An Improved Approach for Mining Frequent Itemsets from Uncertain Data using Compact Tree Structure

Sapna Saparia[1], Dr.Shyamal Tanna[2]

[1] PG Student, Information Technology, LJIET, Ahmedabad, Gujarat, India
[2] Assistant Professor, Information Technology, LJIET, Ahmedabad, Gujarat, India

ABSTRACT

There are too many existing algorithms proposed that mines frequent patterns from some or precise data. But now a day of requirement for uncertain data mining is increased. There are a lot of actual situations in which data are uncertain, where the mining industry is necessary. For reason of frequent mining from uncertain data mainly two approaches are proposed that are step by step and diagram approach to growth. Level approach rational use to generate and test strategy. U-apriori algorithm is the example of the step-by-step approach. Growth approach-pattern uses tree as the structure of the format. The UF-algorithm growth, PFU-algorithm of growth, algorithm of mines CUFP, PUF-algorithm for growth are the example of the real world of Pattern approach to growth. We take here the study of algorithms that are used at the mine frequent patterns from uncertain data to maintain the data generated from real-world applications.

Keyword : - *Frequent Pattern Mining, Data Mining Algorithms, Expected Support, Frequent Patterns, Tree Structures, and Uncertain Data*

1. Introduction

As an important task of extraction of data, often the mining aims to discover implied, unknown and potentially knowledge useful revealing patterns on the collections of frequently co-produce of the elements, objects or events that are integrated into the data. Of our days, often mining is commonly used in various companies in real life, the government, and science (e.g., banking services applications, bioinformatics, the modeling environment, finance, marketing, medical diagnostics, analysis of meteorological data). Uncertain data are present in a large number of these applications. The uncertainty can be caused by (i) Our limited perception or understanding of the reality; (ii) limitations of the equipment of observation; or (iii) the limitations of available resources for the collection, storage, processing or analysis of the data. It may also be of inherent nature .Data collected by chemical, electro magnetic, mechanical, optical radiation, thermal sensors [1] In monitoring the environment, security and manufacturing systems can be noisy. Dynamic errors-such as (i) inherited measurement errors, (ii) the frequency of sampling of sensors, (iii) deviation caused by a rapid change (e.g., drift, noise) of the measured property over time, (iv) the errors of wireless transmission, or (v) the network latencies also introduces the uncertainty in the data reported by these sensors. In addition, there is also the uncertainty in the survey data and the uncertainty due to granularity of the data (e.g., city, province) in the field of taxonomy. Missing data disguised, which are not explicitly represented as such but rather appear as potentially of data values, valid also introduce some uncertainty. In addition, in preserving the confidentiality of the data of applications , the sensitive data may be intentionally blurred via the aggregation or disruption in order to preserve the anonymity of the data. All these sources of uncertainty gives birth to enormous quantities of uncertain data in applications in the current life [1].

2. LITRACURE REVIEW

I. UF-growth

To mine frequent patterns from probabilistic datasets of uncertain data, [2] proposed a tree-based mining algorithm called **UF-growth**. Similar to its counterpart for mining precise data (the FP-growth algorithm), UF-growth also constructs a tree structure to capture the contents of the datasets. However, it does not use the FP-tree (as in FP-growth) because each node in the FP-tree only maintains (i) an item and (ii) its occurrence count in the tree path. When mining precise data, the actual support of an pattern X depends on the occurrence counts of items within X. However, when mining uncertain data, the expected support Xof X is the sum of the product of the occurrence count and existential probability of every item within X. Hence, each node in the **UF-tree** consists of three components: (i) an item, (ii) its existential probability, and (iii) its occurrence count in the path. Such a UF-tree is constructed in a similar fashion as the construction of the FP-tree, except that a new transaction is merged with a child node only if the same item and the same existential probability exist in both the transaction and the child node. As such, it may lead to a lower compression ratio than the original FP-tree. Fortunately, To reduce the memory consumption, UF-growth incorporates two improvement techniques.. The first technique is to discretize the existential probability of each node (e.g., round the existential probability to k decimal places such as k=2 decimal places), which reduces the potentially infinite number of possible existential probability values to a maximum of 10k possible values. The second improvement technique is to limit the construction of UF-trees.

