Mining Frequent Itemsets over Uncertain Databases: An Experimental Study

YONGXIN TONG, Hong Kong University of Science and Technology
LEI CHEN, Hong Kong University of Science and Technology
YURONG CHENG, Northeastern University, China
PHILIP S. YU, University of Illinois at Chicago

In recent years, due to the wide applications of uncertain data, i.e. RFID data, sensor data, real-time video data, etc., mining frequent itemsets over uncertain databases has attracted much attention. In uncertain databases, the *support* of an itemset is a discrete random variable instead of a fixed occurrence count of this itemset. Thus, unlike the corresponding problem in deterministic databases where the frequent itemset has a unique definition, the frequent itemset under uncertain databases has two different definitions so far. The first definition, referred as the expected support-based frequent itemset, employs the expectation of the *support* of an itemset to measure whether this itemset is frequent. The second definition, referred as the probabilistic frequent itemset, determines whether an itemset is frequent or not based on the probability that the *support* of this itemset is greater than the user-specific minimum support threshold. Therefore, existing works on mining frequent itemsets over uncertain databases are divided into two distinct groups, and no study is conducted to comprehensively compare the two different definitions. In addition, since no uniform experimental platform exists, current solutions for the same definition even generate inconsistent results. In this paper, we first identify the relationship between the two different definitions. Through extensive experiments, we verify that the discovered connection between the two definitions and demonstrate that the two definitions can be unified together when the size of data is large enough. Secondly, we provide baseline implementations of eight existing representative algorithms and test their performances with the uniform measures. Finally, according to the test results over many different benchmark data sets, we clarify several existing inconsistent conclusions and discuss some new findings.

Categories and Subject Descriptors: H.2.8 [**Information Systems**]: Database Applications

General Terms: Algorithms, Experimentation, Performance, Theory

Additional Key Words and Phrases: Uncertain database, frequent itemset mining

ACM Reference Format:
Yongxin Tong, Lei Chen, Yurong Cheng, Philip S. Yu, 2013. Mining Frequent Itemsets over Uncertain Databases: An Experimental Study. *ACM Trans. Knowl. Discov. Data.* X, X, Article X (September 20XX), 31 pages.
DOI:http://dx.doi.org/10.1145/0000000.0000000

1. INTRODUCTION

With the wide usage of Internet of Things (IoT) and pervasive computing in recent years, a large amount of uncertain data has been collected, such as such as radio fre-

Author's addresses: Y. Tong and L. Chen, Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong, China, email: {yxtong, leichen}@cse.ust.hk; Y. Cheng, College of Information Science and Engineering, Shenyang, China, email: cyrneu@gmail.com; P.S. Yu, Department of Computer Science, University of Illinois at Chicago, Chicago, IL, USA, email: psyu@cs.uic.edu.

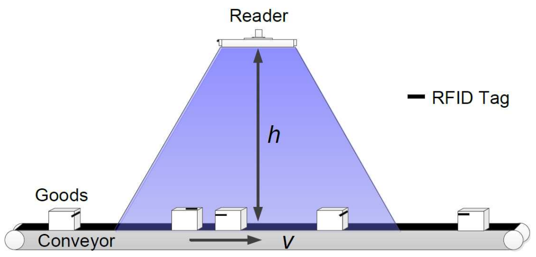


Fig. 1. A Real-Time Airport Luggage Tracking System.

quency identification (RFID)/ sensor network monitoring data [Liu et al. 2010; Mo et al. 2009], Global-Positioning System (GPS) data [Cheng et al. 2004], real-time moving object data [Chen and Ng 2004; Chen et al. 2005] and protein-protein interaction (PPI) network data [Suthram et al. 2006; Yuan et al. 2011; Yuan et al. 2012], uncertain data mining has become a hot topic in the data mining and database communities [Aggarwal 2009; Aggarwal et al. 2009; Aggarwal and Yu 2009; Jiang and Pei 2011; Jiang et al. 2013; Kao et al. 2010]. Since the problem of mining frequent itemsets is one of the most fundamental research topics in data mining, mining frequent itemsets over uncertain databases has also attracted much attention [Aggarwal et al. 2009; Bernecker et al. 2009; Calders et al. 2010a; 2010b; Chui and Kao 2008; Chui et al. 2007; Gao and Wang 2010; Leung et al. 2008; Sun et al. 2010; Tong et al. 2012a; Tong et al. 2012b; Tong et al. 2012c; Wang et al. 2010; Zhang et al. 2008]. In the following, we show two real application examples for mining frequent items/itemsets over uncertain data.

Example 1.1. (*A Real-Time Airport Luggage Tracking System*) Fig. 1 shows an RFID-based airport luggage monitoring system, which employs some RFID readers to monitor the situation of luggage in the airport in real time. Each RFID reader has a broadcast range which is shown as the shadow part Fig. 1. Each luggage is attached a tag and is put on a conveyor belt. A tag can respond to the broadcasting signals from readers when it enters the broadcast ranges of corresponding readers.

To optimize the schedule of forklifts, which are used to carry luggage from conveyor belts to flights, airport managers need to know how many pieces of luggage of each flight are on the current conveyor belts in real time. If the number of luggage in a flight exceeds a specific threshold, the forklift will be allocated. Unfortunately, due to the limitation of hardwares and protocols of RFID, i.e., ALOHA [Schoute 1983], the accuracy of response of a tag depends on the distance between the reader and the tag and the signal strength. Therefore, the collected data from RFID readers is usually incomplete and inaccurate. In this example, if the pieces of luggage, which belong to the same flight, are considered as the same item, the above problem of scheduling forklifts is actually equal to the problem of finding frequent items in an uncertain database.

Example 1.2. (*A Sensors-based Monitoring System*) The other example is about a sensors-based monitoring system. In this system, there are many sensors, which continuously capture and collect readings of different measurements (items) (i.e. density of carbon dioxide, temperature, etc.). Due to the transmission errors or other imprecise measurement, the collected data is usually inconsistent. Thus, the uncertain data models are more suitable for represent this type of data [Zhang et al. 2008]. In order to discover potential combinational relationships in the real-time monitoring data, that is to find which monitored items frequently appear together, namely finding frequent itemsets over an uncertain database.

In the aforementioned examples, the inherent uncertainty of data is ignored if we simply apply the traditional methods of frequent itemset mining in deterministic data. Thus, it is necessary to design specialized algorithms for mining frequent itemsets over uncertain databases.

Before designing the algorithms of mining frequent itemsets in uncertain databases, the definition of frequent itemsets is the most essential issue. In deterministic data, it is clear that an itemset is frequent if and only if the support (frequency) of this itemset is not smaller than a specified minimum support, *min_sup* [Agrawal et al. 1993; Agrawal and Srikant 1994; Han et al. 2000; Zaki 2000]. However, different from the deterministic case, the definition of a frequent itemset over uncertain data has two different semantics: *expected support-based frequent itemset* [Aggarwal et al. 2009; Chui et al. 2007] and *probabilistic frequent itemset* [Bernecker et al. 2009]. Both of which consider *support* of an itemset as a discrete random variable. However, the two definitions differ on using the random variable to define frequent itemsets. 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. In this definition [Aggarwal et al. 2009; Chui and Kao 2008; Chui et al. 2007; Leung et al. 2008], *an itemset is frequent if and only if its expected support is no less than a specified minimum expected support threshold, min_esup*. In the definition of probabilistic frequent itemset [Bernecker et al. 2009; Sun et al. 2010; Wang et al. 2010], 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, and *an itemset is frequent if and only if the frequent probability of the itemset is greater than a given probabilistic threshold*.

The definition of expected support-based frequent itemset uses the expectation to measure the uncertainty, which is a simple extension of the frequent itemset in deterministic data. The definition of probabilistic frequent itemset includes the complete probability distribution of the support of an itemset. Although the expectation is known as an important statistic, it cannot show the complete probability distribution. Most prior researches believe that the two definitions should be studied respectively [Bernecker et al. 2009; Sun et al. 2010; Wang et al. 2010].

However, we find that the two definitions have a rather close connection. Both definitions consider the support of an itemset as a random variable following the Poisson Binomial distribution, which is the discrete probability distribution of a sum of independent Bernoulli trials that are not necessarily identically distributed. The existing statistical theory shows that the Poisson distribution and Normal distribution can approximate the Poisson Binomial distribution with high confidence [Wang et al. 2010; Calders et al. 2010a]. Based on the *Lyapunov Central Limit Theory* [Chung 2000], the Normal distribution can approximate the Poisson Binomial distribution with high probability. Moreover, the Poisson Binomial distribution has a sound property: the computation of the expectation and variance are the same in terms of computational complexity. Therefore, the frequent probability of an itemset can be directly computed, as long as we know the expected value and variance of the support of this itemset when the number of transactions in the uncertain database is large enough [Calders et al. 2010a]. In other words, the second definition is identical to the first definition if the first definition also considers the variance of *support* at the same time. In addition, the other interesting result is that existing algorithms for mining expected support-based frequent itemsets are applicable to the problem of mining probabilistic frequent itemsets as long as they also calculate the variance of *support* of each itemset. Thus, the efficiency of mining probabilistic frequent itemsets can be greatly improved. In this paper, we verify the conclusion via extensive experimental comparisons.

Besides the overlooking of the hidden relationship between the two above definitions, existing researches on the same definition also show contradictory conclusions. For example, in the researches of mining expected support-based frequent itemsets, [Leung et al. 2008] shows that *UFP-growth* algorithm always outperforms *UApriori* algorithm with respect to the running time. However, [Aggarwal et al. 2009] reports that *UFP-growth* algorithm is constantly slower than *UApriori* algorithm. These inconsistent conclusions make later researchers confused about which result is correct.

The lacking of uniform baseline implementations is one of the factors causing the inconsistent conclusions. Thus, different experimental results originate from discrepancy among many implementation skills, blurring what are the contributions of the algorithms. For instance, the implementation for *UFP-growth* algorithm uses the "float type" to store each probability while the implementation for *UH-Mine* algorithm adopts the "double type". The difference of their memory cost cannot reflect the effectiveness of the two algorithms objectively. Thus, uniform baseline implementations can eliminate interferences from implementation details and report true contributions of each algorithm.

Except uniform baseline implementations, the selection of objective and scientific measures is also one of the most important factors in the fair experimental comparison. Because uncertain data mining algorithms need to process a large amount of data, the running time, memory cost and scalability are basic measures when the correctness of algorithms is guaranteed. In addition, to trade off the accuracy for efficiency, approximate probabilistic frequent itemset mining algorithms are also proposed [Calders et al. 2010a; Wang et al. 2010]. For comparing the relationship between the two types of definitions, we use *precision* and *recall* as measures to evaluate the approximation effectiveness. Moreover, since the above inconsistent conclusions may be caused by the dependence on datasets, in this work, we choose five different datasets, two dense ones and three sparse ones with different probability distributions (e.g. *normal distribution vs. Zipf distribution or high probability vs. low probability*).

To sum up, we try to achieve the following goals:

— Clarify the relationship of the existing two definitions of frequent itemsets over uncertain databases. In fact, there is a mathematical close relationship between them. Based on this relationship, instead of spending expensive computation cost to mine probabilistic frequent itemsets, we can directly use the solutions for mining expected support-based itemset, as long as the size of data is large enough.
— Verify the contradictory conclusions in the existing research and summarize a series of fair results.
— Provide uniform baseline implementations for the most existing representative algorithms under two definitions. These implementations adopt common basic operations and offer a base for comparing with the future work in this area. In addition, we also proposed a novel approximate probabilistic frequent itemset mining algorithm, *NDUH-Mine*, which is combined with two existing techniques: *UH-Mine* algorithm and *Normal distribution-based approximation mining* algorithm.
— Propose an objective and sufficient experimental evaluation and test the performances of the existing representative algorithms over extensive benchmarks.

The rest of the paper is organized as follows. In Section 2, we give some basic definitions about mining frequent itemset over uncertain databases. Eight representative algorithms are reviewed in Section 3. Section 4 presents all the experimental comparisons and the performance evaluations. We conclude in Section 6.

Table I. Summary of Notations

Notation	**Meaning**
$\mathcal{UDB}$	an uncertain database
T_i	the i-th transaction in $\mathcal{UDB}$
min_sup	the minimum support ratio
min_esup	the minimum expected support ratio
pft	the probabilistic frequent threshold
$sup(X)$	the support count of an itemset X
$esup(X)$	the expectation of support of X
$Pr(X)$	the frequent probability of X

2. PROBLEM DEFINITION

In this section, we give several basic definitions of mining frequent itemsets over uncertain databases.

Let $I = \{i_1, \dots, i_v\}$ be a set of v distinct items. We name a non-empty subset, X, of I as an itemset. For brevity, we use $X = x_1 \dots x_w$ to denote itemset $X = \{x_1, x_2, \dots, x_w\}$. X is a *l-itemset* if it has l items. Given an uncertain transaction database $\mathcal{UDB}$, each transaction is denoted as a tuple $< tid, Y >$ where tid is the transaction identifier, and $Y = \{y_1(p_1), \dots, y_m(p_m)\}$. Y contains m units. Each unit has an item y_i and a probability, p_i, denoting the possibility of item y_i appearing in the tid tuple. The number of transactions containing X in $\mathcal{UDB}$ is a random variable, denoted as $sup(X)$. Please note that most existing studies of mining frequent itemsets in uncertain databases assume that all entries, which maybe items or transactions, in a uncertain database are independent from each other [Aggarwal et al. 2009; Bernecker et al. 2009; Calders et al. 2010a; 2010b; Chui and Kao 2008; Chui et al. 2007; Gao and Wang 2010; Leung et al. 2008; Sun et al. 2010; Tong et al. 2012c; Wang et al. 2010; Zhang et al. 2008]. In other words, the correlated relationship of uncertainty of different transactions is not considered in this work. In addition, Table I summarizes the symbols we use.

