure 2 UF-Tree after all the transactions are processed



It can be easily seen from the construction process that too many nodes are generated in a UF tree since it is hard for the same item in two transactions to have exactly the same probability, thus causing the tree to be not compressed at all and execution time of this mining algorithm is more. After the UF tree is constructed, the UF-tree mining algorithm called UFP-growth algorithm was used to derive the frequent item sets from it. Leung et al. proposed a k-decimal improvement to reduce the number of tree nodes in the UF tree [7]. Assume that, two nodes have the same k-digit value and after the decimal point, hence they are merged into one. However, the expected support of the item sets is not accurate after the k-decimal improvement is performed. Besides, the number of nodes is still large when k is not small.

III. Proposed Methodology

In this section, the proposed tree construction algorithm for building a compressed uncertain frequent pattern tree is described. Modules identified for this methodology are

- CUFP-Tree Construction Algorithm
- CUFP-Mine Algorithm

A. CUFP-Tree Constuction Algorithm

In this work, the CUFP tree structure is designed to efficiently store the related information in the mining process. This process uses the occurrence frequencies to sort the items in the transactions to construct the CUFP-tree

structures. The number of tree nodes can thus be reduced to form a more compressed tree structure than that of the UF-tree [7].

Notations

D The original uncertain database

n Total number of uncertain transactions

I An item or item sets

freq(I) Frequency of an item or item sets

L_1 Frequent-1 item set

μ Predefined minimum support threshold

expAry(I) An array attached to the node which keeps the expected support of super item sets

expSup(I) Expected support of an item or item sets

INPUT: D, μ

OUTPUT: A CUFP-Tree(Compressed uncertain frequent pattern tree)

Step1: $\text{expSup}(I) = \text{MinExpSup}(D, \mu)$ //Threshold values are given for calculating expected support of an item.
j=1;

Step2: for each number of transactions $n \geq ++j$ do

Step 3: for each frequent item sets $I \in L_1$ do

if $\text{expSup}(I) \geq n * s$

then I is an uncertain frequent-1 item sets

end for

Step 4: sort($L_1(I)$) with its frequency

Step 5: Build the Header_Table by keeping the 1-itemsets (items) in L_1 in the sorted order in STEP 3.

Step 6: Initially set the root node of the CUFP tree as root.

Step7: Remove the items not in L_1 from the transactions and sort the remaining items in each transaction according to the sorted order in step 4 and insert the sorted transactions into the CUFP-tree.

Step 7.1: calculate the expected support of the super-item sets of I in the path by the multiplication operation and add the values to the corresponding elements of the array (called expAry) of the node

Step 7.2: Otherwise, add the node with I to the end of the corresponding path and set the existential probability of I in the transaction as the value (expected support) in the node. Then, calculate the expected support of the super-item sets of I in the path by the multiplication operation and set the values to the corresponding elements of the array (called expAry) of the node.

An Example

The uncertain database in Table II is used as an example to illustrate the steps of proposed compressed tree-construction algorithm for first two transactions. Expected support for each node is calculated while constructing the CUFP-tree. The super-item set with its expected support is kept in the expAry of the node c and d

Figure 3 CUFP-Tree Construction after two transactions are processed

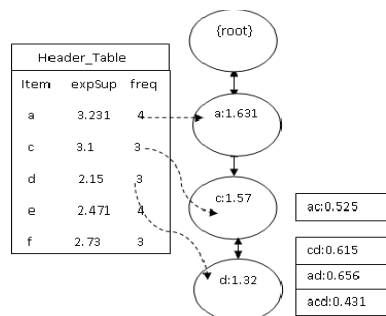


Figure 4 CUFP-Tree Construction after all transactions are processed

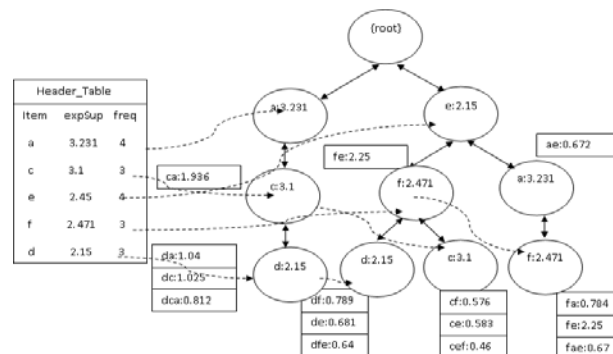


Figure 4 shows the final construction of CUFP-Tree after six transactions are processed. It can be observed from Figure 4 that, proposed CUFP tree has a more compressed and condensed structure than the UF tree. It is obvious that, proposed CUFP-tree algorithm ran faster than the UF-tree algorithm. Since each item in the transactions was attached with its calculated expected count of its super-item sets accumulated in array (expAry).