II. UFP-growth

To make the tree more compact by reducing the tree size (via a reduction in the number of tree nodes) proposed the **UFP-growth algorithm**. Like UF-growth, the UFP-growth algorithm also scans the probabilistic dataset of uncertain data twice and builds a **UFP-tree**. As nodes for item x having similar existential probability values are clustered into a mega-node, the resulting mega node in the UFP-tree captures (i) an item x, (ii) the maximum existential probability value (among all nodes within the cluster), and (iii) its occurrence count (i.e., the number of nodes within the cluster). Tree paths are shared if the nodes on these paths share the same item but similar existential probability values. In other words, the path sharing condition is less restrictive than that of the UF-tree. At the same time, due to the approximate nature (e.g., caused by the use of the maximum existential probability value among all the nodes clustered into a mega-node) of UFP-growth, UFP-growth also finds some infrequent patterns (i.e., some false positives) in addition to those truly frequent patterns (i.e., true positives). Hence, a third scan of the probabilistic dataset of uncertain data is then required to remove these false positives.

III. PUF- Tree: A Compact Tree Structure for Frequent Pattern Mining of Uncertain data

Along this direction, Leungand Tanbeer observed that (i) the transaction cap provides CUF-growth with an upper bound to expected support of patterns and (ii) such an upper bound can be tightened in a tree-based environment. They introduced the concept of a prefixed item cap, which can be defined as follows[2]**.**

Definition 14.3 The **prefixed item cap**—denoted by I Cap(xr , ti)—of an item xr in a transaction ti = {x1, . . . , xr , . . . , xh}, where 1 ≤ r ≤ h (i.e., h=|ti | represent the length of ti), is defined as the product of P(xr , ti) and the highest existential probability value M of items from x1 to xr−1 in ti (i.e., in the proper prefix of xr in ti). More formally,

P Icap(xr , ti) = P(xr , ti) ×M if |ti | > 1
P(x1, ti) if|ti| = 1 (i.e., ti={x1})
Where M = maxq∈[1,r−1]P(xq, ti).

Assume that items are arranged in the order _a, b, c, d, e_ from the root to leaves. Then, Table shows the prefixed item cap for every item in a transaction in a probabilistic dataset D2 of uncertain data. Fig. shows for how these prefixed item caps are captured in a new tree structure called **PUF-tree**, from which the corresponding algorithm called **PUF-growth** mines uncertain frequent patterns. Like UFP-growth and CUF-growth, the PUF-growth algorithm also takes three scans of the probabilistic dataset of uncertain data to mine frequent patterns. With the first scan, PUF-growth computes the prefixed item caps. With the second scan, PUF growth builds a PUF-tree to capture (i) an item and (ii) its corresponding prefixed item cap. [2]

Like those in CUF-tree, paths in the PUF-tree are shared if the nodes on these paths share the same item. Hence, the resulting PUF-tree is of the same size as the CUF-tree (also for capturing uncertain data), which can be as compact as the FP-tree (for capturing precise data). The header table associated with the PUF-tree gives the expected support of frequent 1-itemsets (i.e., singleton patterns or frequent items). The prefixed item caps in the PUF-tree provide upper bounds to the expected support of k-itemsets (for $k \geq 2$). For any k-itemset X, if the upper bound to its expected support is less than minsup, then X can be safely pruned.

By extracting appropriate tree paths and constructing PUF-trees for subsequent projected databases, PUF-growth finds all potentially frequent patterns at the end of the second scan of the probabilistic dataset of uncertain data. As these potentially frequent patterns include all truly frequent patterns and some infrequent patterns.

PUF-growth then quickly scans the dataset a third time to check each of them to verify whether or not they are truly frequent (i.e., prune false positives). As illustrated by Table shows, the prefixed item caps tighten the upper bound to the expected support of non-singleton patterns (when compared with the transaction caps in the CUF-tree). Consequently, the number of false positives that need to be examined by PUF-growth during the third scans of the probabilistic dataset[2] of uncertain data is usually smaller than that by CUF-growth. Hence, PUF-growth runs faster than CUF-growth.

IV. Fast Algorithms for Frequent Itemset Mining from Uncertain Data

Here we discus about Frequent itemset mining algorithms, called tube-growth to find all and only those frequent itemsets (i.e., no false negatives and no false positives) from tube-trees capturing uncertain data[6].