Given $\mathcal{UDB}$, the expected support-based frequent itemsets and probabilistic frequent itemsets are defined as follows.

Definition 2.1. (Expected Support [Chui et al. 2007]) Given an uncertain transaction database $\mathcal{UDB}$ which includes n transactions, and an itemset X, the expected support of X is:

$$esup(X) = \sum_{i=1}^{n} p_i(X) \tag{1}$$

Definition 2.2. (Expected-Support-based Frequent Itemset [Chui et al. 2007]) Given an uncertain database $\mathcal{UDB}$ including n transactions, and a minimum expected support ratio, min_esup, an itemset X is an expected support-based frequent itemset if and only if $esup(X) \geq n \times min\_esup$

Example 2.3. (Expected Support-based Frequent Itemset) Given an uncertain database in Table II and the minimum expected support, min_esup=0.5, there are only two expected support-based frequent itemsets: $A(2.1)$ and $C(2.6)$ where the number in each bracket is the expected support of the corresponding itemset.

Definition 2.4. (Frequent Probability [Bernecker et al. 2009; Zhang et al. 2008]) Given an uncertain transaction database $\mathcal{UDB}$ which includes n transactions, a minimum support ratio min_sup, and an itemset X, X's frequent probability, denoted as $Pr(X)$, is shown as follows,

$$Pr(X) = Pr\{sup(X) \geq n \times min\_sup\} \tag{2}$$

Table II. An Uncertain Database

TID	Transactions
T1	A (0.8) B (0.2) C (0.9) D (0.7) F(0.8)
T2	A (0.8) B (0.7) C (0.9) E (0.5)
T3	A (0.5) C (0.8) E (0.8) F (0.3)
T4	B (0.5) D (0.5) F (0.7)

Table III. The Probability Distribution of *sup(A)*

sup(A)	0	1	2	3
Probability	0.02	0.18	0.48	0.32

Definition 2.5. (Probabilistic Frequent Itemset [Bernecker et al. 2009; Zhang et al. 2008]) Given an uncertain transaction database $\mathcal{UDB}$ which includes n transactions, a minimum support ratio min_sup, and a probabilistic frequent threshold pft, an itemset X is a probabilistic frequent itemset if X's frequent probability is larger than the probabilistic frequent threshold, namely,

$$Pr(X) = Pr\{sup(X) \geq n \times min\_sup\} > pft \tag{3}$$

Example 2.6. (Probabilistic Frequent Itemset) Given an uncertain database in Table II, min_sup=0.5, and pft = 0.7, the probability distribution of the support of $\{A\}$ is shown in Table III. The frequent probability of $\{A\}$ is, $Pr(X) = Pr\{sup(A) \geq 4 \times 0.5\} = Pr\{sup(A) = 2\} + Pr\{sup(A) = 3\} = 0.48 + 0.32 = 0.8 > 0.7 = pft$. Thus, $\{A\}$ is a probabilistic frequent itemset.

3. ALGORITHMS OF UNCERTAIN FREQUENT ITEMSET MINING

We categorize the eight representative algorithms into three groups. The first group is the expected support-based frequent itemset mining algorithms. These algorithms aim to find all expected support-based frequent itemsets. For each itemset, these algorithms only consider the expected support to measure its frequency. The complexity of computing the expected support of an itemset is $\mathcal{O}(n)$, where n is the number of transactions in the given uncertain database. The second group is the exact probabilistic frequent itemset mining algorithms. These algorithms discover all probabilistic frequent itemsets and report exact frequent probability for each itemset. Instead of the simple expectation, these algorithms need to spend at least $\mathcal{O}(nlog^2n)$ computation cost for each itemset. The third group is the approximate probabilistic frequent itemset mining algorithms. Due to the sound properties of the *Poisson Binomial* distribution, this group of algorithms can obtain the approximate frequent probability for each itemset with high quality by only acquiring the first moment (*expectation*) and the second moment (*variance*). Thus, the third type of algorithms spend the $\mathcal{O}(n)$ computation cost for each itemset. To sum up, the third type of algorithms actually build a bridge between two different definitions of frequent itemsets when uncertain databases are large enough.

3.1. Expected Support-based Algorithms

In this subsection, we summarize three most representative expected support-based frequent itemset mining algorithms: *UApriori* [Chui and Kao 2008; Chui et al. 2007], *UFP-growth* [Leung et al. 2008], *UH-Mine* [Aggarwal et al. 2009]. The first algorithm is based on the generate-and-test framework employing the breath-first search strategy. The other two algorithms are based on the divide-and-conquer framework, which uses the depth-first search strategy. Although *Apriori* algorithm is slower than the other two algorithms into deterministic databases, *UApriori* which is the uncertain version

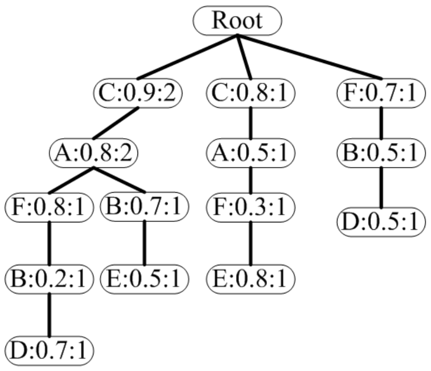


Fig. 2. UFP-Tree.

of *Apriori*, actually performs rather well among the three algorithms and is usually the fastest one in dense uncertain datasets based on our experimental results in Section 4. We further explain the three algorithms in the following subsections and Section 4.

3.1.1. UApriori. The first expected support-based frequent itemset mining algorithm was proposed by Chui et al. in 2007 [Chui et al. 2007]. This algorithm extends the well-known *Apriori* algorithm [Agrawal and Srikant 1994] to the uncertain environment and uses the generate-and-test framework to find all expected support-based frequent itemsets. We generally introduced *UApriori* algorithm as follows. The algorithm first finds all the expected support-based frequent items. Then, it repeatedly joins all expected support-based frequent i-itemsets to produce $(i+1)$-itemset candidates and test $(i+1)$-itemset candidates to obtain expected support-based frequent $(i+1)$-itemsets. Finally, it ends when no one $(i+1)$-frequent itemset is generated.

Fortunately, the *downward closure property* [Agrawal and Srikant 1994] still works in uncertain databases. Thus, the traditional *Apriori* pruning can be used when we check whether an itemset is an expected support-based frequent itemset. In other words, all supersets of this itemset must not be expected support-based frequent itemsets. In addition, several decremental pruning methods [Chui and Kao 2008; Chui et al. 2007] were proposed for further improving the efficiency. These methods mainly aim to find the upper bound of the expected support of an itemset as early as possible. Once the upper bound is lower than the minimum expected support, the traditional *Apriori* pruning can be used. However, the decremental pruning methods depend on the structure of datasets, thus, the most important pruning in *UApriori* is still the traditional Apriori pruning.

3.1.2. UFP-Growth. *UFP-growth* algorithm [Leung et al. 2008] was extended from the *FP-growth* algorithm [Han et al. 2000] which is one of the most well-known frequent itemset mining algorithms in deterministic databases. Similar to the traditional *FP-growth* algorithm, *UFP-growth* algorithm also builds an index tree, called *UFP-tree* [Han et al. 2000], to store all information of the uncertain database. Then, based on the *UFP-tree*, the algorithm recursively builds conditional subtrees and finds expected support-based frequent itemsets. The *UFP-tree* for the $\mathcal{UDB}$ in Table II is shown in Fig. 2 when min_esup=0.25.

In the process of building the *UFP-tree*, we first find all expected support-based frequent items and order these items via their expected supports. For the uncertain database in Table II, the ordered item list is $\{C$:2.6, A:2.1, F:1.8, B:1.4, E:1.3, D:1.2$\}$ where the real number following each colon is the expected support for each item.

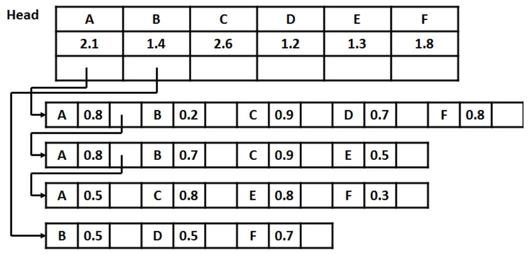


Fig. 3. UH-Struct Generated from Table II.

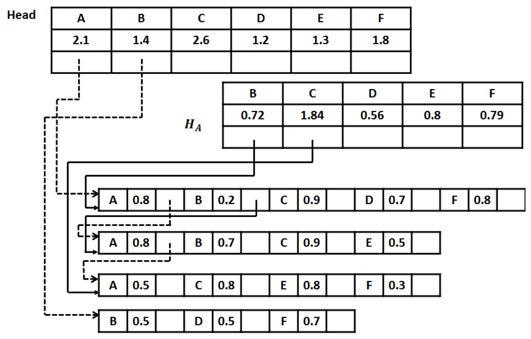


Fig. 4. UH-Struct of Head Table of A.

Based on the list, the algorithm sorts and inserts each transaction into the *UFP-tree*. Each node includes three values in *UFP-tree*. The first value is the label of the item; the second value is the appearance probability of this item; and the third value is the numbers that this node is shared from root to it.

Different from the traditional *FP-tree*, the compression quality of *UFP-tree* is substantially reduced because it is hard to take the advantage of the shared prefix path in the *FP-tree* under uncertain databases. In *UFP-tree*, items may share one node only when their labels and appearance probabilities are both same. Otherwise, items must be presented in two different types of nodes. In fact, the probabilities in an uncertain database make the corresponding deterministic database become sparse due to fewer shared nodes and paths. Thus, uncertain databases are often considered as the sparse databases. Given an $\mathcal{UDB}$, we have to build many conditional subtrees in the corresponding *UFP-tree*, which leads much redundant computation.

3.1.3. UH-Mine. *UH-Mine* [Aggarwal et al. 2009] also adopts the divide-and-conquer framework and the depth-first search strategy. The algorithm was extended from the *H-Mine* algorithm [Pei et al. 2001] which is a classical algorithm in deterministic frequent itemset mining. In particular, *H-Mine* is quite suitable for sparse databases due to its effective index structure, called *H-Struct* [Pei et al. 2001].

UH-Mine algorithm can be outlined as follows.

The first step. It scans the given uncertain database and finds all expected support-based frequent items.

The second step. The algorithm builds a global head table which contains all expected support-based frequent items. For each item, the head table stores three elements: the label of this item, the expected support of this item, and a pointer. After building the global head table, the algorithm inserts all transactions into the data structure, *UH-Struct*, [Aggarwal et al. 2009].

Table IV. Comparison of Expected Support-based Algorithms

Methods	Search Strategy	Data Structure
UApriori	Breadth-first Search	None
UFPgrowth	Depth-first Search	UFP-tree
UH-Mine	Depth-first Search	UH-Struct

The third step. In this data structure, based on a given order (i.e., alphabetical order) in the global head table, *UH-Mine* algorithm recursively builds the local head tables of corresponding projected uncertain databases. In each local head table, each item is associated with an expected support that this item appears in the corresponding projected uncertain database. Since different itemsets generate the different prefix-based projected uncertain databases, *UH-Mine* can find all expected support-based frequent itemsets until all the local expected support-based frequent items are returned in each projected uncertain database. Moreover, in order to reduce the space cost, *UH-Mine* algorithm uses the pseudo-projection method to compute the expected support of all itemsets in each projected uncertain database.

The following example further illustrates *UH-Struct*.

Example 3.1. (UH-Struct) Given an uncertain database in Table II, the *UH-Struct* and global head table of Table II are shown in Fig. 3. After building the *global UH-Strut*, the algorithm adopts the depth-first strategy to build the local head table based on the prefix $\{A\}$ in Fig. 4 where the item $\{B\}$ is associated with the expected support 0.72 in H_A since the item $\{B\}$ only appears with the item A in the first two transactions.

In the existing works on frequent itemset mining over deterministic databases, *H-Mine* algorithm fails to compress the data structure and use the dynamic frequency ordered sub-structure, such as conditional *FP-trees*, which only builds the head tables of different itemsets recursively. Therefore, for the dense databases, the *FP-growth* is superior because many items are stored in fewer shared paths. However, for the sparse databases, *H-Mine* is faster when building head tables of all levels is faster than building all conditional subtrees. Thus, *H-Mine* often performs better than *FP-growth* does in sparse databases. As discussed in Section 3.1.2, uncertain databases are quite sparse databases, so *UH-Mine* usually has the good performance.

Comparison of The Three Expected Support-based Frequent Mining Algorithms: To sum up, the search strategies and data structures of the three representative algorithms are shown in Table IV, respectively.

3.2. Exact Probabilistic Frequent Algorithms

In this subsection, we summarize two existing representative probabilistic frequent itemset mining algorithms: DP (Dynamic Programming-based Apriori algorithm) and DC (Divide-and-Conquer-based Apriori Algorithm). The exact probabilistic frequent itemset mining algorithms first calculate or estimate the frequent probability of each itemset. Then, only for itemsets whose frequent probabilities are larger than the given probability threshold are returned together with their exact frequent probabilities. Because computing the frequent probability is more complicated than calculating the expected support, a quick estimation about whether an itemset is a probabilistic frequent itemset can improve the efficiency of the algorithms. Therefore, a probability tail inequality-based pruning technique, Chernoff bound-based pruning technique, becomes a key tool to improve the efficiency of the exact mining algorithms.

