ISSN 2395-1621

# Incremental Mining Algorithm for Efficient Mining of Frequent Item Sets on Large Uncertain Databases

[#1]Rahul Arun Misal, [#2]Sadashiv Prabhakar Shinde

[1]rahulmisal26@gmail.com
[2]sada.shinde83@gmail.com

[#1]B.E. Student,
[#2]Ass.Professor(Guide)

Rajiv Gandhi College of Engineering,
A'nagar

**ABSTRACT**

**The data handled in emerging applications like location based services, sensor monitoring systems, and data integration, are often inexact in nature. In recent years, due to the wide applications of uncertain data, mining frequent item sets over uncertain databases has attracted much attention. In uncertain databases, the support of an item set is a random variable instead of a fixed occurrence counting of this item set. In this paper, the important problem of extracting frequent item sets from a large uncertain database, interpreted under the Possible World Semantics (PWS) is presented. This issue is technically challenging, since an uncertain database contains an exponential number of possible worlds. By observing that the mining process can be modelled as a Poisson binomial distribution, an algorithm was developed, which can efficiently and accurately discover frequent item sets in a large uncertain database. The important issue of maintaining the mining result for a database that is evolving (e.g., by inserting a tuple) can be presented. Specifically, the proposed mining algorithm can enable Probabilistic Frequent Item set (PFI) results to be refreshed. This reduces the need of re-executing the whole mining algorithm on the new database, which is often more expensive and unnecessary. The proposed algorithm can support incremental mining and provides the accurate results on mining the uncertain database. The extensive evaluation on real data set to validate the approach is performed.**

**General Terms: Frequent item sets, uncertain dataset, approximate algorithm, incremental mining.**

**Keywords: PFI, PWS, S-PMF, CDF.**

ARTICLE INFO

**Article History**

Received: 31st October 2017

Received in revised form :

31st October 2017

Accepted: 3nd November 2017

**Published online :**

**3rd November 2017**

## I. INTRODUCTION

The data bases used in many important and novel applications are often uncertain. For example, the locations of users obtained through RFID and GPS systems are not precise due to measurement errors [22], [28]. As another example, data collected from sensors in habitat monitoring systems (e.g., temperature and humidity) are noisy [17]. Customer purchase behaviours, as captured in supermarket basket databases, contain statistical information for predicting what a customer will buy in the future [3], [6]. Integration and record linkage tools also associate confidence values to the output tuples according to the quality of matching [16]. In structured information extractors, confidence values are appended to rules for extracting patterns from unstructured data [31]. To meet the increasing application needs of handling a large amount of uncertain data, uncertain databases have been recently developed [10], [16],[27]. Performing the data mining under the possible world semantics (PWS) can be technically challenging. In fact, the mining of uncertain data has recently attracted research attention [3]. For example, in [23], efficient clustering algorithm was developed for uncertain objects; in [21] and [32], naive Bayes and decision tree classifiers designed for uncertain data were studied. Here ,the algorithm for finding frequent item sets (i.e., sets of attribute values that appear together frequently in tuples) for uncertain databases was developed. The proposed

 

algorithm can be applied to two important uncertainty models: attribute uncertainty and tuple uncertainty, where every tuple is associated with a probability to indicate whether it exists [15], [34]. The frequent item sets discovered from uncertain data are naturally probabilistic, in order to reflect the confidence placed on the mining results.

A probabilistic frequent item(PFI) is a set of attribute values that occurs frequently with a sufficiently high probability. A database induces a set of possible worlds, each giving a (different) support count for a given item set. Hence, the support of a frequent item set is described by a probability mass function (pmf). A simple way of finding PFIs is to mine frequent patterns from every possible world obtained by the possible world semantics (PWS), and then record the probabilities of the occurrences‟ of these patterns. This is impractical, due to the exponential number of possible worlds. To remedy this, some algorithm has been recently developed to successfully retrieve PFIs without instantiating all possible worlds [6], [30], [35]. These algorithm can verify whether an item set is a PFI in O(n2) time (where n is the number of tuples contained in the database). However, the experimental results reveal that they can require a long time to complete(e.g., with a 300k real data set, the dynamic programming algorithm in [6] needs 30.1 hours to find all PFIs). The support pmf of a PFI can be captured by a Poisson binomial distribution, for both attribute and tuple uncertain data. The proposed algorithm make use of this intuition to propose a method for approximating a PFI‟s pmf with a Poisson distribution, which can be efficiently and accurately estimated. This algorithm can verify a PFI in O(n2), and is thus more suitable for large databases. The algorithm can be used to mine the frequent item sets whose probabilities of being true frequent item sets are larger than some user defined threshold [6]. The algorithm only needs very less time to find all PFIs [33] when compared with the existing algorithm, which is four orders of magnitudes faster than the method used in [6].