Tube S-growth Algorithm

Tube S-growth algorithm first constructs a tree structure, called tube S-tree, to capture important information of uncertain data.[6] Specifically, the algorithm scans an uncertain database to find all distinct frequent items (i.e., every domain item xi with expSup (xi) ≥ minsup). As expSup satisfies the downward closure property (i.e., if expSup (X) < minsup , then expSup(X) < minsup for all X⊒ X), infrequent items can be safely removed. Then, the algorithm scans the uncertain database a second time to insert each database transaction into the tubeS-tree in a fashion similar to that of the FP-tree. For example, “prefixes” of two tree paths are merged if they share the same items. A key difference is that, when inserting a frequent item xi ∈ tj, we compute and capture both its IC value and M2 value in the node for xi. Infrequent items in any transaction are omitted and not inserted into the tree[6]. If a node containing xi already exists in a tree path p, we update the IC and M2 values by (i) adding its IC(xi,t j) into the existing IC(xi,p) and (ii) taking the maximum between its M2(xi,t j) and the existing M2(xi,p), as described in Specifically, for each frequent domain item xi the algorithm constructs an {xi}-conditional tree by extracting all relevant tree paths (from xi to the root) and passing both IC(xi,p) and M2(xi,p). By doing so, the algorithm computes tube S for every 2-itemset {y,xi} (e.g., {c,d}) where y is a frequent item located above xi (i.e., closer to the root) on a tree path. If tube S ({y,xi}) ≥ minsup (i.e., {y,xi} is potentially frequent), then the algorithm performs a similar mining step by constructing a {y,xi}-conditional tree to mine potentially frequent 3-itemsets, and so on. After finding all potentially frequent itemsets (including true positives and false positives), tube S-growth algorithm scans the uncertain database a third time to verify whether or not if a potentially frequent itemset is truly frequent.

Tube P-growth Algorithm

In contrast, tube P-growth algorithm constructs its corresponding tree structure, called tube P-tree, in a similar fashion to that for the tube S-growth algorithm, except that P(xi ,t j) is captured instead of the M2(xi, t j) value .Once the tube P-tree structure is constructed, our tube P-growth algorithm[6] then recursively mines frequent item sets from the tube P-tree in a similar fashion as our tube S-growth algorithm. For each frequent domain item xi , the tube P-growth constructs an {xi}-conditional tree by extracting all relevant tree paths and passing only IC(xi ,p) but not P(xi ,p) because tube P for any 2-itemset {y, xi} is computed using P(y, p) instead of P(xi, p) where y is a frequent item located above xi on a tree path. Similar to tube S-growth, our tube P-growth algorithm also scans the uncertain data a third time to remove all false positives [6].

V. DISC: Efficient Uncertain Frequent Pattern Mining with Tightened Upped Bounds

UF-growth is a tree-based algorithm for mining frequent patterns from uncertain data. While it directly calculates the expected support of a pattern, it requires a significant amount of storage space to capture all existential probability values among the items. To eliminate the extra space requirement of UF-growth, the CUF-growth algorithm combines nodes with the same item by storing an upper bound on expected support. In this paper, we (i) introduce a new concept of domain item specific capping (DISC) and (ii) propose three new scalable data analytics algorithms that use this concept to achieve a tighter upper bound than CUF-growth. Experimental results show the effectiveness of uncertain frequent pattern mining with tightened upper bounds provided by using the concept of DISC.1) the concepts of domain item-specific capping (DISC) for tightening the upper bound on expected support, DISC-trees for capturing the contents of uncertain databases, and algorithms that use the DISC-trees for mining frequent patterns.

To tighten the upper bound, we introduce the concept of domain item specific capping (DISC). As its name suggests, DISC involves having caps specific to each domain item yi in a given transaction tj = {y1,..., yh}. The key idea is that, instead of multiplying the highest existential probability value M1 in tj by the second highest existential probability value M2 in tj , DISC multiplies the existential probability value P(yi ,tj) of the domain item yi by M1 in tj. In other words, let (i) M1(tj) = maxq P(yq,tj) be the highest existential probability value among all h items in tj, and (ii) yg = arg maxq P(yq,tj) be the item having M1(tj) so that M1(tj)=P(yg,tj). Then, for tree-based mining, we build a tree structure such that each tree path represents some transactions in uncertain data. In a tree path representing a single transaction, each tree node keeps (i) an item yi in tj and (ii) its disc(yi ,tj): disc(yi,tj) ={ P(y1,tj) if h=1 P(yi,tj)×M1(tj) if h≥2 With the information yi disc(yi,tj) stored at each tree node, DISC provides an upper bound to expected support disc(X,tj) for k-itemset X={x1,...,xk} ⊆ tj={y1,...,yr,...,yh} where xk=yr and k≥2 by using disc(X,tj) = disc(xk,tj) = P(yr,tj)×M1(tj).