3.2.1. Dynamic Programming-based Algorithms. Under the definition of the probabilistic frequent itemset, it is critical to compute the frequent probability of an itemset efficiently. [Bernecker et al. 2009] is the first work proposing the concept of frequent

ALGORITHM 1: Dynamic-Programming Algorithm (DP)

```
Input: an uncertain database $UDB$, an itemset $X$, a minimum support ratio $min\_sup$
Output: The frequent probability of $X$, $Pr(X)$
Scan $UDB$, pull out probability $p_1, \dots, p_n$ for $X$;
Initialize $LPD_X \leftarrow \{1, \dots, 0\}, PD_X \leftarrow \{0, \dots, 0\}$;
for $i \leftarrow 1$ to $|UDB|$ do
    $PD_X[0] \leftarrow (1 - p_i) \times LPD_X[0]$;
    for $k \leftarrow 1$ to $i$ do
        $PD_X[k] \leftarrow p_i \times LPD_X[k-1] + (1 - p_i) \times LPD_X[k]$
    end
end
$Pr(X) \leftarrow \sum_{m=|UDB| \times min\_sup}^{|UDB|} PD_X[m]$;
return $Pr(X)$;
```

probability of an itemset and designing a dynamic programming-based algorithm to compute the frequent probability. For the sake of the following discussion, we define that $Pr_{\geq i,j}(X)$ denotes the probability that the itemset X appears at least i times among the first j transactions in the given uncertain database. $Pr(X \subseteq T_j)$ is the probability that X appears in the j-th transaction T_j. Since different transactions in uncertain databases are independent, we can get the following recursive relationship.

$$Pr_{\geq i,j}(X) = Pr_{\geq i-1,j-1}(X) \times Pr(X \subseteq T_j) + Pr_{\geq i,j-1}(X) \times (1 - Pr(X \subseteq T_j)) \quad (4)$$

$$\text{Boundary Case:} \begin{cases} Pr_{\geq 0,j}(X) = 1 & 0 \leq j \leq n \\ Pr_{\geq i,j}(X) = 0 & i > j \end{cases}$$

where n is the number of transactions.

The frequent probability of X equals $Pr_{\geq min\_sup,n}(X)$. According to the formula (4), DP algorithm adopts the dynamic programming method to compute the frequent probability of each possible itemset and uses the Apriori framework to find all probabilistic frequent itemsets.

Because *support* of an itemset follows the Poisson Binomial distribution, from which we can deduce that the frequent probability actually equals one subtracting the probability computed from the corresponding cumulative distribution function (CDF) of the support. The details about DP algorithm is given in Algorithm 1. The algorithm pulls out the probabilities that X possibly appears in each transaction. LPD_X and PD_X are two n-dimensional vectors where n is the size of UDB in line 1. PD_X is used to store the probability distribution of $sup(X)$. LPD_X is used as a buffer to record the last updated PD_X. They are initialized in line 2. In lines 3-6, the probability distribution of $sup(X)$ is calculated according to the formula (4). In lines 7-8, DP algorithm accumulates and returns the frequent probability of X from PD_X.

Different from *UApriori*, DP algorithm computes the frequent probability instead of the expected support for each itemset. The time complexity of the dynamic programming computation for each itemset is $\mathcal{O}(n^2 \times min\_sup)$.

3.2.2. Divide-and-Conquer-based Algorithms. Apart from the dynamic programming-based algorithm, the other divide-and-conquer-based algorithm was proposed to compute the frequent probability [Sun et al. 2010]. Unlike DP algorithm, DC divides an uncertain database, UDB, into two sub-databases: UDB_1 and UDB_2. Then, in the two sub-databases, the algorithm recursively calls itself to divide the database until only one transaction left. In each partition, the algorithm calculates the probability distribution of *support* in each sub-database. Since different transactions are independent,

ALGORITHM 2: Divide-and-Conquer Algorithm (DC)

```
Input: an uncertain database $\mathcal{UDB}$, an itemset $X$
Output: The probability distribution of $sup(X)$, $PD_X$
$n \leftarrow |\mathcal{UDB}|$;
if $n \neq 1$ then
    Split $n$ transactions into two groups $\mathcal{UDB}_1$ and $\mathcal{UDB}_2$;
    $PD_X^1 \leftarrow DC(\mathcal{UDB}_1, X)$;
    $PD_X^2 \leftarrow DC(\mathcal{UDB}_2, X)$;
    $PD_X \leftarrow$ FFT-based Convolution$(PD_X^1, PD_X^2)$;
    return $PD_X$;
end
else
    $PD_X[0] \leftarrow 1 - p_1$;
    $PD_X[1] \leftarrow p_1$;
    return $PD_X$;
end
```

by computing the convolution of the two probability distributions of *support* in $\mathcal{UDB}_1$ and $\mathcal{UDB}_2$, we can obtain the probability distribution of *support* in $\mathcal{UDB}$. The computation of convolution is shown in the following formula. Finally, through the conquering part, the complete probability distribution of *support* is obtained when the algorithm terminates.

$$PD_X[k] = \sum_{i=0}^{k} PD_X^1[i] \times PD_X^2[k-i] \quad (5)$$

where PD_X^1 and PD_X^2 are used to store the probability distribution of $sup(X)$ in $\mathcal{UDB}_1$ and $\mathcal{UDB}_2$, respectively.

The pseudo codes of DC algorithm are shown in Algorithm 2. The algorithm divides the set of investors $\mathcal{UDB}$ into two groups, $\mathcal{UDB}_1$ and $\mathcal{UDB}_1$, as long as $\mathcal{UDB}$ is not empty in lines 1-2. Then, the algorithm recursively computes the probability distribution of $sup(X)$ in the two partitioned sub-databases in lines 3-5. Based on the formula (5), we compute the convolution of PD_X^1 and PD_X^2 in line 6. In order to speed up the convolution process, DC algorithm uses the *fast Fourier Transform (FFT)* calculate the convolution efficiently. Moreover, the recursive exit is in lines 8-11. Therefore, based on Algorithm 2, we obtain the probability distribution of $sup(X)$ in $\mathcal{UDB}$. After having the probability distribution of $sup(X)$, it is easy to get the frequent probability of X via the line 7 in Algorithm 1.

If DC involves the above divide-and-conquer process, its time complexity of calculating the frequent probability of an itemset is $\mathcal{O}(n^2)$ where n is the number of transaction in the uncertain database. However, in the conquering part, DC algorithm can use the Fast Fourier Transform (FFT) method to speed up the efficiency. Thus, the final time complexity of DC algorithm is $\mathcal{O}(nlog^2 n)$. In most practical cases, DC algorithm outperforms DP algorithm according to the experimental comparisons in Section 4.

3.2.3. Effect of the Chernoff Bound-based Pruning. Both the dynamic programming-based and divide-and-conquer-based methods aim to calculate the exact frequent probability for an itemset. However, the computation of the frequent probability is redundant if an itemset is infrequent. Thus, for efficiency improvement, it is a key problem to filter out unpromising probabilistic infrequent itemsets as early as possible. Because *support* of an itemset follows Poisson Binomial distribution, Chernoff bound [Chernoff 1952;

Table V. Comparison of Complexity about Determining the Frequent Probability of An Itemset

Methods	**Complexity**	**Accuracy**
DP	$\mathcal{O}(n^2 \times min\_sup)$	Exact
DC	$\mathcal{O}(nlog^2 n)$	Exact
Chernoff Bound	$\mathcal{O}(n)$	False Positive

Mitzenmacher and Upfal 2005] is a well-known tight upper bound of the frequent probability. The Chernoff bound-based pruning is shown as follows.

LEMMA 3.2. *(Chernoff Bound-based Pruning [Sun et al. 2010]) Given an uncertain database $\mathcal{UDB}$, an itemset X, a minimum support threshold min_sup, a probabilistic frequent threshold pft, the expected support of X, μ=$esup(X)$, X is a probabilistic infrequent itemset if,*

$$\begin{cases} 2^{-\delta\mu} < pft & \delta > 2e-1 \\ e^{-\frac{\delta^2\mu}{4}} < pft & 0 < \delta < 2e-1 \end{cases} \tag{6}$$

where $\delta = (n \times min\_sup - \mu - 1)/\mu$ and n is the number of transactions in UDB.

Time Complexity and Accuracy Analysis: Given an uncertain database, which contains n transactions, the time complexities and the accuracies of different methods calculating or estimating the frequent probability of an itemset are shown in Table V. We can observe that DC algorithm usually outperforms DP algorithm in terms of the efficiency. However, it is possible that DP algorithm is faster than DC algorithm if $n^2 \times min\_sup < nlog^2 n$. The Chernoff bound-based pruning spends $\mathcal{O}(n)$ to test whether an itemset is infrequent and hence it is the fastest. With respect to the accuracy, itemsets must be probabilistic frequent itemsets if they can pass the test of DP and DC. However, for Chernoff bound-based pruning, there may exist a few false-positive results because the Chernoff bound is only an upper bound of the frequent probability.

3.3. Approximate Mining Algorithms

In this subsection, we focus on three approximate probabilistic frequent algorithms. Since *support* of an itemset is the random variable following Poisson Binomial distribution, it can be approximated by the Poisson distribution and the Normal distribution effectively when uncertain databases are large enough. Moreover, for random variables following Poisson distribution and Normal distribution, we can efficiently calculate their probabilities if their expectations and variances are known. Thus, approximate probabilistic frequent algorithms have the same efficiency of expected support-based algorithms and find all probabilistic frequent itemsets with high confidence.

3.3.1. Poisson Distribution-based UApriori. In [Wang et al. 2010], the authors proposed the Poisson distribution-based approximate probabilistic frequent itemset mining algorithm, called $PDUApriori$. Since we know that *support* of an itemset follows Poisson Binomial distribution that can be approximated by the Poisson distribution [Cam. 1960], the frequent probability of an itemset is equal to the cumulative distribution function of the Poisson distribution as follows.

$$Pr(X) \sim 1 - e^{-\lambda} \sum_{i=0}^{n \times min\_sup} \frac{\lambda^i}{i!} \tag{7}$$

where $\lambda = esup(X)$, which is the expected support of X, in the above formula since the parameter λ in the Poisson distribution is the expectation. Thus, *PDUApriori* algorithm is implemented as follows. Firstly, based on the given probabilistic frequent threshold pft and the formula (7), the algorithm computes the corresponding expected support λ_{pft}. Then, the algorithm treats λ_{pft} as the minimum expected support and runs *UApriori* algorithm to find all the expected support-based frequent itemsets and treat them as the probabilistic frequent ones.

PDUApriori utilizes a sound property of the Poisson distribution, namely the parameter λ is the expectation and variance of the random variable following Poisson distribution. Because the CDF of Poisson distribution is monotonic with respect to λ, *PDUApriori* computes the corresponding λ_{pft} of the given pft and calls *UApriori* to find the results. However, this algorithm only approximately determines whether an itemset is probabilistic frequent itemset, and it cannot return the frequent probability values.

3.3.2. Normal Distribution-based UApriori. The Normal distribution-based approximate probabilistic frequent itemset mining algorithm, *NDUApriori*, was proposed in [Calders et al. 2010a]. According to the *Lyapunov Central Limit Theory*, Poisson Binomial distribution converges to Normal Distribution with high probability [Calders et al. 2010a]. Thus, the frequent probability of an itemset can be rewritten by the Standard Normal Distribution in the following formula.

$$Pr(X) \sim \Phi(\frac{n \times min\_sup - 0.5 - esup(X)}{\sigma}) \quad (8)$$

where $\Phi(.)$ is CDF of Standard Normal distribution, and $\sigma^2 = \sum_{i=1}^{n} p_i(X)(1-p_i(X))$ is the variance of the support of X. *NDUApriori* algorithm employs *UApriori* framework and uses the CDF of Standard Normal distribution to calculate the frequent probability of each itemset. Thus, different from *PDUApriori*, *NDUApriori* algorithm returns frequent probabilities for all probabilistic frequent itemsets.

However, it is impractical to apply *NDUApriori* on very large sparse uncertain databases since the *UApriori* framework may generate too many redundant candidates.

3.3.3. Normal Distribution-based UH-Mine. According to the discussion in Section 3.1, we can conclude that *UH-Mine* algorithm usually outperforms other expected support-based algorithms in sparse uncertain databases. Moreover, the Normal distribution-based approximation algorithm can provide the high quality approximate frequent probability. Due to the merits of both the *UH-Mine* algorithm and Normal Distribution approximation, we propose a novel algorithm, called *NDUH-Mine*, which integrates the framework of *UH-Mine* and the Normal Distribution approximation in order to achieve a win-win partnership in sparse uncertain databases. Compared to *UH-Mine*, we calculate the variance of each itemset when *UH-Mine* obtains the expected support of each itemset. In Section 4, we can observe that *NDUH-Mine* has a better performance than *NDUApriori* on large sparse uncertain databases.

Therefore, the Normal Distribution-based approximation algorithms build a bridge between the expected support-based frequent itemsets and the probabilistic frequent itemsets. In particular, existing efficient expected support-based mining algorithms can directly be reused and keep their intrinsic properties. In other words, under the definition of mining probabilistic frequent itemsets, *NDUApriori* is the fastest algorithm in large dense uncertain databases, while *NDUH-Mine* algorithm requires reasonable memory space and scales well to very large sparse uncertain databases.