### 1.1 Mining evolving databases.

The important problem of maintaining mining results for changing, or evolving, databases is presented here. The type of evolving data that we address here is about the appending, or insertion of a batch of tuples to the database. Tuple insertion is common in the applications that we consider. For example, a GPS system may have to handle location values due to the registration of a new user; in an online marketplace application, information about new purchase transactions may be appended to the database for further analysis. Notice that these new tuples may induce changes to the mining result. A straightforward way of refreshing the mining results is to re- evaluate the whole mining algorithm on the new database. This can be costly, however, when new tuples are appended to the database at different time instants. In fact, if the new database D is similar to its older version, D, it is likely that most of the PFIs extracted from D remain valid for D+..Based on this intuition, The developed mining algorithm uses the PFIs of D to derive the PFIs of D+, instead of finding them from scratch. In this paper, The proposed mining algorithm for the method studied in [6], discovers the PFIs. The algorithm discovers PFIs, which can be extended to handle evolving data. As the experiments show, when the change of the database is small, running the mining algorithm on D+ is much faster than finding PFIs on D+ from scratch. In an experiment on a real data set, the mining algorithm addresses a fivefold performance improvement over its non counterpart.

## II. RELATED WORK

Mining frequent itemsets is an important problem in data mining, and is also the first step of deriving association rules [4]. Hence, many efficient itemset mining algorithms (e.g., Apriori [4] and FP-growth [18]) have been proposed.

While these algorithms work well for databases with precise values, it is not clear how they can be used to mine probabilistic data. Here we develop algorithms for extracting frequent itemsets from uncertain databases. Although our algorithms are developed based on the Apriori framework, they can be considered for supporting other algorithms (e.g., FP-growth) for handling uncertain data. For uncertain databases, Aggarwal et al. [2] and Chui et al. [14] developed efficient frequent pattern mining algorithm based on the expected support counts of the patterns. However, Bernecker et al. [6], Sun et al. [30], and Yiu et al. [35] found that the use of expected support may render important patterns missing. Hence, they proposed to compute the probability that a pattern is frequent, and introduced the notion of PFI. In [6], dynamic programming based solutions were developed to retrieve PFIs from attribute uncertain databases. However, their algorithm compute probabilities, and verify that an item set is a PFI in O(n2) time. the algorithm avoid the use of dynamic programming, and are able to verify a PFI much faster (in O(n) time). Zhang et al. [35] only considered the extraction of singletons (i.e., sets of single items), the solution discovers patterns with more than one item. Recently, Sun et al. [30] developed an threshold based PFI mining algorithm. However, it does not support attribute uncertain data considered in this paper other works on the retrieval of frequent patterns from imprecise data include: [9], which studied frequent patterns on noisy data; [24], which examined association rules on fuzzy sets; and [26], which proposed the notion of a “vague association rule.” However, none of these solutions are developed on the uncertainty models studied here. For evolving databases. A few mining algorithm that work for data have been developed. For example, in [11], the Fast Update algorithm (FUP) was proposed to efficiently maintain frequent item sets, for a database to which new tuples are inserted. the mining framework is inspired by FUP. In [12], the FUP2 algorithm was developed to handle both addition and deletion of tuples. ZIGZAG [1] also examines the efficient maintenance of maximal frequent item sets for databases that are constantly changing. In [13], a data structure, called CATS Tree, was introduced to maintain frequent item sets in evolving databases. Another structure, called CanTree [25], arranges tree nodes in an order that is not affected by changes in item frequency. The data structure is used to support mining on a changing database. To the best knowledge, maintaining frequent item sets in evolving uncertain databases has not been examined before. The proposed algorithm can also support attribute and tuple uncertainty models.