VI. NEW ADAPTATIONS OF CLASSIC ALGORITHM FOR MINING FREQUENT ITEMSETS FROM UNCERTAIN:

Tree structures (e.g., UF-trees, UFP-trees) corresponding to many existing uncertain frequent pattern mining algorithms can be large. Other tree structures for handling uncertain data may achieve compactness at the expense of loose upper bounds on expected supports. To solve this problem, we propose a compact tree structure that captures uncertain data with tighter upper bounds than the Consider three classic frequent itemsets mining algorithms (Apriori, FP-growth, and H-mine) and their adaptions. Although Aproiri algorithm is slower than the other two algorithms in deterministic databases, with the well-known downward closure property, UApriori algorithm performs very well among the three algorithms and is usually the fastest one in uncertain databases with high min_esup. However, FP-growth algorithm does not show similar efficient behavior in the uncertain cases.

There are some possible reasons: Firstly, the probabilities of items in uncertain database make the database sparse due to fewer shared nodes and paths. Secondly, in the uncertain cases, the compression of UFP-tree is substantially reduced because it is hard to take the advantage of shared prefix path in FP-tree. Therefore, as the support threshold goes down, the UFP-trees constructed become large and too many candidate itemsets generated, which leads to sharp increase of memory usage. So the FP-tree structure does not extend well in the sparse uncertain databases.

Based on the divide-and-conquer framework and the depth-first search strategy, UH-mine is quite suitable for sparse uncertain databases. As an extension from classical algorithm in deterministic case, UH-mine algorithm fails to compress the data structure and use the dynamic frequency order sub-structure, so it is faster in building head tables of all levels than building all conditional sub trees. Therefore UH-mine always outperforms other algorithms in uncertain sparse databases with low min_sup.

VII. BLIMP: A Compact Tree Structure for Uncertain Frequent Pattern Mining

The corresponding algorithm mines frequent patterns from this compact tree structure. It shows compactness of our tree structure and the tightness of upper bounds to expected supports provided by our uncertain frequent pattern mining algorithm[5].
A branch-level item prefixed-cap tree (BLIMP-tree), which can be as compact as the original FP tree and PUF-tree; and a mining algorithm (namely, BLIMP-growth), which finds all frequent patterns from uncertain data.

To tighten the upper bound for all k-item sets (k>2), we propose a branch-level item prefixed-cap tree structure (BLIMP-tree)[5]. The key idea is to keep track of a value calculated the maximum of all existential probabilities for the single item represented by that node. Every time a frequent extension is added to the suffix item to form a k-itemset (where k>2), this "blimp" value will be used. Hence, each node in a BLIMP-tree contains: (i) an item xr, (ii) an item cap ICap(xr,t j) and (iii) a "blimp" value, which is the maximum existential probability of xr in tj. Fig. 1(c) shows the contents of a BLIMP-tree for the database. With this information, BLIMP-trees give a tightened upper bound on the expected support of an itemset by the product of ICap (xr,t j) and the "BLIMP" values in the prefix of xr. This new compounded item cap of any k-itemset X = {x1, x2,...,xk} in a tree path tj = x1,x2,...,xh(denoted as I(X,tj) where xk = xr) can be defined as follows. Let tj = x1,x2,...,xr,...,xhbe a path in a BLIMP-tree, where h = |tj| and r ∈ [1,h]. Let Mxi denote the highest existential proba- bility of xi in the prefix of xr in tj. IfX = {x1,x2,...,xk} is a k-itemset in tj such that xk = xr, its compounded item cap

I(X,tj) is defined as follows:
I(X,tj)= Icap(xr,t j) if k ≤ 2
ICap(xr,t j)×k−2 i=1 Mxi if k ≥ 3
Where ICap(xr,t j) is the prefixed item cap

To construct a BLIMP-tree, we scan the transactional database of uncertain data to compute the expected support of every domain item. Any infrequent items are removed. Then, we scan the database a second time to insert each transaction into the BLIMP-tree. An item is inserted into the BLIMP-tree ac- cording to a predefined order. If a node containing that item already exists in the tree path, we (i) update its item cap by summing the current ICap (xr,t j) with the existing item cap value and (ii) update its "blimp" value by taking the maximum of the current P(xr,t j) with the existing "blimp" value. Otherwise, we create a new node with ICap(xr,t j) and P(xr,t j) (i.e. the initial "blimp" value). For a better understanding of BLIMP-tree construction [5].