Comparison of Algorithm Framework and Approximation Methods: Different from the exact probabilistic frequent itemset mining algorithms, the computational complexities of computing the frequent probability of each itemset for different approx-

Table VI. Comparison of Approximate Probabilistic Algorithms

Methods	**Framework**	**Approximation Methods**
PDUApriori	UApriori	Poisson Approximation
NDUApriori	UApriori	Normal Approximation
NDUH-Mine	UH-Mine	Normal Approximation

imate probabilistic frequent algorithms are the same, $\mathcal{O}(n)$ where n is the number of transactions of the given uncertain database. Thus, we mainly compare the different algorithm frameworks and approximation approaches in Table VI.

4. EXPERIMENTAL EVALUATION

4.1. Experimental Settings

In this subsection, we introduce the experimental environment, the implementations, and the evaluation methods.

Firstly, in order to conduct a fair comparison, we build a common implementation framework which provides common data structures and subroutines for implementing all the algorithms. All the experiments are performed on an Intel(R) Core(TM) i7 3.40GHz PC with 4GB main memory, running on Microsoft Windows 7. Moreover, all algorithms were implemented and compiled using Microsoft's Visual C++ 2010. The executable code and experiment data sets used in this paper can be downloaded at URL:*http://www.cse.ust.hk/ ∼yxtong/vldb.rar*.

According to the discussion in Section 3, we separate comparisons into the three categories: expected support-based algorithms, exact probabilistic frequent algorithms, approximation probabilistic frequent algorithms.

For each algorithm, we use the existing robust implementation for our comparison. For expected support algorithms, we use the version in [Chui and Kao 2008; Chui et al. 2007] to implement the *UApriori* algorithm which employed decremental pruning and hashing techniques to speed up the mining process. The implementation based on [Leung et al. 2008] is used to test *UFP-growth* algorithm. We do not use the UCFP-tree implementation [Aggarwal et al. 2009] since there is no obvious optimization between UFP-growth algorithm and UCFP-tree algorithm in terms of the running time and the memory cost. The *UH-Mine* algorithm is implemented based on the version in [Aggarwal et al. 2009]. For four exact probabilistic frequent algorithms, *DPNB* (Dynamic Programming-based Algorithm with No Bound) algorithm that does not include the Chernoff bound-based pruning technique is implemented based on the version in [Bernecker et al. 2009]. Correspondingly, *DCNB* (Divide-and-Conquer-based Algorithm No Bound) algorithm is modified based on the version in [Sun et al. 2010]. However, what is different is in our algorithm, each item has their own probability, while in [Sun et al. 2010], all items in a transaction share the same appearance probability. Moreover, *DPB* (Dynamic Programming-based Algorithm with Bound) algorithm [Bernecker et al. 2009] and *DCB* (Divide-and-Conquer-based Algorithm with Bound) algorithm [Sun et al. 2010] represent the corresponding algorithms of *DPNB* and *DCNB*, but include the Chernoff bound-based pruning, respectively. For three approximation mining algorithms, the implementation of *PDUApriori* algorithm is based on [Wang et al. 2010] and integrates all optimized pruning techniques. *NDUApriori* [Calders et al. 2010a] and *NDUH-Mine* are implemented based on the frameworks of *UApriori* and *UH-Mine*, respectively. A hashing function is used to compute the cumulative distribution function of Standard Normal distribution efficiently.

Based on the experimental comparisons of existing researches, we choose five classical deterministic benchmarks from FIMI repository [1], and assign a probability gener-

[1]http://fimi.us.ac.be

Table VII. Characteristics of Datasets

Dataset	# of Trans.	# of Items	Ave. Len.
Connect	67557	129	43
Accident	340183	468	33.8
Kosarak	990002	41270	8.1
Gazelle	59601	498	2.5
T25I15D320k	320,000	994	25

Table VIII. Default Parameters of Datasets

Dataset	Mean	Var.	min_sup	pft
Connect	0.95	0.05	0.5	0.9
Accident	0.5	0.5	0.5	0.9
Kosarak	0.5	0.5	0.0005	0.9
Gazelle	0.95	0.05	0.025	0.9
T25I15D320k	0.9	0.1	0.1	0.9

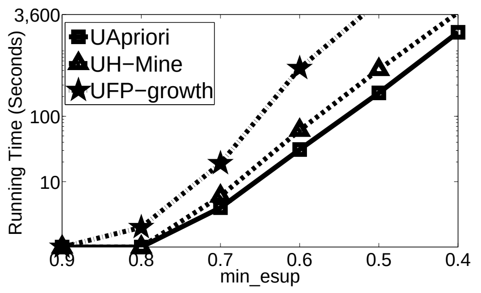
ated from Gaussian distribution to each item. Assigning probabilities to a deterministic database to generate uncertain data is widely accepted by the current community [Aggarwal et al. 2009; Bernecker et al. 2009; Calders et al. 2010a; 2010b; Chui and Kao 2008; Chui et al. 2007; Gao and Wang 2010; Leung et al. 2008; Sun et al. 2010; Tong et al. 2012a; Wang et al. 2010]. Five data sets include two dense datasets, Connect and Accident, two sparse datasets, Kosarak and Gazelle, and a very large synthetic dataset T25I15D320k which was used for testing the scalability of uncertain frequent itemset mining algorithms [Aggarwal et al. 2009]. The characteristics of above datasets are shown in Table VII, where the average length is used to measure the dense or sparse degree of a dataset. In other words, a bigger average length means that the dataset is denser. Thus, in the five datasets, Connect is the densest, and Gazelle is the sparsest. In addition, in order to verify the influence of uncertainty, we also test another probability distribution, Zipf distribution, instead of Gaussian distribution. Among the cases that datasets following the Gaussian distribution, we further categorize four scenarios. The first scenario is that a dense dataset with high mean and low variance, namely Connect with the mean (0.95) and the variance (0.05). The second scenario is that a dense dataset with low mean and high variance, namely Accident with the mean (0.5) and the variance (0.5). The third scenario is that a sparse dataset with high mean and low variance, namely Gazelle with the mean (0.95) and the variance (0.05). The fourth scenario is that a sparse dataset with low mean and high variance, namely Kosarak with the mean (0.5) and the variance (0.5). Moreover, in the case that the dataset following the Zipf distribution, only a dense dataset following the Zipf distribution varying the skew from 0.8 to 2 is tested because the sparse datasets following Zipf distribution have a very smaller number of frequent itemsets, thus we did not get any meaningful results from these datasets.

The probability parameters and their default values of each dataset are shown in Table VIII. For all the tests, we perform 10 runs per experiment and report the averages. In addition, we do not report the running time over 1 hour.

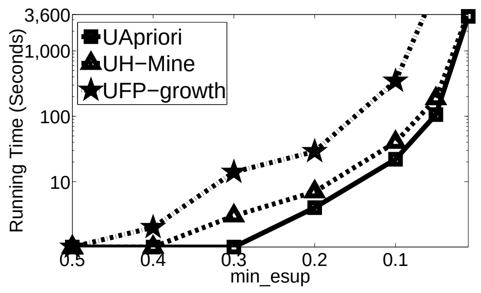
4.2. Expected Support-based Algorithms

In this section, we compare three expected support-based algorithms, *UApriori*, *UFP-growth*, and *UH-Mine*. Firstly, we report the running time and the memory cost in two dense datasets and two sparse datasets. Secondly, we present the scalability of the three algorithms. Finally, we study the influence of the skew in the Zipf distribution.

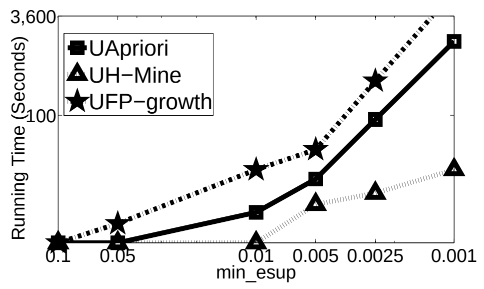
Running Time. Fig.5(a) - Fig.5(d) show the running time of expected support-based algorithms w.r.t min_esup in Connect, Accident, Kosarak, and Gazelle datasets. When min_esup decreases, we observe that the running time of all the algorithms goes up.



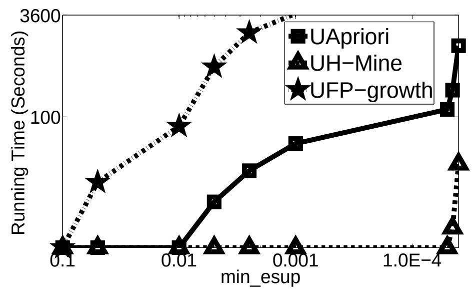
(a) Connect: min_esup vs. time



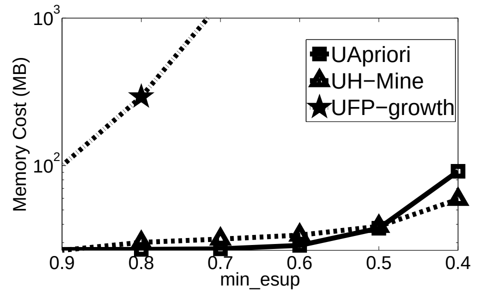
(b) Accident: min_esup vs. time



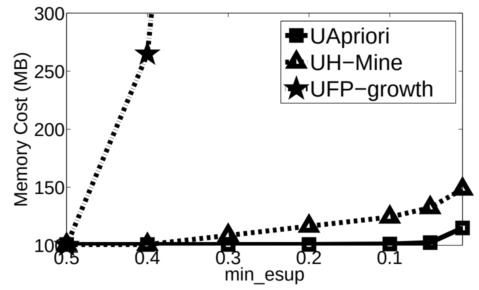
(c) Kosarak: min_esup vs. time



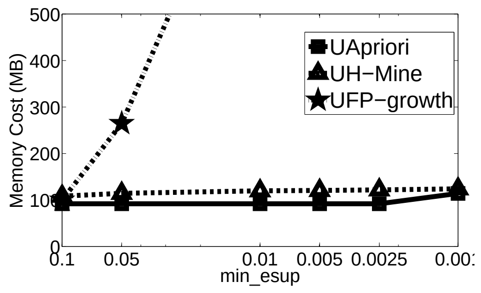
(d) Gazelle: min_esup vs. time



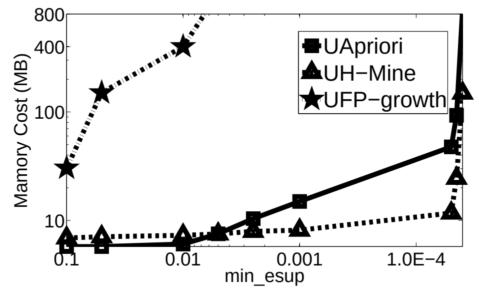
(e) Connect: min_esup vs. memory



(f) Accident: min_esup vs. memory



(g) Kosarak: min_esup vs. memory



(h) Gazelle: min_esup vs. memory

Fig. 5. Running Time and Memory Cost of Expected Support-based Frequent Algorithms

Moreover, *UFP-growth* is always the slowest in the above results. *UApriori* is faster than *UH-Mine* in Fig.5(a) and Fig.5(b), on the other hand, *UH-Mine* is faster than *UApriori* in Fig.5(c) and Fig.5(d).

It is reasonable because *UApriori* outperforms other algorithms when the uncertain dataset is dense, and min_esup is higher. These conditions cause that the search spaces of mining algorithms are relatively small. Under this case, the breadth-first-search-based algorithms are faster than the depth-first-search-based algorithms. Thus, *UApriori* outperforms the other two depth-first-search-based algorithms in Fig.5(a) and Fig.5(b). Otherwise, the depth-first-search-based algorithm, *UH-Mine*, is better, which is proven by Fig.5(c) and Fig.5(d). However, even *UFP-growth* using depth-first-search strategy, it does not perform well because *UFP-growth* spends too much time on recursively constructing many redundant conditional subtrees with limited shared paths. In addition, the other interesting observation is that slopes of curves in Fig.5(a) are larger than those in Fig.5(c). This result makes sense because the slope of the curve depends on the density of a dataset, which is measured by the average length of transaction of a dataset.

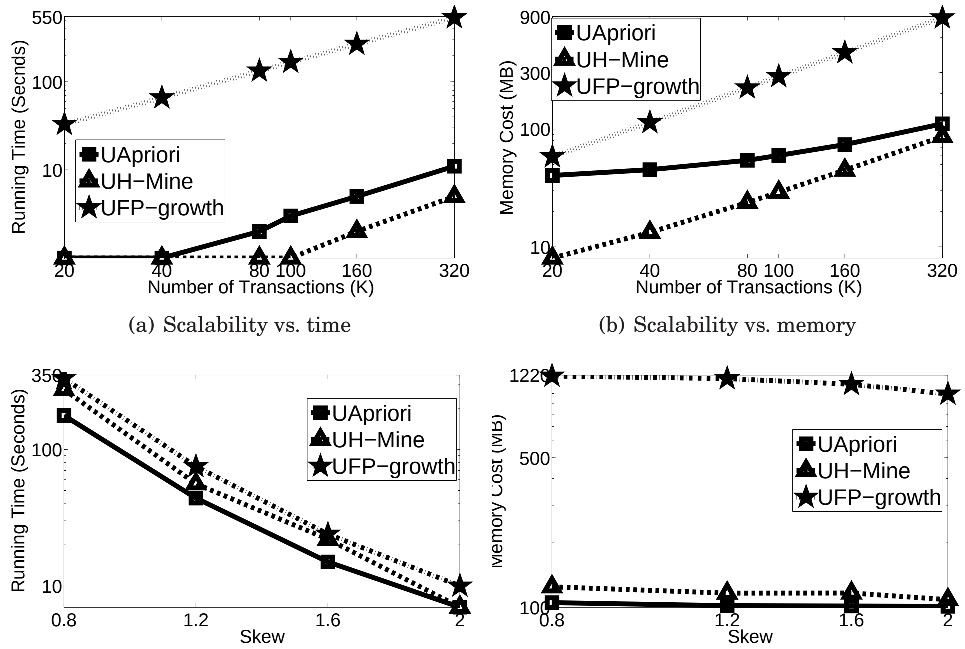
Memory Cost. According to Fig.5(e) - Fig.5(h), *UFP-growth* spends the most memory among all the three algorithms. Similar to the conclusion given in the above analysis of Running Time, *UApriori* is superior to *UH-Mine* if and only if the uncertain dataset is dense and min_esup is high enough, otherwise, *UH-Mine* is the winner.