B. CUFP-Mine Algorithm

The proposed CUFP-mine algorithm is similar to the FP-growth[13] mining approach. But it is designed for uncertain frequent patterns.

INPUT: A constructed CUFP tree, its corresponding Header_Table, and a user-defined minimum support thresholds(μ)

OUTPUT: The uncertain frequent item sets

Step1: Process the items in the Header_Table one by one and bottom-up by the following steps.

Step2: Find all the nodes with the currently processed item I in the CUFP tree through the links.

Step3: Extract the item sets and their expected counts from the array of expAry(K) stored in each node K extracted in step2.

Step4: Sum the expected supports of the same item sets together.

Step5: If the summed expect support of an item set in step 4 is larger than or equal to the predefined minimum count ($n*s$), output it as an uncertain frequent item set.

Step6: Repeat steps 2–5 for another item until all the items in the Header_Table are processed.

An Example

The proposed CUFP-mine algorithm finds the uncertain frequent item sets from the CUFP tree given in Figure 4. Table III shows the uncertain frequent item sets after applying CUFP-mine algorithm.

Table III The final uncertain frequent item sets after CUFP-Mine algorithm

1-item sets		2-item sets	
Item	expCount	Item	expCount
e	2.219	ac	1.9364
b	2.72	ef	2.211
c	2.621		
d	2.563		
f	2.73		

IV. Performance Evaluation

The performance evaluation was carried out using chess dataset taken from the Frequent Item set Mining (FIMI) Repository (They are available from the <http://fimi.cs.helsinki.fi/data/>). First 2k tuples are considered to test the performance of proposed algorithm. Random existential probabilities from the range (0,1] were assigned to the items in the transactions. In the experiments, the minimum support thresholds were tuned to find the appropriate ranges in the limited memory. The minimum support threshold is varied from 58% to 60%, by increasing it 0.5% each time. The CUFP-Mine algorithm was implemented using java programming language and compiled using Eclipse IDE in Windows 7Ultimate. PC is equipped with a Pentium 2.20 GHz CPU and 2GB of RAM memory.

A. Accuracy

Accuracy can be examined by using recall and precision measure [16] which quantifies the false positives and negatives. Let frequent item sets by UF-Tree be the $F_{UF-Tree}$ and frequent item sets by the CUFP-Tree be the $F_{CUFP-Tree}$. The recall(r) and the precision (p) of CUFP-Tree relative to UF-Tree are defined as follows:

$$r = \frac{|F_{UF-Tree} \cap F_{CUFP-Tree}|}{|F_{UF-Tree}|}$$

$$p = \frac{|F_{UF-Tree} \cap F_{CUFP-Tree}|}{|F_{CUFP-Tree}|}$$

The frequent item sets which are retrieved by the CUFP-Tree mining resembles closely to those of UF-Tree[7].

B. Evaluation on CUFP-Mine algorithm

From Figure.5, it is obvious to see that the proposed CUFP-tree algorithm runs faster than the UF-tree[7] and U-Apriori[5] algorithm.

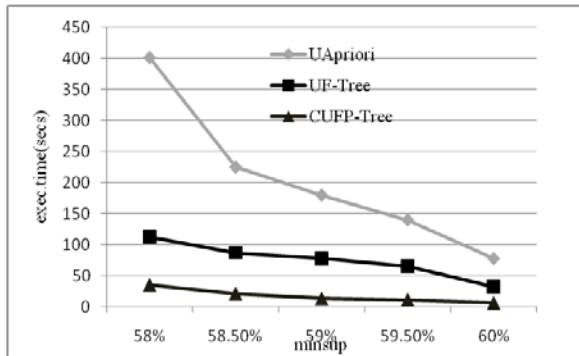


Figure 5 Efficiency of CUFP-Mine algorithm

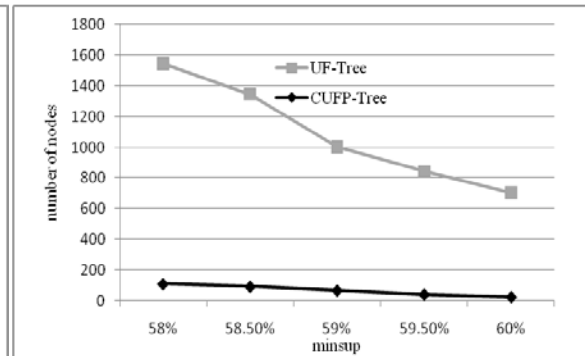


Figure 6 Number of nodes generation in CUFP-Mine algorithm

From Figure 6, The number of nodes of two algorithms for different minimum support thresholds is greatly reduced in CUFP-Tree than UF-Tree[7] in the chess dataset.