Table 1 : The comparison Study of Different Algorithm

Sr no	**Algorihtm**	**Advantage**	**Disadvantage**
1	U-Apriori algorithm	This algorithm greatly reduces the size of candidate set.	It scans database many times and thus performance is affected.
2	UF-growth algorithm	This algorithm uses UF-trees to mine frequent patterns from uncertain databases in two database scans	It contains a distinct tree path for each distinct item, existential probability pair.
3	UFP growth algorithm	This algorithm scans the database twice, As nodes for item having similar existential probability values are clustered into a meganode, the resulting mega-node in the UFP-tree.	It contains a distinct tree path for each distinct item, existential probability pair
4	UH mine algorithm	This algorithm Stores all frequent items in each database transaction in a hyper-structure called UH-struct.	It Suffer from the high computation cost of calculating the expected support.
5	PUF groth algorithm	This algorithm Mines frequent pattern with constructing a projected database for each potential frequent pattern and recursively mine its potential frequent extensions.	It creates some false positives.

4. PROPOSED METHOD.

4.1 Introduction to Proposed Algorithm Design:

In proposed algorithm uncertain database , minimum support to find a frequent pattern items. Scan an uncertain database to find frequent itemset.if expected support x_i> minimum support then scan an uncertain database in second time to insert each transaction into the tree. for inserting any item into the tree check the value of item. If k=1, then $p(x_i,t_j)$ If k=2, then, If $x_{k=}y_g$ then $p(x_k,t_j) * M_2(t_j)$ Else $x_{k\neq}y_g$ then $p(x_k,t_j) * M_1(t_j)$ If $k\geq3$, then $p(x_k,t_j) * M_i(t_j) * M_{1+1}(t_j)$.once a tree structure is constructed recursively mines frequent Itemset in uncertain data. It construct a condition by extracting all relevant tree path and passing the values .all trans action of database scan is completed than get a frequent item set.

4.2 Proposed Algorithm:

```
Input:      Uncertain Database, min support
Output:     Frequent Patterns
Step: 1  Scan an Uncertain database to find frequent 1 itemset (expsup $x_i \geq$ minsup)
Step: 2  Scan an Uncertain database second times to insert each database transaction into the tree.
Step: 3     For inserting item into tree
            1). If k=1, then $p(x_i,t_j)$
            2). If k=2, then, If $x_{k=}y_g$ then $p(x_k,t_j) * M_2(t_j)$
                        Else $x_{k\neq}y_g$ then $p(x_k,t_j) * M_1(t_j)$
            3). If $k\geq3$, then $p(x_k,t_j) * M_i(t_j) * M_{1+1}(t_j)$

Step: 4  Once tree structure is constructed recursively mines frequent itemset from the proposed algorithm.
Step: 5  Each frequent domain item $x_i$ the algorithm constructs conditional tree by extracting all relevant tree path
and passing the values.
Step: 6  If Itemset $I^{cap}(x_i,t_j) \geq$ minsup then algorithm performs a similar mining step by constructing condition tree
in frequent B Itemset.
Step: 7     Get frequent itemset
Step: 8     Stop
```

4. EXPERIMENTAL RESULTS

Eclipse is used for compiling and executing purpose. Eclipse is an integrated development environment (IDE) with using SPMF tools. It contains a base workspace and an extensible plug-in system for customizing the environment which mostly written in Java. Experiment was carried out on real-life datasets having varied characteristics. Those datasets are Mashroom and IBM. In experiment the proposed system is compared with the existing algorithm. We consider two parameters execution time and memory required to justify our propo sed algorithm.