 

### 2.1 ALGORITHMS OF FREQUENT ITEMSET MINING

We categorize the eight representative algorithms into three groups. The first group is the expected support-based frequent 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 O(N), where N is the number of transactions. The second group is the exact probabilistic frequent algorithms. These algorithms discover all probabilistic frequent itemsets and report exact frequent probability for each itemset. Due to complexity of computing the exact frequent probability instead of the simple expectation, these algorithms need to spend at least O(Nlog2N) computation cost for each itemset. Moreover, in order to avoid redundant processing, the Chernfi bound-based pruning is a way to reduce the running time of this group of algorithms. The third group is the approximate probabilistic frequent algorithms. Due to the sound properties of the Poisson Binomial distribution, this group of algorithms can obtain the approximate frequent probability with high quality by only acquiring the first moment (expectation) and the second moment (variance). Therefore, the third kind of algorithms have the O(N) computation cost and return the complete probability information when uncertain databases are large enough. To sum up, the third types of algorithms actually build a bridge between two different definitions of frequent itemsets over uncertain databases.

## III. PROBLEM DEFINITION

Let V be a set of items. The algorithm adopt the following variant [6]: a database D contains n tuples, or transactions. Each transaction, $tj$ is associated with a set of items taken from V. Each item v$\epsilon$V exists in $tj$ with an existential probability Pr $v\epsilon tj \in 0,1$ , which denotes the chance that v belongs to $tj$. Under the Possible World Semantics, D generates a set of possible worlds. Each world consists of a subset of attributes from each transaction, occurs with probability Pr(*wi*).

The probabilities are one, and the number of possible worlds is exponentially large. the goal is to discover frequent patterns without expanding D into possible worlds. Each transaction $tj \in D$ is associated with a set of items and an existential probability Pr $tj \in (0,1]$ which indicates that $tj$ exists in D with probability($tj$). Again, the number of possible worlds for this model is exponentially large. The problem of mining threshold− based PFIs is then described in Section 3.1. The below Table 1 summarizes the symbols used in this paper

### 3.1 Mining Probabilistic Frequent Item Sets

Let $I \subseteq V$ be a set of items, or an *itemset*. The *support* of *I*, denoted by *s*(*I*), is the number of transactions in which *I* appears in a transaction database [4]. In precise databases, *s*(*I*) is a single value. This is no longer true in uncertain databases, because in different possible worlds, *s*(*I*) can have different values. Let *S*(*wj, I*) be the support count of *I* in possible world *wj* . Then, the probability that *s*(*I*) has a value of *i*, denoted by *PrI* (*i*), is:

$$Pr^I(i) = \sum_{w_j \in W, S(w_j, I)=i} \Pr(w_j) \quad (1)$$

Hence, $PrI(i)$( i =*1,.....,n* ) form a probability mass function (pmf) of S(I), where n is the size of database D. Now, let *minsup*$\epsilon$ (0, 1] be a percentage value, which is generally used to define minimal support in a deterministic database. An item set I is said to be frequent in a database D if s $I \geq msc(D)$ where $msc\ D = minsup \times n$ is called the minimal support count of D[4].

**Table 1. Summary of Notations**

| Notation | description |
|---|---|
| D | An uncertain database of n tuples |
| V | The set of items that appear in D |
| v | An item, where v $\in V$ |
| $t_j$ | The j-th tuple in D |
| W | The set of all possible worlds |
| $w_j$ | A possible world $w_j \in W$ |
| I | An itemset, where I $\subseteq$ V |
| minsup | A real value between (0, 1] |
| msc(D) | A minimal support count in D |
| s(I) | The support count of I in D |
| minprob | A real value between (0, 1] |
| $Pr^I(i)$ | Support prob. ( prob. I has a support count of i ) |
| $Pr_{freq}(I)$ | Frequentness probability of I |
| $\mu^I$ | Expected value of $X^I$ in D |
| $d$ | Delta database with $n^-$ tuples |
| $D^+$ | New database with $n^+$ tuples |
| $F^D$ | Set of all PFIs in D |
| $F_k^D$ | Set of all K-PFIs in D |
| $C_k^+$ | Set of size k candidates for $D^+$ |
| $DB$ | A database, can be D, d or $D^+$ |
| $S^{DB}(I)$ | The support count of I in DB |
| $Pr_{freq}^{DB}(I)$ | The frequentness probability I in DB |
| $\mu^I(DB)$ | The expected value of $X^I$ in DB |