V. Conclusion

The mining process in an uncertain database is more complex than that in a certain database. In this work, CUFP tree structure is proposed to efficiently mine from an uncertain database in which items may appear in transactions with existential probabilities ranging from 0 to 1. By creating additional array (expAry) for each item in uncertain transactional database to keep the information of its super-item sets in CUFP-mine, redundant node generation and the computational cost in the mining process is greatly reduced. It gives the better performance than other existing algorithms on uncertain database.

The proposed approach will be useful, when mining problem sets the constraint (minsap) of only at most a fixed number of items in an item sets. However, if the length of the frequent item sets in a transaction is large and no constraint is set, then the proposed approach is not so suitable. In the future, other data structures and good estimation strategies for solving the non-constraint based problem can be designed.

References

- [1] Dai, X., Yiu, M. L., Mamoulis, N., Tao, Y., & Vaitis, M. (2005). Probabilistic spatial queries on existentially uncertain data. In The 11th international symposium on spatial and temporal databases (pp. 400–417).
- [2] Khoussainova, N., Balazinska, M., & Suciu, D. (2006). Towards correcting input data errors probabilistically using integrity constraints. In MobiDE.
- [3] Agrawal, R., Imielinski, T., & Swami, A. (1993). Mining association rules between sets of items in large databases. In International conference on management of data (pp. 207–216).
- [4] Han, J., Pei, J., Yin, Y., & Mao, R. (2004). Mining frequent patterns without candidate generation: A frequent-pattern tree approach. *Data Mining and Knowledge Discovery*, 8, 53–87.
- [5] Chui, C. K., Kao, B., & Hung, E. (2007). Mining frequent itemsets from uncertain data. *Advances in Knowledge Discovery and Data Mining*, 4426, 47–58.
- [6] Leung, C. K. S., Carmichael, C. L., & Hao, B. (2007). Efficient mining of frequent patterns from uncertain data. In The 7th IEEE international conference on data mining workshops (pp. 489–494).
- [7] Leung, C. K. S., Mateo, M. A. F., & Brajczuk, D. A. (2008). A tree-based approach for frequent pattern mining from uncertain data. *Lecture Notes in Computer Science*, 5012, 653–661.
- [8] Han, J., Pei, J., & Yin, Y. (2000). Mining frequent patterns without candidate generation. In SIGMOD Conference.
- [9] Agarwal, R., Aggarwal, C., & Prasad, V. V. V. (2001). A tree projection algorithm for generation of frequent itemsets. *Journal of Parallel and Distributed Computing*, 61, 350–371.
- [10] Deshpande, A., Guestrin, C., Madden, S., Hellerstein, J., & Hong, W. (2004). "Model-Driven Data Acquisition in Sensor Networks," *Proc. 13th Int'l Conf. Very Large Data Bases (VLDB)*.
- [11] Sistla, P., Wolfson, O., Chamberlain, S., & Dao, S. (1998). *Querying the Uncertain Position of Moving Objects*. Temporal Databases: Research and Practice, Springer Verlag.
- [12] Dalvi, N., & Suciu, D. (2004). Efficient query evaluation on probabilistic databases. In *Very Large Database (VLDB)*.
- [13] Cheng, R., Kalashnikov, D. V., & Prabhakar, S. (2003). Evaluating probabilistic queries over imprecise data. In *ACM SIGMOD international conference on management of data* (pp. 551–562).
- [14] Cheng, R., Kalashnikov, Cheng, R., Xia, Y., Prabhakar, S., Shah, R. & Vitter, J. (2004). Efficient Indexing Methods for Probabilistic Threshold Queries over Uncertain Data in *Very Large Database (VLDB) Conference Proceedings*.
- [15] Cormode, G., and McGregor, A. (2008). Approximation algorithms for clustering uncertain data, in *PODS Conference*, pp. 191-200
- [16] Van Rijsbergen, C. J. (1979) *Information Retrieval*. Butterworth.