Table 2 : The comparison of the Different Dataset with Different dataset

Approach	Mashroom Support = 0.25		IBM Support = 0.03	
	A (ms)	B (MB)	A (ms)	B (MB)
Base Paper	6.9528	22742	7701	15.0253
Proposed Algorithm	2291	114.67	7392	8.0005

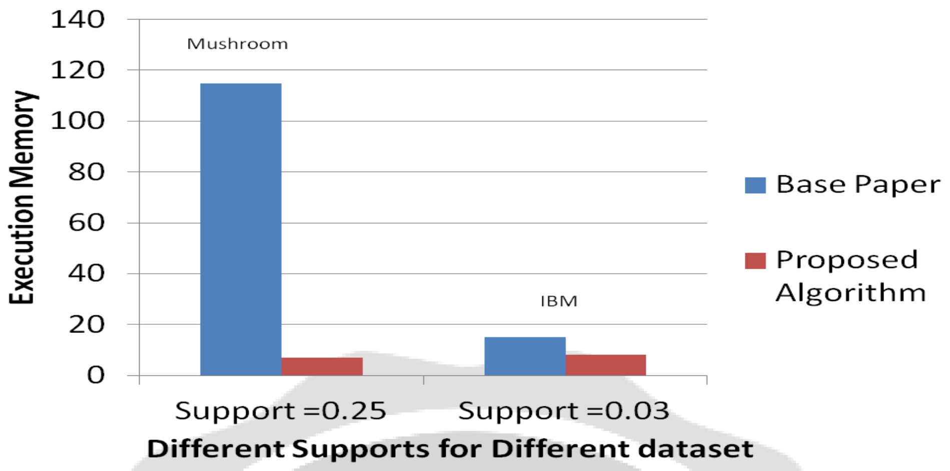

Chart 1: Comparision of Execution time for Different Dataset

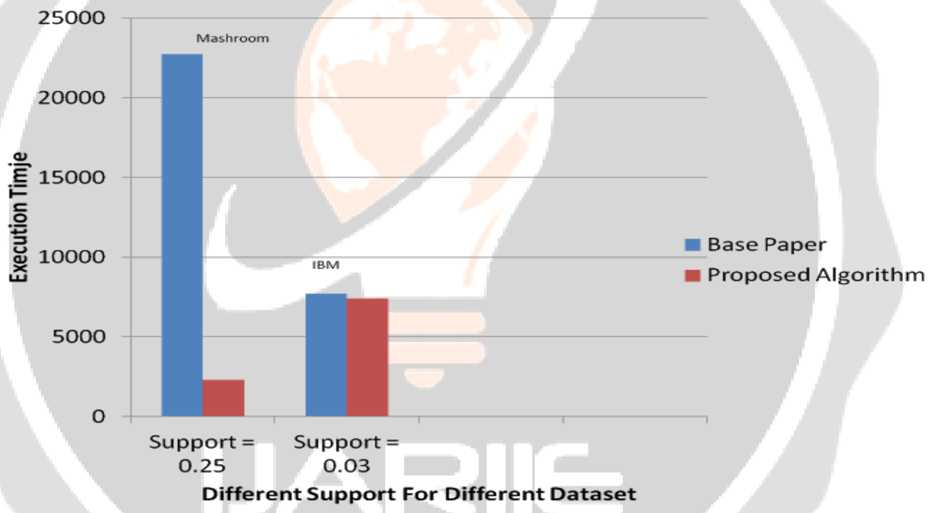

Chart 2: Comparision of Execution time for Different Dataset

5. CONCLUS IONS.

In this document, the detailed study of the algorithms of exploration of pattern of frequent uncertain data is made and identified many of the strengths and weaknesses of each. New variants of the existing algorithms are compared with the algorithms of the mining and classic tours to significant benefits and limitations. This comparison can also take place in various questions of optimization that will evolve for best performance. The effectiveness of mining is no longer of algorithms obstacle but still there is a need to develop methods to obtain excellent results.

6. REFERENCES

1. Carson Kai-Sang Leung "Uncertain Frequent Pattern Mining" C. C. Aggarwal, J. Han (eds.), Frequent Pattern Mining Springer International Publishing Switzerland 2014
2. Leung, C.K.-S.and S.K.Tanbeer. "PUF-Tree: A Compact tree structure for frequent pattern mining of uncertain data." In: J. Pei, T., Suzuki, E., Ting, K.M., Inokuchi, A. (eds.) PAKDD 2013. LNCS (LNAI), vol. 7818, pp. 13–25. Springer, Heidelberg (2013).