 

For uncertain databases the frequentness probability of I, denoted by $Prfreq(I)$ is the probability that an item set is frequent [6]. Notice that $Prfreq\ I$ can be expressed as

$$Pr_{freq}(I) = \sum_{i \geq msc(D)} Pr^I(i) \qquad (2)$$

Using frequentness probabilities, the proposed algorithm can determine whether an item set I is frequent. In this paper, the algorithm adopt the definition in [6]: I is a Threshold based PFI if its frequentness probability is larger than some user defined threshold [6]. Formally, given a real value $minprob\ \epsilon(0,1]$, I is a threshold− based PFI, if

$$Pr_{freq}(I) \geq minprob \qquad (3)$$

Here the minimum probability (*minprob)* is the frequentness probability threshold.

## IV. EVALUATING S− PMF

This section deals with the computing method of the probability of the item set which is presented in the dataset. The s-pmf s(I) of item set I plays an important role in determining whether I is a PFI. An interesting observation about s(I) is that it is essentially the number of successful Poisson trials[29]. To explain, let $XjI$ be a random variable, which is equal to one if I is a subset of the items associated with transaction $tj$ (i.e., I is a subset of $tj$) or zero otherwise. Given a database of size n, each I is associated with random variables $X1I,X2I,\ldots,XnI$ . all tuples are independent. Therefore, these n variables are independent, and they represent n Poisson trials. Moreover

$$X^I = \sum_{j=1}^{n} X_j^I \qquad (4)$$

follows a Poisson binomial distribution. Next, observe an important relationship between $XI$ and $PrI(i)$ (i.e., the probability that the support of I is i )

$$Pr^I(i) = Pr(X^I = i) \qquad (5)$$

This is simply because $XI$ is the number of times that I exists in the database. Hence, the s− pmf of I, i.e., $PrI\ (i)$ is the pmf of $XI$, a Poisson binomial distribution. Using (5), it can rewrite (2), which computes the frequentness probability of I, as

$$Pr_{freq}(I) = \sum_{i \geq msc(D)} Pr(X^I = i) \qquad (6)$$

$$= Pr\left(X^I \geq msc(D)\right) \qquad (7)$$

Therefore, if the cumul**a**tive distribution function (cdf) of $XI$ is known, $Prfreq(I)$I) can also be evaluated. Next, an approach to this cdf, in order to compute $Prfreq\ I$ efficiently is presented.

### 4.2 Approximating s− pmf

From (7), it is possible to express $Prfreq\ I$ as

$$Pr_{freq}(I) = 1 - Pr(X^I \leq msc(D) - 1) \qquad (8)$$

For notational convenience, let $PrjI$ be Pr( $I \subseteq tj$ ) .Then, the expected value of $XI$ in D, denoted by $\mu I$, can be computed by

$$\mu^I = \sum_{j=1}^{n} P_j^I \qquad (9)$$

Since a Poisson binomial distribution can be well approximated by a Poisson distribution [8], (8) can be written as

$$Pr_{freq}(I) \approx 1 - F(msc(D) - 1, \mu^I) \qquad (10)$$

Where F is the cdf of the Poisson distribution with mean $\mu^I$, i.e., $F(msc(D)-1,\mu^I) = 1 - \frac{\Gamma(msc(D),\mu^I)}{(msc(D)-1)!}$ expressed using the incomplete gamma function

$$\Gamma(s,x) = \int_x^{\infty} t^{s-1} e^{-t} dt$$

mpirical results may be obtained using this function. To estimate can first compute $\mu I$ by scanning D once and summing up $PjI$ "s for all tuples $tj$ in D.

Then, $F(msc\ D\ -1,\mu I)$ is evaluated, and (10) is used to approximate $Prfreq(I)$. The following theorem can be used to support the proposed algorithm which can be presented as follows

*Theorem 1*. $Prfreq\ I\ ,if$ approximated by (10), increases Monotonically with $\mu I$ .