UApriori uses less memory when min_esup is high and dataset is dense because there are a small number of frequent itemsets. However, as min_esup decreasing and dataset becoming sparse, *UApriori* has to require much more memory to store redundant infrequent candidates. Thus, the memory usage trend of *UApriori* changes sharply with decreasing min_esup. For *UH-Mine*, the main memory cost is used to initialize its *UH-Struct*. However, with the increase of min_esup, *UH-Struct* only spends the limited memory cost on building the head tables for different prefixes. Thus, the memory usage of *UH-Mine* increases smoothly. However, *UFP-growth* is the most memory consuming one among three algorithms as well.

Scalability. We further analyze the scalability of three expected support-based algorithms. In Fig.6(a), varying the number of transactions in the dataset from 20k to 320k, we observe that the running time is linear. With the increase of the size of the dataset, the time of *UApriori* is close to that of *UH-mine*. This is because all the items in T25I15D30k have similar distributions. Fig.6(b) reports the memory usages of three algorithms, which demonstrate the linearity in terms of the number of transactions. Moreover, we can find that the memory usage increase of *UApriori* is more stable than that of two other algorithms. This is because *UApriori* does not need to build a special data structure to store the uncertain database. However, the two other algorithms have to spend the extra memory cost for storing their data structures.

Effect of the Zipf distribution. For verifying the influence of uncertainty under different distributions, Fig.6(c) and Fig.6(d) show the running time and the memory cost of the three algorithms in terms of the skew parameter of Zipf distribution. We can observe that the running time and the memory cost decrease with the increase of the skew parameter. Due to the property of Zipf distribution, more items are assigned the zero probability with the increase of the skew parameter, which results in fewer frequent itemsets. Specifically, when the skew parameter increases, we can observe that *UH-Mine* outperforms *UApriori* gradually.

Conclusions. To sum up, there is no clear winner among current proposed mining algorithms. In the condition of dense datasets and higher min_esup, *UApriori* spends the least time and memory. Otherwise, *UH-Mine* is the winner.



(a) Scalability vs. time (b) Scalability vs. memory

(c) Zipf: skew vs. time (d) Zipf: skew vs. memory

Fig. 6. Scalability and Effect of the Zipf distribution of Expected Support-based Frequent Algorithms

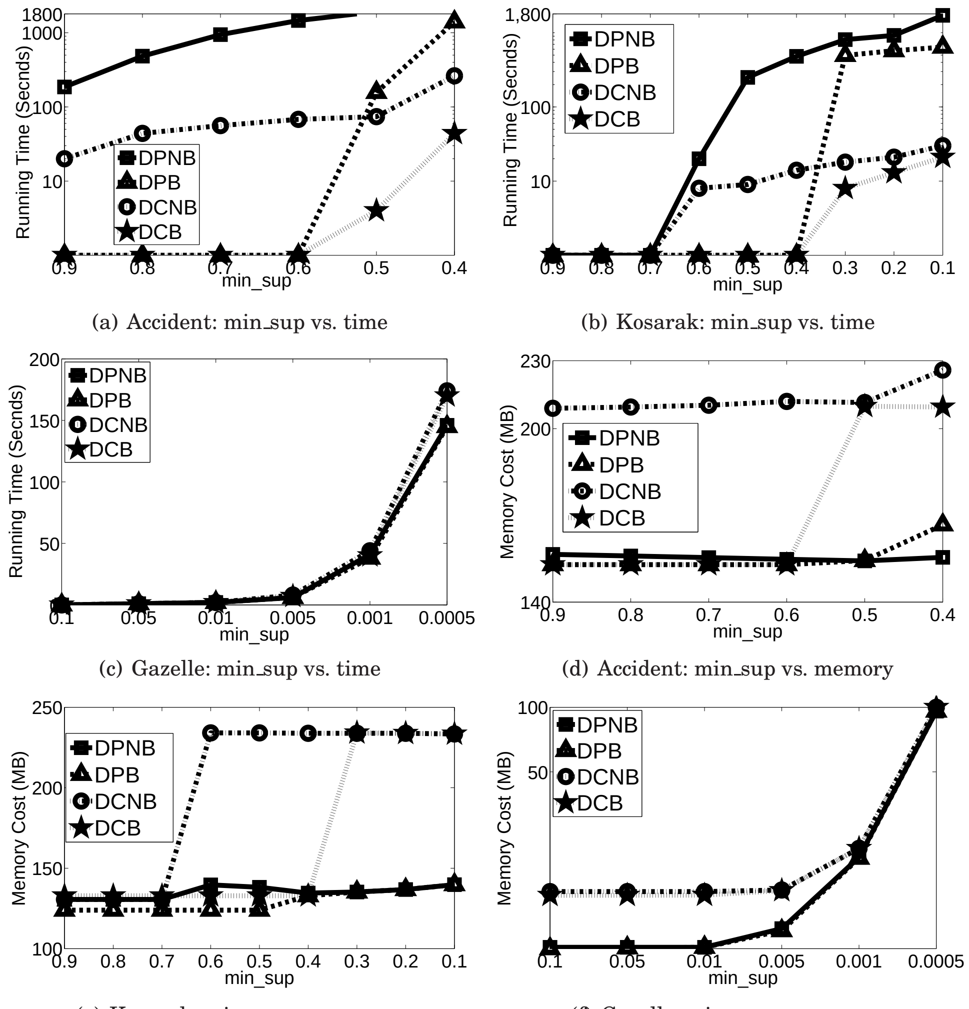
Moreover, *UFP-growth* is often the slowest algorithm and spends the largest memory cost since *UFP-growth* has only limited shared paths so that it has to spend too much time and memory on redundant recursive computation.

Finally, the influence of the Zipf distribution is similar to that of a very sparse dataset. Under the Zipf distribution, *UH-Mine* algorithm usually performs very well.

4.3. Exact Probabilistic Frequent Algorithms

In this section, we compare four exact probabilistic frequent algorithms: *DPNB*, *DCNB*, *DPB* and *DCB*. Firstly, we show the running time and the memory cost in terms of changing min_sup. We then present the influence of pft on the running time and the memory cost. Moreover, the scalability of the three algorithms is studied. Finally, we report the influence of the skew in the Zipf distribution.

Effect of min_sup. Fig.7(a) and Fig.7(b) show the running time of the four competitive algorithms w.r.t. min_sup in Accident and Kosarak datasets, respectively. With the Chernoff-bound-based pruning, we can see that *DCB* is often faster than *DPB*. However, without the Chernoff-bound-based pruning, we can find that *DCNB* is always faster than *DPNB*. This is reasonable because the time complexity of computing the frequent probability of each itemset in divide-and-conquer-based algorithms is $O(nlog^2n)$, which is more efficient than that of dynamic programming-based algorithms, $O(n^2 \times min\_sup)$. When the same type algorithms are compared, we can find that *DCB* is faster than *DCNB* and *DPB* is faster than *DPNB*. These results show that most infrequent itemsets can be filtered by the Chernoff bound-based pruning quickly. Moreover, we also observe that *DPB* is faster than *DCNB*, this is because there are only a small number of frequent itemsets when min_sup is high, most of the infrequent itemsets are already pruned by the Chernoff bound.



(a) Accident: min_sup vs. time

(b) Kosarak: min_sup vs. time

(c) Gazelle: min_sup vs. time

(d) Accident: min_sup vs. memory

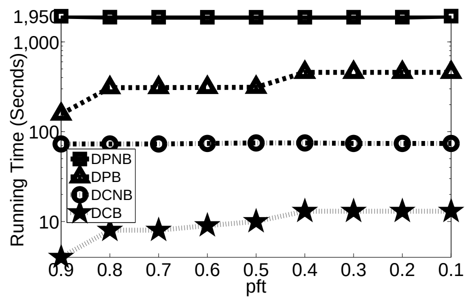
(e) Kosarak: min_sup vs. memory

(f) Gazelle: min_sup vs. memory

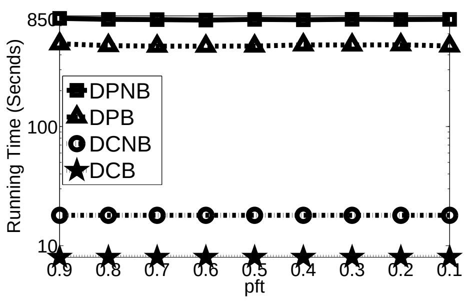
Fig. 7. Running Time and Memory of Exact Probabilistic Frequent Algorithms via Varying min_sup

Apart from the above results, we also verify the running time of the four algorithms over a quite sparse data set, *Gazelle*. Fig.7(c) reports the running time of these algorithms while varying min_sup in *Gazelle*. To our surprise, the results show that *DCB* and *DCNB* algorithms are slower than *DPB* and *DPNB*, which seems contracting to our analysis before. However, after investigating the details about the datasets, we find that this is due to the sparsity of *Gazelle* which has 59601 transactions, but the average length is only 2.5 items. Thus, when min_sup is smaller, the condition, $n^2 \times min\_sup < nlog^2 n$, holds.

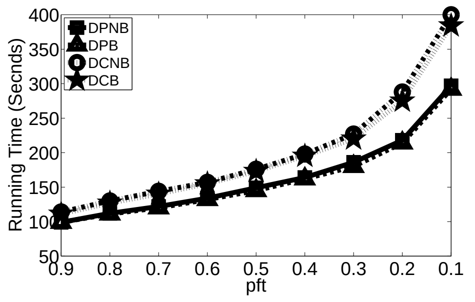
In addition, according to Fig.7(d) and Fig.7(e), this is very clear that *DPB* and *DPNB* require less memory than *DCB* and *DCNB* in Accident and Kosarak datasets. It is reasonable because both *DCB* and *DCNB* trade off the memory for the efficiency based on the divide-and-conquer strategy. In addition, we can observe that the memory usage trend of *DCNB* changes sharply with decreasing min_sup because there are a few frequent itemsets when min_sup is high, and most of the infrequent itemsets are filtered



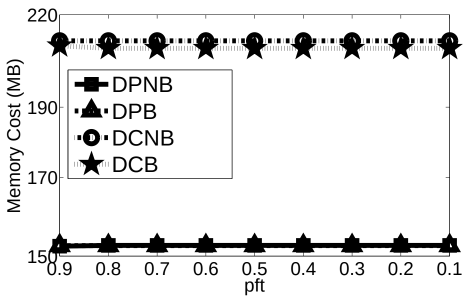
(a) Accident: pft vs. time



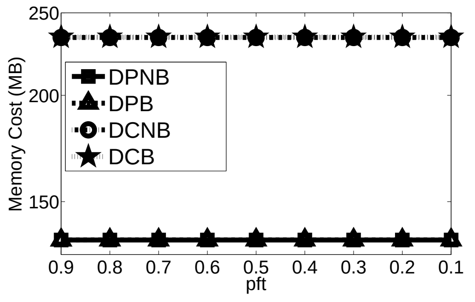
(b) Kosarak: pft vs. time



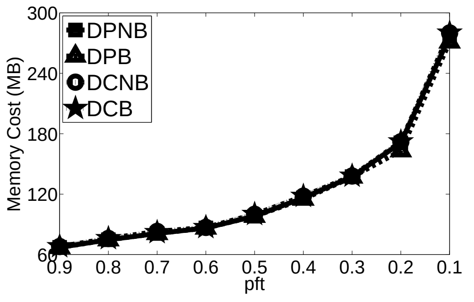
(c) Gazelle: pft vs. time



(d) Accident: pft vs. memory



(e) Kosarak: pft vs. memory

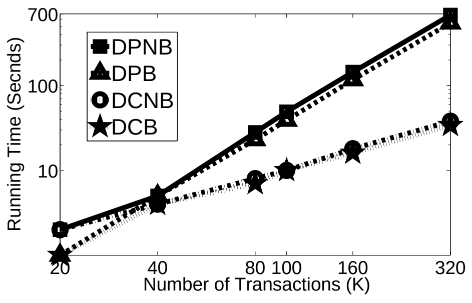


(f) Gazelle: pft vs. memory

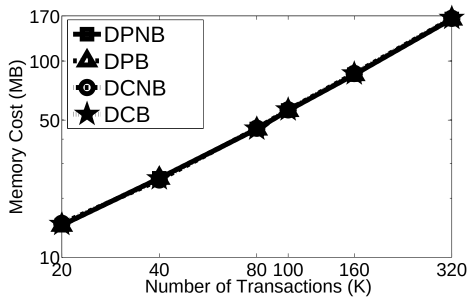
Fig. 8. Running Time and Memory of Exact Probabilistic Frequent Algorithms via Varying pft

out by the Chernoff bound-based pruning. In particular, we can find that similar observations w.r.t min_sup are shown in both the dense and the sparse datasets, which indicate that the density of the databases is not the key factor affecting the running time and the memory usage of exact probabilistic frequent algorithms. Moreover, we also confirm the memory usages of the four algorithms over Gazelle in Fig.7(f). *DCB* and *DCNB* still need more memory than *DPB* and *DPNB*. However, the effect of Chernoff bound-based pruning is not obvious since *Gazelle* is very sparse which causes less frequent itemsets.