3. Kai-Sang Leung, Mark Anthony F. Mateo, Dale A. Brajczuk- "A Tree-Based Approach for Frequent Pattern Mining from Uncertain Data", Springer-Verlag Berlin Heidelberg 2008.
4. Radhika Ramesh Naik, Prof. J.R.Mankar- "Mining Frequent Itemsets from Uncertain Databases using probabilistic support", International Journal -2013.
5. Carson Kai-Sang Leung and Richard Kyle MacKinnon- "BLIMP: A Compact Tree Structure for Uncertain Frequent Pattern Mining." Springer International Publishing Switzerland 2014.
6. Carson Kai-Sang Leung, Richard Kyle MacKinnon, Syed K. Tanbeer- "Fast Algorithms for Frequent ItemSet mining from Uncertain Data", IEEE International Conference-2014
7. Tan, P.-N., Chawla, S., Ho, C.K., Bailey, J. (eds.) PAKDD 2012, Part II. LNCS (LNAI), vol. 7302, pp. 322–334. Springer, Heidelberg (2012)
8. L. Wang, R. Cheng, S. D. Lee, et.al, "Accelerating probabilistic frequent itemset mining: a model-based approach, " In CIKM'10, Toronto, Ontario, Canada, pp.429–438, 2010.
9. Toon Calders, Calin Garbini, Bart Goethals, "Approximation of Frequentness Probability of Itemsets in Uncertain Data, " in ICDM, pp.749-754, 2010
10. Calders, T., Garboni, C., Goethals, B.: Approximation of frequentness probability of itemsets in uncertain data. In: IEEE ICDM 2010, pp. 749–754 (2010) .
11. Calders, T., Garboni, C., Goethals, B.: Efficient pattern mining of uncertain data with sampling. In: Zaki, M.J., Yu, J.X., Ravindran, B., Pudi, V. (eds.) PAKDD 2010, Part I. LNCS (LNAI), vol. 6118, pp. 480–487. Springer, Heidelberg (2010).
12. Jiang, F., Leung, C.K.-S.: Stream mining of frequent patterns from delayed batches of uncertain data. In: Bellatreche, L., Mohania, M.K. (eds.) DaWaK 2013. LNCS, vol. 8057, pp. 209–221. Springer, Heidelberg (2013)
13. B. Dong, R. Liu, and W. H. Wang, "Integrity verification of outsourced frequent itemset mining with deterministic guarantee," in Proc. IEEE ICDM 2013, pp. 1025–1030.
14. H. Liu, P. LePendu, R. Jin, D. Dou, "A hypergraph-based method for discov- ering semantically associated itemsets. ," in Proc. IEEE ICDM 2011, pp. 398– 406.
15. J. Liu, K. Wang, B. C. M. Fung, "Direct discovery of high utility itemsets without candidate generation," in Proc. IEEE ICDM 2012, pp. 984–989.
16. H. Liu, P. LePendu, R. Jin, D. Dou, "A hypergraph-based method for discov- ering semantically associated itemsets. ," in Proc. IEEE ICDM 2011, pp. 398– 406. [15] J. Liu, K. Wang, B. C. M. Fung, "Direct discovery of high utility itemsets without candidate generation," in Proc. IEEE ICDM 2012, pp. 984–989.
17. C. K.-S. Leung, "Uncertain frequent pattern mining," in Frequent pattern mining, pp. 417-453, Oct. 2014.
18. [C. K.-S. Leung, P. P. Irani, and C. L. Carmichael, "WiFIsViz: effective visualization of frequent itemsets," in Proc. IEEE ICDM 2008, pp. 875–880.
19. C. K.-S. Leung and Q. I. Khan, "DSTree: a tree structure for the mining of frequent sets from data streams," in Proc. IEEE ICDM 2006, pp. 928–932.
20. C. K.-S. Leung, R. K. MacKinnon, and F. Jiang, "Distributed uncertain data mining for frequent patterns satisfying anti-monotonic constraints," in Proc. IEEE AINA Workshops 2014, pp. 1–6.
21. C. K.-S. Leung, M. A. F. Mateo, and D. A. Brajczuk, "A tree-based approach for frequent pattern mining from uncertain data," in Proc. PAKDD 2008, pp. 653– 661.