 

*Proof*. The cdf of a Poisson distribution, $F(i,\mu)$ can be written as

$$F(i,\mu) = \frac{\Gamma(i+1,\ \mu)}{i!} = \frac{\int_{\mu}^{\infty} t^{(i+1)-1} e^{-t} dt}{i!}$$

Since minsup is fixed and independent of $\mu$, let us examine the partial derivative w.r.t. $\mu$

$$\frac{\partial F(i,\ \mu)}{\partial \mu} = \frac{\partial}{\partial \mu}\left(\frac{\int_{\mu}^{\infty} t^{(i+1)-1} e^{-t} dt}{i!}\right)$$

$$= \frac{1}{i!}\frac{\partial}{\partial \mu}\left(\int_{\mu}^{\infty} t^{i} e^{-t} dt\right)$$

$$= \frac{1}{i!}\left(-\mu^{i} e^{-\mu}\right)$$

$$= -f(i,\ \mu) \leq 0\,.$$

Thus, the cumulative distribution (cdf) of the Poisson distribution $F\ i,\mu$ is monotonically decreasing w.r.t. $\mu$, when i is fixed. Consequently, $1-F(\ i-1,\mu)$ increases monotonically with $\mu$. Theorem 1 follows immediately by substituting $i=msc(D)$ .Intuitively, Theorem 1 states that the higher value of $\mu I$, the higher is the chance that I is a PFI. Next, the fact of how this theorem avoids the costly computations of F, and improves the efficiency of finding threshold-based PFIs was presented.

## V. MINING THRESHOLD-BASED PFIS

Can we quickly determine whether an itemset $I$ is a threshold-based PFI? Answering this question is crucial, since in typical PFI mining algorithms (e.g., [6]), candidate itemsets are first generated, before they are tested on whether they are PFI's. In Section 5.1, we develop a simple method of testing whether $I$ is a threshold-based PFI, without computing its frequentness probability. We
then enhance this method in Section 5.2. We demonstrate an adaptation of these techniques in an existing PFI-mining algorithm, in Section 5.3.

Given the values of *minsup* and *minprob*, we can test whether $I$ is a threshold-based PFI, in three steps:
**Step 1.** Find a real number $\mu m$ satisfying the equation:

$$minprob = 1 - F(msc(D) - 1, \mu_m) \qquad (11)$$

The above equation can be solved efficiently by employing numerical methods.
*Step-2*. Use (9) to compute $\mu I$ Notice that the database D has to be scanned once.

*Step-3*. If $\mu I \geq \mu m$, The algorithm conclude that I is a PFI, otherwise, I must not be a PFI.
To understand this, first notice that the right side of (11) is the same as that of (10), an expression of frequentness probability. essentially, Step-1 finds out the value of μm that corresponds to the frequentness probability threshold (i.e.,.*minprob*). In Steps 2 and 3, if μI ≥ μm,, Theorem 2 allows us to deduce that the $Prfreq\ I \geq minprob$.
Hence, these steps together can test whether an item set is a PFI. In order to verify whether I is a PFI, once μm is found, The algorithm might not have to evaluate $Prfreq(I)$. Instead, it compute μI in Step 2, which can be done in O(n) time. This is a more scalable method compared with solutions in [6] and [35], which evaluate $Prfreq(I)$ in O(n2) time. Next, the process of how this method can be further improved is presented in below section .

### 5.2 Improving the PFI Testing Process

In Step 2 of the last section, D has to be scanned once to obtain μI , for every item set I. This can be costly if D is large, and if many item sets need to be tested. For example, in the Apriori algorithm [6], many candidate item sets are generated first before testing whether they are PFIs. next the process of how the PFI testing can still be carried out without scanning the whole database was explained. Let $\mu_l^I = \sum_{j=1}^{l} p_j$ where l ∈ (0, n]. Essentially, $\mu lI$ is the "partial value" of $\mu I$ which is obtained after scanning $l$ tuples. Notice that $\mu nI=\mu I$ Suppose that $\mu m$ has been obtained from (11).
**Corollary 1.** An item set I cannot be a PFI if there exists I ∈ (0, μm ] such that

$$\mu_{n-i}^{I} < \mu_m - i \qquad (12)$$

The above equations can be used to improve the speed of the PFI testing process. Specifically, after a tuple has been scanned, The algorithm checks whether the s-pmf value of the item set exceeds the threshold value; if so, it immediately conclude that I is a PFI. After scanning $n-\mu m$ or more tuples, we examine whether I is not a PFI, by using Corollary 1. These testing procedures continue until the whole database is scanned, yielding $\mu$I , Then, the algorithm execute Step 3 (Section 5.1) to test whether I is a PFI.