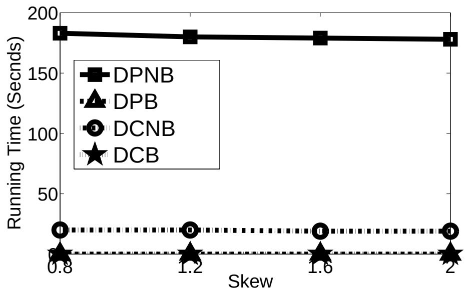
Effect of pft. Fig.8(a) and Fig.8(b) report the running time w.r.t. pft in Accident and Kosarak datasets, respectively. We can observe that *DCB* is still the fastest algorithm, and *DPNB* is the slowest one. Different from the results w.r.t. min_sup. In addition, in Fig.8(c), we also observe the similar result w.r.t. min_sup in *Gazelle*, where *DPB* and *DCPB* algorithms are faster than *DPC* and *DPNC* since the sparse dataset satisfies the condition, $n^2 \times min\_sup < nlog^2n$.



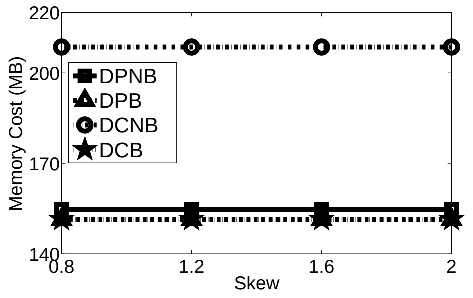
(a) Scalability vs. time



(b) Scalability vs. memory



(c) Zipf: skew vs. time



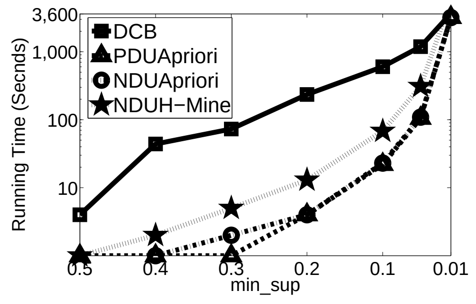
(d) Zipf: skew vs. memory

Fig. 9. Scalability and Effect of Zipf Distribution of Exact Probabilistic Frequent Algorithms

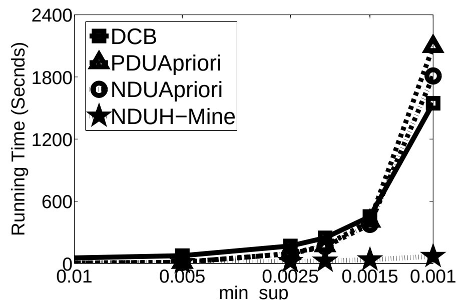
Fig. 8(d) and Fig.8(e) show the memory cost w.r.t. pft. The memory usages of both *DPB* and *DPNB* are always significantly smaller than those of both *DCB* and *DCNB*. However, in Gazelle dataset, the memory usages of the four algorithms are similar based on Fig.8(f). This is reasonable because most frequent probabilities of the frequent itemsets often equal one. Thus, as shown in Fig.8(d) and Fig.8(e), the changing trends of the running time and the memory cost are usually quite stable by varying pft. In other words, pft does not have a significant impact at the running time and the memory of the four mining algorithms. However, for a very sparse dataset, the distinction of memory usages is not evident since the probabilistic frequent itemsets are usually limited.

Scalability. Similar to the scalability analysis in Section 4.2, we still use the T25I15D320k dataset to test the scalability of four exact probabilistic frequent itemset mining algorithm. In Fig.9(a), we can find that the trends of running time of all algorithms are linear with respect to the increase of the number of transactions. In particular, the trends of both *DC* and *DCNB* are more smooth than those of *DP* and *DPNB* because the computing frequent probability for *DC* and *DCNB* are faster than those of *DP* and *DPNB*. In Fig.9(b), we observe that the memory cost of four algorithms linearly varies w.r.t. the number of transactions.

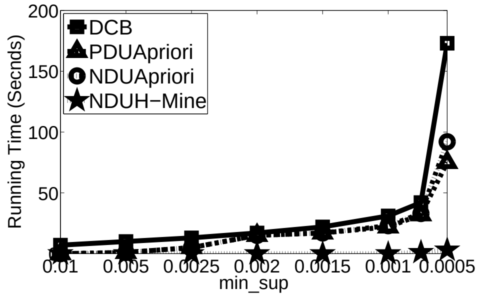
Effect of the Zipf distribution. Fig.9(c) and Fig.9(d) show the running time and the memory cost of the four algorithms in terms of the skew parameter of Zipf distribution. We can observe that the running time and the memory cost decrease with the increase of the skew parameter. By varying the skew parameter, the changing trends of the running time and the memory cost are quite stable. Therefore, the skew parameter of Zipf distribution does not have a significant impact at the running time and the memory cost.



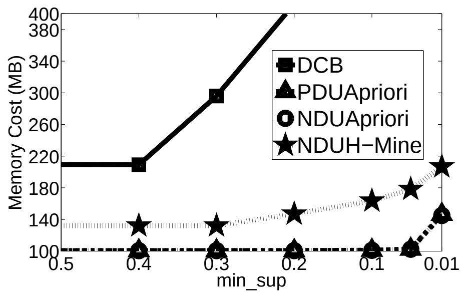
(a) Accident: min_sup vs. time



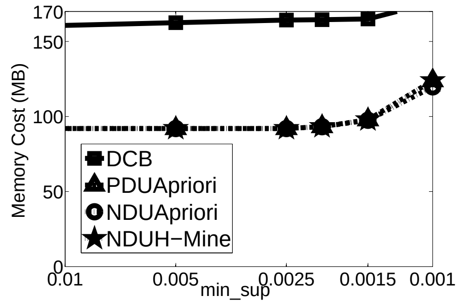
(b) Kosarak: min_sup vs. time



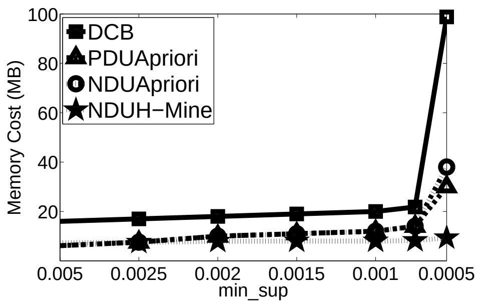
(c) Gazelle: min_sup vs. time



(d) Accident: min_sup vs. memory



(e) Kosarak: min_sup vs. memory



(f) Gazelle: min_sup vs. memory

Fig. 10. Time and Memory of Approximation Probabilistic Frequent Algorithms via Varying min_sup

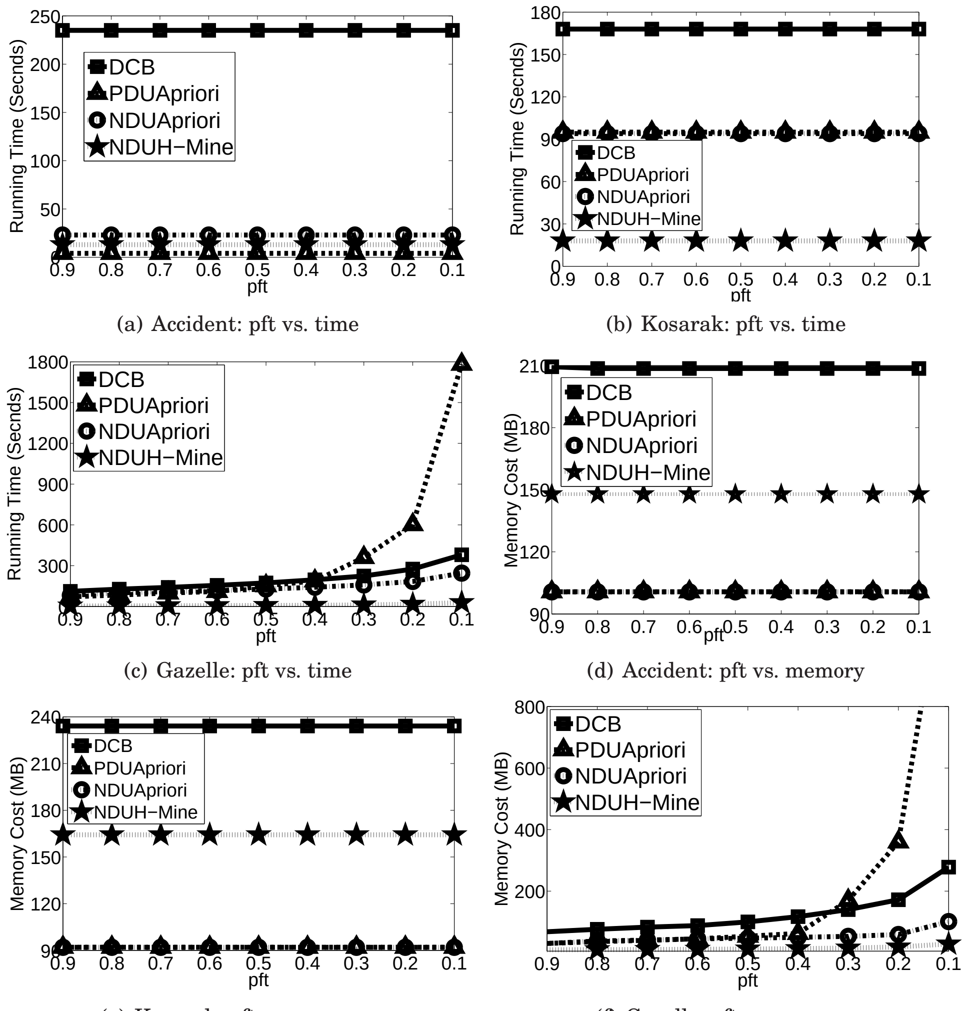
Conclusions. First of all, *DCB* algorithm is the fastest algorithm in most cases. However, compared to *DPB*, it spends the more memory.

In addition, the Chernoff bound-based pruning is an important tool to speed up exact probabilistic frequent itemset mining algorithms. Based on computational analysis, the computing Chernoff bound of each itemset is only $\mathcal{O}(n)$. However, *DC* and *DP* algorithms have to spend $\mathcal{O}(nlog^2 n)$ and $\mathcal{O}(n^2 \times min\_sup)$ to calculate the exact frequent probability for each itemset, respectively. Therefore, it is clear that the Chernoff bound-based pruning can reduce the running time if it can filter out some infrequent itemsets.

Thirdly, *DC* algorithm does not always outperform DP algorithm. In some sparse datasets, *DP* algorithm is the winner if $n^2 \times min\_sup) > nlog^2 n$.

4.4. Approximate Mining Algorithms

In this section, we mainly compare three approximation probabilistic frequent algorithms, *PDUApriori*, *NDUApriori*, and *NDUH-Mine*, and an exact probabilistic fre-



(a) Accident: pft vs. time (b) Kosarak: pft vs. time

(c) Gazelle: pft vs. time (d) Accident: pft vs. memory

(e) Kosarak: pft vs. memory (f) Gazelle: pft vs. memory

Fig. 11. Time and Memory of Approximation Probabilistic Frequent Algorithms via Varying pft

quent algorithm, *DCB*. Firstly, we report the running time and the memory cost in terms of min_sup. Then, we present the running time and the memory cost when pft is changed. In addition, we test the precision and the recall to evaluate the approximation quality. Finally, we test the scalability of the approximation algorithms.

Effect of min_sup. First of all, we test the running time of the four algorithms w.r.t. min_sup, the results are shown in Fig.10(a), Fig.10(b) and Fig.10(c). In Fig.10(a), both *PDUApriori* and *NDUApriori* are faster than the other two. In Fig.10(b) and Fig.10(c), *NDUH-Mine* is always the fastest. Moreover, *DCB* is the slowest one among the four algorithms since it offers exact answers. This is reasonable because *PDUApriori* and *NDUApriori* are based on the *UApriori* framework which performs best under the conditions that uncertain dataset is dense and min_sup is enough high. Otherwise, *NDUH-Mine* is the best in the sparse scenarios.