### 5.3 Case Study: The Apriori Algorithm

The testing techniques just mentioned are not associated with any specific threshold-based PFI mining algorithms. Moreover, these methods support both attribute- and tuple uncertainty models. Hence, they can be easily adopted by existing algorithms. We now explain how to incorporate our techniques to enhance the Apriori [6] algorithm, an important PFI mining algorithms.

 

**Algorithm 1: Apriori-based PFI Mining**

**Input:** Uncertain database $D$, *minsup*, *minprob*
**Output:** All PFI: $F = \{F_1, F_2, \ldots, F_m\}$ // $F_k$ is set of $k$-PFIs

```
1  begin
2    μm = MinExpSup(minsup, minprob, D);
3    C1.GenerateSingleItemCandidates(D);
4    k = 1; j = 0;
5    while |Ck| ≠ 0 do
6      foreach I ∈ Ck do
7        I.μ = 0;
8      while (++j) ≤ n and |Ck| ≠ 0 do
9        foreach I ∈ Ck do
10         I.μ = I.μ + Pr(I ⊆ tj);
11         if I.μ ≥ μm then
12           Fk.push(I);
13           Ck.remove(I);
14         else if j ≥ n − ⌊μm⌋ then
15           if Pruning(I, μm, j, n) == true then
16             Ck.remove(I);
17     Ck+1.GenerateCandidate(Fk);
18     k = k + 1; j = 0
19   return F;
20 end
```

The resulting procedure (Algorithm 1) uses the "bottomup" framework of the Apriori: starting from $k$ = 1, size-$k$ PFIs (called $k$-PFIs) are first generated. Then, using Theorem 1, size-($k$ + 1) candidate itemsets are derived from the $k$-PFIs, based on which the ($k$ +1)-PFIs are found. The process goes on with larger $k$, until no larger candidate itemsets can be discovered. The main difference of Algorithm 1 compared with that of Apriori [6] is that all steps that require frequentness probability computation are replaced by our PFI testing methods. In particular, Algorithm 1 first computes $\mu m$ (Line 2). Then, for each candidate itemset $I$ generated on Line 3 and Line 17, we scan $D$ and compute its $\mu Ii$
(Line 10). If Lemma 1 is satisfied, then $I$ is put to the result (Lines 11- 3). However, if Corollary 1 is satisfied, $I$ is pruned from the candidate itemsets (Lines 14-16). This process goes on until no more candidates itemsets are found.
**Complexity.** In Algorithm 1, each candidate item needs $O(n)$ time to test whether it is a PFI. This is much faster than the Apriori [6], which verifies a PFI in $O(n2)$ time. Moreover, since $D$ is scanned once for all $k$-PFI candidates $Ck$, at most a total of $n$ tuples is retrieved for each $Ck$ (instead of $|Ck| \cdot n$). The space complexity is $O(|Ck|)$ for each candidate set $Ck$, in order to maintain $\mu I$ for each candidate. Next, we examine how to maintain PFIs in a database that is constantly evolving.

## VI. INCREMENTAL MINING

Now the method of how to efficiently maintain a set of PFIs in an evolving database is presented here, where new tuples, or transactions, are constantly appended to it. We assume that every tuple has a timestamp attribute, which indicates the time that it is created. This timestamp is not used for mining; it is only used to differentiate new tuples from existing ones. Let D be the "old" database that contains n tuples, and d be a delta database of nI tuples, whose timestamps are larger than those of tuples in D. Let D+ be a "new" database, which is a concatenation of the tuples in D and d, and has a size of N+ = n + nI. Given the set of PFIs and their s− pmfs in D, the goal is to discover PFIs on D+, under the same *minsup* and *minprob* values used to mine the PFIs of D. The algorithm uses sDB(I) and PrDBfreq(I) to respectively denote the support count and the frequentness probability of item set I in some database DB, where DB is in{D, d, D+ }. The mining problem described above can be treated as a special case of *stream mining*, which refers to the maintenance of mining results for stream data. Particularly, the database d as the arrival of |d| data units from a stream source is assumed. Moreover, assume that the sliding window initially contains D, which then expands to incorporate new stream units. Mining D+ is then equivalent to updating the mining results for the arrival of |d| stream units. In Section 8.3, an adaptation of a stream algorithm in [35] for use in mining is observed. A simple way of obtaining PFIs from D+ is to simply rerun a PFI− mining algorithm on it. However, this approach is not very economical, since 1) running a PFI algorithm on a large database is not trivial; and 2) the same algorithm has to be frequently executed if a lot of update activities occur. In fact, if only a few tuples in d are appended to D, it may not be necessary to compute all PFIs on D+ from scratch. This is because the PFIs found in D+ should not be very different from those discovered in D. Based on this intuition, a mining algorithm that finds PFIs in D+, without rerunning a complete PFI algorithm was designed. This algorithm works the best when the size of d is very small compared with that of D; nevertheless, it works with any size of d. The framework of the solution, discovers PFIs in D+, based on the PFIs found in D. this solution is extended to discover PFIs in Section 7.

### 6.1 Algorithm

The algorithm maintains frequent item set results in an evolving database. The proposed algorithm extracts frequent item sets in an "Apriori" fashion: it utilizes a bottom− up approach, where (k+1)− PFIs are generated from k− PFIs. The working process of the algorithm is shown in the below diagram.

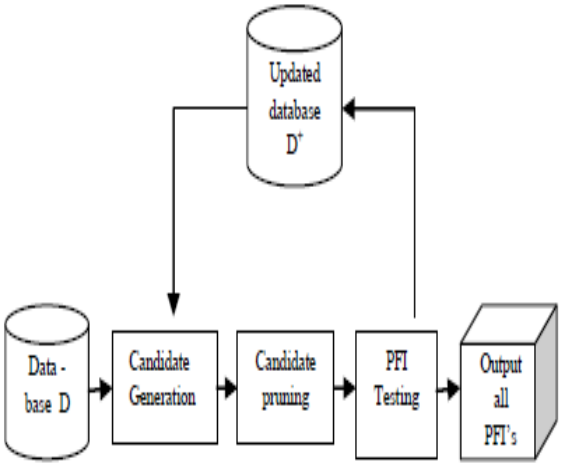


**Fig.1: PFI mining from the uncertain database**

 

The algorithm contains three phases. These are given below,

*1. Candidate generation*. In the first iteration, size−1 item sets that can be 1− PFIs are obtained, using the PFIs discovered from D, as well as the delta database d. In subsequent iterations, this phase produces size (k+1) candidate item sets, based on the k− PFIs found in the previous iteration. If no candidates are found, the algorithm halts.

*2. Candidate pruning*. With the aid of d and the PFIs found from D, this phase filters the candidate item sets that must not be a PFI.

*3. PFI testing*. For item sets that cannot be pruned, they are tested to see whether they are the true PFIs. This involves the use of database D+, as well as the s− pmf‟s of PFIs on D. Notice that in Phases 1 and 2, only d and the PFIs of D are needed. Since these pieces of information are relatively small in size (compared with D or D+ ), they are usually not very expensive to evaluate. Phase 3 involves deriving the s−pmf‟s of item sets, with the use of D+ , and is thus more expensive than other phases. If Phase 2 successfully removes a lot of candidates from consideration, the cost of executing Phase 3 can be reduced.

The above discussion is formalized in the Algorithm-1, which uses the databases D and d, as well as the set of PFIs FD collected from D (e.g., using the method of [6]). The output of the algorithm is a set F+ of PFIs for D+ , where $Fk+=\{F1+,F2+,\ldots,Fn+\}$ and Fk+ is the set of „k− PFIs‟ for D+ Let Ck+ be a set of size− k candidates found from D+. Initially k=1. The algorithm generates C1+(Phase 1). In the kth iteration, first the algorithm removes candidate item sets that cannot be k− PFIs, from Ck+ (Phase 2). if Ck+ is not empty, the algorithm performs testing on these candidates, in order to find out the true k− PFIs (i.e., Fk+,) Phase 3 generates size (k+1)−candidate item sets by using the k− PFIs. The whole process is repeated until no more candidates are found., The algorithm takes D, d, minsup, minprob as inputs and produces the output as a set of Probabilistic Frequent item sets (PFIs) in the function F, which is given as follows,