In addition, from Fig.10(d), Fig.10(e) and Fig.10(f), we can observe that *PDUApriori*, *NDUApriori*, and *NDUH-Mine* require less memory than that of *DCB*. This is because

Table IX. Accuracy in Accident

min_sup	PDUApriori		NDUApriori		NDUH-Mine	
	P	R	P	R	P	R
0.2	0.91	1	0.95	1	0.95	1
0.3	1	1	1	1	1	1
0.4	1	1	1	1	1	1
0.5	1	1	1	1	1	1
0.6	1	1	1	1	1	1

Table X. Accuracy in Kosarak

min_sup	PDUApriori		NDUApriori		NDUH-Mine	
	P	R	P	R	P	R
0.0025	0.91	1	0.95	1	0.95	1
0.005	0.94	1	0.96	1	0.96	1
0.01	0.97	1	0.98	1	0.98	1
0.05	0.98	1	1	1	1	1
0.1	1	1	1	1	1	1

Table XI. Accuracy in Gazelle

min_sup	PDUApriori		NDUApriori		NDUH-Mine	
	P	R	P	R	P	R
0.0005	0.76	1	0.92	1	0.92	1
0.001	0.8	1	0.94	1	0.94	1
0.002	0.83	1	0.95	1	0.95	1
0.005	0.88	1	0.99	1	0.99	1
0.01	0.91	1	1	1	1	1

DCB aims to obtain the exact results. In Fig.10(d), both *PDUApriori* and *NDUApriori* require less memory because the dataset is dense, while in Fig.10(e) and Fig.10(f), *NDUH-Mine* requires less memory since the two data sets are sparse.

Effect of pft. Fig.11(a) reports the running time in terms of varying pft. We observe that both of *PDUApriori* and *NDUApriori* are still the fastest ones in Accident dataset. However, in Fig.11(b), *NDUH-Mine* is the fastest algorithm. More interestingly, in Fig.11(c), *PDUApriori* spends much more time than the other three algorithms when pft is lower. Although Kosarak and Gazelle are sparse datasets, the results seem to be inconsistent. It is reasonable because Gazelle is so sparse that makes *PDUApriori* generating bigger approximation errors. The results also prove that the approximation quality of Poisson distribution-based approximation is worse than that of Normal distribution-based approximation. The above results confirm that the density of databases is the most important factor w.r.t. the efficiency of the approximate algorithms.

In addition, Fig.11(d) shows that the memory cost of all four algorithms is steady. Similar results are also observed in Fig.11(e). Hence, varying pft usually does not affect the memory cost of algorithms. However, in Fig.11(f), we find that the memory usage of *PDUApriori* is very high when pft is lower. As we discussed above, *PDUApriori* generates has a bigger approximation error.

Precision and Recall. Since the approximation accuracy is an important measure for approximation mining algorithms, we use the precision and the recall measures, which can quantify the number of false positives and false negatives, respectively. The precision is the ratio about the number of intersection of itemsets generated from the exact and approximate mining algorithms and the number of itemsets generated from the approximate mining algorithms, that is formalized $\frac{|AR \cap ER|}{|AR|}$. Furthermore, the recall is the ratio about the the number of intersection of itemsets generated from the exact and approximate mining algorithms and the number of itemsets generated from the exact mining algorithms, that is denoted $\frac{|AR \cap ER|}{|ER|}$. Please note that AR means the result generated from the approximation mining algorithm, and ER is the result generated from the exact mining algorithm. Moreover, we only test the precision and the recall by varying min_sup because the influence of pft is far less than that of min_sup. Tables IX, X and XI are shown the precisions and the recalls of the three approximation mining algorithms in Accident, Kosarak and Gazelle, respectively. We can find that the precision and the recall are almost 1 in Accident dataset which means there is no false positive and false negative. In Kosarak, we also observe that there are a few false positives with decreasing of min_sup. In Gazelle, we can observe that the false positive of *PDUApriori* is much bigger than that of the other two algorithms. The aforementioned observations are caused by the following reasons.

On the one hand, the Normal distribution-based approximation algorithms can get better approximation effect than the Poisson distribution-based approximation algorithm. This is because the expectation and variance in Poisson distribution is the same, but the expected support and the variance of an itemset are usually unequal.

On the other hand, we especially observe that the worst approximation quality (the bigger false positive) of Poisson distribution-based approximation algorithm is in Gazelle (Table XI). This is because the approximation accuracy of the Poisson distribution-based approximation algorithm depends on the Le Cam's theorem [Cam. 1960], which provides a loose approximation error bound. According to the Le Cam's theorem [Cam. 1960], we know that the error bound between the exact frequent probability and the Poisson distribution approximation probability of an itemset, X, is,

$$\sum_{i=0}^{\infty} |Pr(sup(X) = i) - \frac{\lambda^i e^{-\lambda}}{i!}| < 2 \sum_{i=1}^{n} p_i(X)^2 \tag{9}$$

where $p_i(X)$ is the probability that this itemset appears in i-th transaction, and the parameter λ is the expected support of X. Thus, the error can be as high as $\frac{n}{2}$. Since we assign the highest probabilities (the mean (0.95) and the variance (0.05)) to Gazelle, the error bound becomes looser. This explains why the Poisson distribution-based approximation algorithm has the worst performance in Gazelle dataset.

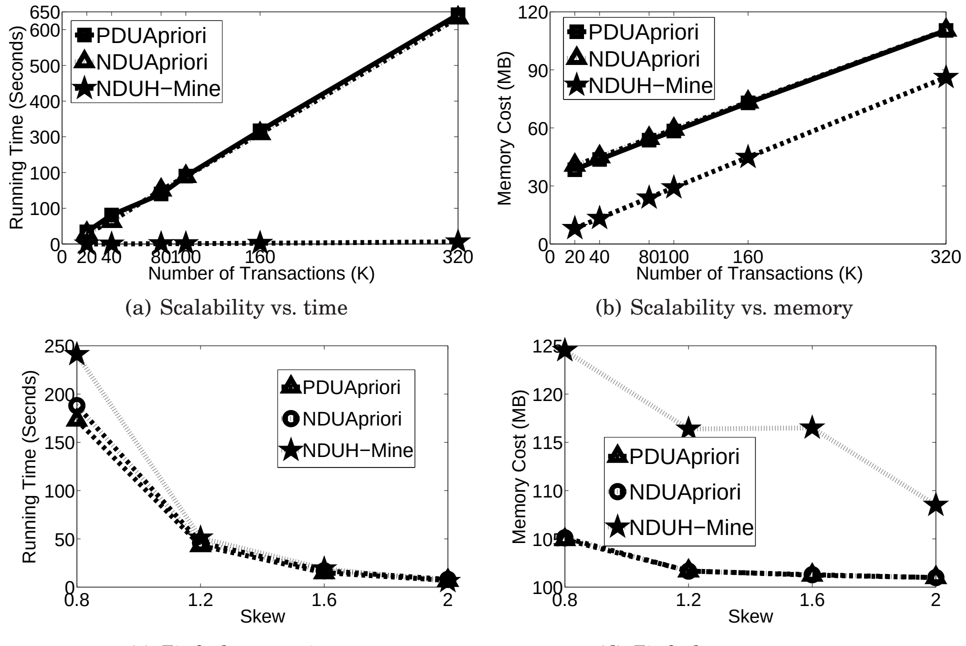
Scalability. We further analyze the scalability of the three approximate probabilistic frequent mining algorithms. In Fig.12(a), varying the number of transactions in the dataset from 20k to 320k, we find that the running time is linear. Fig.12(b) reports the memory cost of three algorithms, which also shows the linearity in terms of the number of transactions. Therefore, *NDUH-Mine* performs best.

Effect of the Zipf distribution. Fig.12(c) and Fig.12(d) show the running time and the memory cost of three approximate algorithms in terms of the skew parameter of Zipf distribution. We can observe that the running time and the memory cost decrease with the increase of the skew parameter. In particular, when the skew parameter increases, *PDUApriori* and *NDUApriori* outperforms *NDUH-Mine*.

Conclusions. First of all, approximation probabilistic frequent itemset mining algorithms can get high-quality approximation when the uncertain database is large enough. These approximation algorithms almost have no false positive or false negative. In particular, the approximation quality of the Poisson distribution-based algorithm may be worse than that of Normal distribution-based algorithm when datasets are quite sparse.

In addition, in terms of the efficiency, approximation probabilistic frequent itemset mining algorithms are much better the existing exact mining algorithms.

Finally, similar to the case of expected support-based frequent algorithms, *NDUApriori* is always the fastest algorithm in dense uncertain databases, while *NDUH-Mine* usually the best algorithm in sparse uncertain databases.



(a) Scalability vs. time (b) Scalability vs. memory

(c) Zipf: skew vs. time (d) Zipf: skew vs. memory

Fig. 12. Scalability and Effect of Zipf Distribution of Approximation Probabilistic Frequent Algorithms

4.5. Summary of New Findings

We summarize experimental results under different cases in Table XII where '$\surd$' means the winner in that case. Moreover, 'time(D)' and 'time(S)' mean the time costs in the dense data and the sparse data, respectively. The meanings of 'memory(D)' and 'memory(S)' are similar.

— Under the definition of expected support-based frequent itemset, *UApriori* is usually the fastest algorithm with lower memory cost when the database is dense and min_sup is high. On the contrary, when the database is sparse or min_sup is low, *UH-Mine* often outperforms other algorithms. However, *UFP-growth* is almost the slowest one with high memory cost.

— From Table XII, among exact probabilistic frequent itemsets mining algorithms, *DC* algorithm is the fastest algorithm in most cases. However, it trades off the memory cost for the efficiency because it has to store recursive results for the processing of the divide-and-conquer. In addition, when the condition is satisfied, *DP* algorithm is faster than *DC* algorithm.

— Again from Table XII, both *PDUApriori* and *NDUApriori* is the winner in the running time and the memory cost when the database is dense and min_sup is high, otherwise, *NDUH-Mine* is the winner. The main difference between *PDUApriori* and *NDUApriori* is that *NDUApriori* often has better approximation.

Other than the result described in Table XII, we also find:

— Approximation probabilistic frequent itemset mining algorithms usually get a high-quality approximation effect in most cases. To our surprise, the frequent probabilities of most probabilistic frequent itemsets are often 1 when the uncertain databases are large enough such as the number of transaction is more than 10,000. It is a reason-

Table XII. Summary of Eight Representative Frequent Itemset Algorithms over Uncertain Databases

	Expected Support-based Alg			Exact Prob. Freq. Alg		Approx. Prob. Freq. Alg		
	UApriori	UH-Mine	UFP-growth	DP	DC	PDUApriori	NDUApriori	NDUH-Mine
Time(D)	$\surd$(min_esup high)	$\surd$(min_esup low)			$\surd$	$\surd$(min_sup high)	$\surd$(min_sup high)	$\surd$(min_sup low)
Time(S)		$\surd$			$\surd$			$\surd$
Memory(D)	$\surd$(min_esup high)	$\surd$(min_esup low)		$\surd$		$\surd$(min_sup high)	$\surd$(min_sup high)	$\surd$(min_sup low)
Memory(S)		$\surd$		$\surd$				$\surd$
Accuracy	Exact	Exact	Exact	Exact	Exact	Approx.	Approx.(Better)	Approx.(Better)

able result. On the one hand, Lyapunov Central Limit Theory guarantees the high-quality approximation. On the other hand, according to the Poisson distribution-based approximation, we know that the frequent probability of an itemset can be approximated as $1 - e^{-\lambda} \sum_{i=0}^{N \times min\_sup} \frac{\lambda^i}{i!}$ where λ is the expected support of this itemset. When an uncertain database is large enough, the expected support of this itemset is usually large if it is a probabilistic frequent itemset. Thus, as a consequence, the frequent probability of this itemset equals 1.

— Approximation probabilistic frequent itemset mining algorithms usually far outperform any existing exact probabilistic frequent itemset mining algorithms in the algorithm efficiency and the memory cost. Therefore, the result under the definition of probabilistic frequent itemset can be obtained by the existing expected support-based mining algorithms if we compute the variance of the support of itemsets as well.
— Chernoff bound is an important tool to improve the efficiency of exact probabilistic frequent algorithms because it can filter out the infrequent itemsets quickly.

In addition, we summarize advantages and disadvantages of eight representative algorithms in Fig.XII where the term 'W' means the winner in that case. Moreover, 'time(D)' means that the time in the dense data set ('time(S)' means that the time in the sparse data set). The meaning of both 'memory(D)' and 'memory(S)' are similar.

5. EXTENSIONAL DISCUSSIONS

Since mining frequent itemsets over uncertain databases is one of fundamental topics in uncertain data mining, there are many natural variances. Thus, the discussed algorithms above can be extended to solve some variances. We categorize these variances in two groups, mining frequent item or itemsets over other kinds of uncertain data and mining other types of itemsets over uncertain data, respectively.

5.1. Mining Frequent Itemsets over Other Kinds of Uncertain Data

5.1.1. Mining frequent items/itemsets over uncertain streams. Different from mining frequent items/itemsets over static uncertain databases, dynamic uncertain streams confront with more challenges. The solution based on sketch technique for mining expected support-based frequent items was proposed in [Cormode and Garofalakis 2007]. Moreover, an efficient sampling-based method was designed to find probabilistic frequent items in uncertain streams [Zhang et al. 2008]. For the problem of mining frequent itemsets over uncertain streams, the solution based on the *UFP-growth* algorithm was proposed to discover all expected support-based frequent itemsets under the uncertain sliding-window stream model [Leung and Hao 2009]. However, to our best knowledge, there is no research of mining probabilistic frequent itemsets over uncertain stream. Based on our discussion above, Normal distribution-based approximation mining algorithm can be applied on this problem.

5.1.2. Mining frequent complex structured patterns over uncertain data. Similar to extend algorithms of mining frequent itemsets to the problem of mining frequent subgraphs in deterministic databases. Recently, the problem of mining frequent subgraphs over uncertain graph databases was extended from mining frequent itemsets over uncertain databases. $MUSE$ algorithm which is extended from *UApriori* algorithm framework was proposed to find all expected support-based frequent subgraphs [Zou et al. 2010b]. However, in the method of calculating expected supports, there are essential differences between mining frequent itemsets and mining frequent subgraphs. For an itemset, the computational complexity of calculating its expected support is $\mathcal{O}(n)$ where n is the number of transactions in the uncertain database. However, for a subgraph, the problem of calculating its expected support is #P-hard due to the structural correlation among probabilities of different edges. Therefore, the running time of $MUSE$ algorithm is dominated by computing the expected support of each subgraph. In addition, the problem of mining probabilistic frequent subgraphs over uncertain graph databases is also defined and investigated based on the DP algorithm framework [Li et al. 2012; Zou et al. 2010a].

5.2. Mining Other Types of Itemsets over Uncertain Data

5.2.1. Mining frequent closed or maximal itemsets over uncertain databases. Because mining frequent itemsets over uncertain databases can produce an exponential number of frequent itemsets, it is less useful for users to understand the mining results. Two classical solutions were proposed to compress the size of the results in deterministic data, called mining frequent closed itemsets [Tong et al. 2012a] and mining frequent maximal itemsets [Sun et al. 2010]. Mining frequent closed itemsets intends to find frequent itemsets whose supports differ from the supports of all their supersets. [Tong et al. 2012a] extended this idea to the problem of mining probabilistic frequent closed itemsets and proposed the solution which is based on the framework of *DPApriori*. Similar to the definition of the frequent probability, the frequent closed probability was defined to measure whether an itemset is a frequent closed itemset or not. However, calculating the frequent closed probability of an itemset is a #P-Hard.

Besides mining frequent closed itemsets, mining frequent maximal itemsets is another representative compressed method for reducing the size of frequent itemsets. It tries to discover frequent itemsets that any supersets of whom are infrequent itemsets. [Sun et al. 2010] firstly proposed this concept into the uncertain environment. Different from the problem of mining frequent closed itemsets over uncertain databases, mining expected support-based frequent maximal itemsets and probabilistic frequent maximal itemsets can be directly extended from the existing uncertain frequent itemset mining algorithms.

5.2.2. Mining constrained frequent itemsets over uncertain databases. Similar to the problem of mining constrained frequent itemsets in deterministic databases, the problem in uncertain environment is also an important variation of mining frequent itemsets over uncertain databases. In many real applications, users' requirements are always constrained by the attribute values of data, so it is significant to integrate some aggregate constraints, such as MIN, MAX, SUM, Average, etc., into the process of mining frequent itemsets. The problem of mining expected constrained frequent itemsets over uncertain database was defined in [Leung and Brajczuk 2010]. An algorithm based on *UFP-growth* framework was designed to solve this problem. In addition, the problem of mining constrained probabilistic frequent itemsets was not proposed. However, based on *NDUApriori* or *NDUH-Mine* algorithms, the problem of mining constrained probabilistic frequent itemsets can be addressed easily.

6. CONCLUSIONS

In this paper, we conduct a comprehensive experimental study of all the frequent itemset mining algorithms over uncertain databases. Since there are two definitions of frequent itemsets over uncertain data, most existing researches are categorized into two directions. However, through our exploration, we firstly clarify that there is a close relationship between two different definitions of frequent itemsets over uncertain data. Therefore, we need not use the current solution for the second definition and replace them with the efficient existing solutions of first definition. Secondly, we provide baseline implementations of eight existing representative algorithms and test their performances under a uniform measurement fairly. Finally, based on extensive experiments over many different benchmarks, we verify several existing inconsistent conclusions and find some new rules in this area.

REFERENCES

Charu C. Aggarwal. 2009. *Managing and Mining Uncertain Data*. Springer Publishing Company, Incorporated.

Charu C. Aggarwal, Yan Li, Jianyong Wang, and Jing Wang. 2009. Frequent pattern mining with uncertain data. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '09)*. ACM Press, New York, NY, USA, 29–38.

Charu C. Aggarwal and Philip S. Yu. 2009. A Survey of Uncertain Data Algorithms and Applications. *IEEE Trans. on Knowl. and Data Eng.* 21, 5 (May 2009), 609–623.

Rakesh Agrawal, Tomasz Imieliński, and Arun Swami. 1993. Mining association rules between sets of items in large databases. *SIGMOD Rec.* 22, 2 (June 1993), 207–216.

Rakesh Agrawal and Ramakrishnan Srikant. 1994. Fast Algorithms for Mining Association Rules in Large Databases. In *Proceedings of the 20th International Conference on Very Large Data Bases (VLDB '94)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 487–499.

Thomas Bernecker, Hans-Peter Kriegel, Matthias Renz, Florian Verhein, and Andreas Zuefle. 2009. Probabilistic frequent itemset mining in uncertain databases. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '09)*. ACM Press, New York, NY, USA, 119–128.

Toon Calders, Calin Garboni, and Bart Goethals. 2010a. Approximation of Frequentness Probability of Itemsets in Uncertain Data. In *Proceedings of the 2010 IEEE International Conference on Data Mining (ICDM '10)*. IEEE Computer Society, Washington, DC, USA, 749–754.

Toon Calders, Calin Garboni, and Bart Goethals. 2010b. Efficient pattern mining of uncertain data with sampling. In *Proceedings of the 14th Pacific-Asia conference on Advances in Knowledge Discovery and Data Mining - Volume Part I (PAKDD'10)*. Springer-Verlag, Berlin, Heidelberg, 480–487.

L. L. Cam. 1960. An approximation theorem for the Poisson binomial distribution. *Pacific J. Math.* 10, 4 (1960).

Lei Chen and Raymond Ng. 2004. On the marriage of Lp-norms and edit distance. In *Proceedings of the Thirtieth international conference on Very large data bases (VLDB '04)*. VLDB Endowment, 792–803.

Lei Chen, M. Tamer Özsu, and Vincent Oria. 2005. Robust and fast similarity search for moving object trajectories. In *Proceedings of the 2005 ACM SIGMOD international conference on Management of data (SIGMOD '05)*. ACM Press, New York, NY, USA, 491–502.

Reynold Cheng, Dmitri V. Kalashnikov, and Sunil Prabhakar. 2004. Querying Imprecise Data in Moving Object Environments. *IEEE Trans. on Knowl. and Data Eng.* 16, 9 (Sept. 2004), 1112–1127.

H. Chernoff. 1952. A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. *Ann. Math. Statist.* 23, 4 (1952).

Chun-Kit Chui and Ben Kao. 2008. A decremental approach for mining frequent itemsets from uncertain data. In *Proceedings of the 12th Pacific-Asia conference on Advances in knowledge discovery and data mining (PAKDD'08)*. Springer-Verlag, Berlin, Heidelberg, 64–75.

Chun-Kit Chui, Ben Kao, and Edward Hung. 2007. Mining frequent itemsets from uncertain data. In *Proceedings of the 11th Pacific-Asia conference on Advances in knowledge discovery and data mining (PAKDD'07)*. Springer-Verlag, Berlin, Heidelberg, 47–58.

Kailai Chung. 2000. *A Course in Probability Theory, Third Edition*. Academic Press.

Graham Cormode and Minos Garofalakis. 2007. Sketching probabilistic data streams. In *Proceedings of the 2007 ACM SIGMOD international conference on Management of data (SIGMOD '07)*. ACM Press, New York, NY, USA, 281–292.

Chuancong Gao and Jianyong Wang. 2010. Direct mining of discriminative patterns for classifying uncertain data. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '10)*. ACM Press, New York, NY, USA, 861–870.

Jiawei Han, Jian Pei, and Yiwen Yin. 2000. Mining frequent patterns without candidate generation. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data (SIGMOD '00)*. ACM Press, New York, NY, USA, 1–12.

Bin Jiang and Jian Pei. 2011. Outlier detection on uncertain data: Objects, instances, and inferences. In *Proceedings of the 2011 IEEE 27th International Conference on Data Engineering (ICDE '11)*. IEEE Computer Society, Washington, DC, USA, 422–433.

Bin Jiang, Jian Pei, Yufei Tao, and Xuemin Lin. 2013. Clustering Uncertain Data Based on Probability Distribution Similarity. *IEEE Trans. on Knowl. and Data Eng.* 25, 4 (April 2013), 751–763.

Ben Kao, Sau Dan Lee, Foris K. F. Lee, David W. Cheung, and Wai-Shing Ho. 2010. Clustering Uncertain Data Using Voronoi Diagrams and R-Tree Index. *IEEE Trans. on Knowl. and Data Eng.* 22, 9 (Sept. 2010), 1219–1233.

Carson Kai-Sang Leung and Dale A. Brajczuk. 2010. Efficient algorithms for the mining of constrained frequent patterns from uncertain data. *SIGKDD Explor. Newsl.* 11, 2 (May 2010), 123–130.

Carson Kai-Sang Leung and Boyu Hao. 2009. Mining of Frequent Itemsets from Streams of Uncertain Data. In *Proceedings of the 2009 IEEE International Conference on Data Engineering (ICDE '09)*. IEEE Computer Society, Washington, DC, USA, 1663–1670.

Carson Kai-Sang Leung, Mark Anthony F. Mateo, and Dale A. Brajczuk. 2008. A tree-based approach for frequent pattern mining from uncertain data. In *Proceedings of the 12th Pacific-Asia conference on Advances in knowledge discovery and data mining (PAKDD'08)*. Springer-Verlag, Berlin, Heidelberg, 653–661.

Jianzhong Li, Zhaonian Zou, and Hong Gao. 2012. Mining frequent subgraphs over uncertain graph databases under probabilistic semantics. *The VLDB Journal* 21, 6 (Dec. 2012), 753–777.

Yunhao Liu, Kebin Liu, and Mo Li. 2010. Passive diagnosis for wireless sensor networks. *IEEE/ACM Trans. Netw.* 18, 4 (Aug. 2010), 1132–1144.

Michael Mitzenmacher and Eli Upfal. 2005. *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press, New York, NY, USA.

Lufeng Mo, Yuan He, Yunhao Liu, Jizhong Zhao, Shao-Jie Tang, Xiang-Yang Li, and Guojun Dai. 2009. Canopy closure estimates with GreenOrbs: sustainable sensing in the forest. In *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems (SenSys '09)*. ACM Press, New York, NY, USA, 99–112.

Jian Pei, Jiawei Han, Hongjun Lu, Shojiro Nishio, Shiwei Tang, and Dongqing Yang. 2001. H-Mine: Hyper-Structure Mining of Frequent Patterns in Large Databases. In *Proceedings of the 2001 IEEE International Conference on Data Mining (ICDM '01)*. IEEE Computer Society, Washington, DC, USA, 441–448.

Frits Schoute. 1983. Dynamic frame length ALOHA. *IEEE Trans. Communications* 31, 4 (1983).

Liwen Sun, Reynold Cheng, David W. Cheung, and Jiefeng Cheng. 2010. Mining uncertain data with probabilistic guarantees. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '10)*. ACM Press, New York, NY, USA, 273–282.

Silpa Suthram, Tomer Shlomi, Eytan Ruppin, Roded Sharan, and Trey Ideker. 2006. A direct comparison of protein interaction confidence assignment schemes. *BMC Bioinformatics* 7 (2006).

Yongxin Tong, Lei Chen, Yurong Cheng, and Philip S. Yu. 2012c. Mining frequent itemsets over uncertain databases. *Proc. VLDB Endow.* 5, 11 (July 2012), 1650–1661.

Yongxin Tong, Lei Chen, and Bolin Ding. 2012a. Discovering Threshold-based Frequent Closed Itemsets over Probabilistic Data. In *Proceedings of the 2012 IEEE 28th International Conference on Data Engineering (ICDE '12)*. IEEE Computer Society, Washington, DC, USA, 270–281.

Yongxin Tong, Lei Chen, and Philip S. Yu. 2012b. UFIMT: an uncertain frequent itemset mining toolbox. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '12)*. ACM, New York, NY, USA, 1508–1511.

Liang Wang, Reynold Cheng, Sau Dan Lee, and David Cheung. 2010. Accelerating probabilistic frequent itemset mining: a model-based approach. In *Proceedings of the 19th ACM international conference on Information and knowledge management (CIKM '10)*. ACM Press, New York, NY, USA, 429–438.

Ye Yuan, Guoren Wang, Lei Chen, and Haixun Wang. 2012. Efficient subgraph similarity search on large probabilistic graph databases. *Proc. VLDB Endow.* 5, 9 (May 2012), 800–811.

Ye Yuan, Guoren Wang, Haixun Wang, and Lei Chen. 2011. Efficient Subgraph Search over Large Uncertain Graphs. *Proc. VLDB Endow.* 4, 11 (2011), 876–886.

Mohammed J. Zaki. 2000. Scalable Algorithms for Association Mining. *IEEE Trans. on Knowl. and Data Eng.* 12, 3 (May 2000), 372–390.

Qin Zhang, Feifei Li, and Ke Yi. 2008. Finding frequent items in probabilistic data. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data (SIGMOD '08)*. ACM Press, New York, NY, USA, 819–832.

Zhaonian Zou, Hong Gao, and Jianzhong Li. 2010a. Discovering frequent subgraphs over uncertain graph databases under probabilistic semantics. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '10)*. ACM Press, New York, NY, USA, 633–642.

Zhaonian Zou, Jianzhong Li, Hong Gao, and Shuo Zhang. 2010b. Mining Frequent Subgraph Patterns from Uncertain Graph Data. *IEEE Trans. on Knowl. and Data Eng.* 22, 9 (Sept. 2010), 1203–1218.