**Algorithm- 1 For PFI mining**

```
Input:  Data base D, updated delta database d, minsup,
        minprob .
Output: PFI's of D+, Fk+ = { F1+ ,F2+,....,Fn+} // Fk+ is
        the set of PFI's .
Method:
if (deltaDb is empty) then
{
      C(1).Generate( oldDb);
       k=1;
      while(C(k)!=0) do
      {
            C(k).prune(oldDb, F, μm);
           if(C(k)!=0) then
                 F =C(k).Test( oldDb, d, F, μm);
           else
                 break;
           C(k+1).Generate( oldDb ) ;
           k=k+1;
      }
}
else  //deltaDb is not empty
{
      C(1).Generate(deltaDb);
      k=1;
      while(C(k)!=0)
      {
           C(k).prune( deltaDb, F, μmI);
           if(C(k)!=0) then
                 F=C(k).Test(oldDb, d, F, μmI);
           else
                 break;
           C(k+1).Generate(d);
           k=k+1;
      }
}
return Fk+ = {F1+, F2+, .... Fn+}
```

## VII. RESULTS

Now the experimental results on the data set, called accidents, comes from the Frequent Item set Mining (FIMI) Data Set Repository. This data set is obtained from the National Institute of Statistics (NIS) for the region of Flanders (Belgium), for the period of 1991-2000. The data are obtained from the “Belgian Analysis Form for Traffic Accidents,” which are filled out by a police officer for each traffic accident occurring on a public road in Belgium. The data set contains 3,40,184 accident records, with a total of 572 attribute values. On average, each record has 45 attributes. The algorithm uses the first 10k tuples as the default data set as its input. The default value of minsup is 20 percent. To test the mining algorithm, it uses the first 10k tuples as the old database D, and the subsequent tuples as the delta database d. The default size of d is 5 percent of D. , it considers both attribute and tuple uncertainty models. For attribute uncertainty, the existential probability of each attribute is drawn from a Gaussian distribution with mean 0.5 and standard deviation 0.125. This same distribution is also used to characterize the existential probability of each tuple, for the tuple uncertainty model. The default value of minprob is 0.4. In the results presented, minsup is shown as a percentage of the data set size n. Notice that when the values of minsup or minprob are large, no PFIs can be returned; it do not show the results for these values. The experiments were carried out on the Windows XP operating system, on a machine with a 2.66 GHz Intel Core 2 Duo processor and 2 GB memory. The programs were written in Java and compiled with J2SE runtime environment 1.6.0. The proposed algorithm is implemented under the java runtime environment. The algorithm extracts the probabilistic frequent item sets within a fraction of seconds.

## VIII. DISCUSSION

Now the comparison of the performance of the PFI mining algorithms mentioned in this paper:1) APM, the Apriori algorithm used in [6] that employs the PFI testing method and 2) PA (Proposed algorithm), the proposed algorithm that uses the improved version of the PFI testing method is discussed here.

Since APM approximates s-pmf by a Poisson distribution, first examine that its accuracy with respect to AP, which yields PFIs based on exact frequentness probabilities. Here, the standard recall and precision measures [7] is used, which quantify the number of negatives and false positives. Specifically, let $FAPM$ be the set of PFIs generated by APM, and $FPA$ be the set of PFIs produced by PA. To compare the existing algorithm with the proposed algorithm, The recall and precision were used. Both recall and precision have values between 0 and 1. The recall and the precision of APM, relative to PA, are defined as

$$recall = \frac{|F_{APM} \cap F_{AP}|}{F_{APM}} \qquad (13)$$

$$precision = \frac{|F_{APM} \cap F_{AP}|}{F_{AP}} \qquad (14)$$

In these formulas, Also, a higher value reflects a better accuracy. Table 2 shows the recall and the precision of

 

APM (Apriori algorithm), for a wide range of minsup, n, and minprob values. As it can observed that the precision and recall values are always higher than 98 percent. Since The proposed algorithm PA returns the same PFIs as APM, it is also highly accurate result. Both precision and recall are used to compare the performance of mining algorithms.

**Table 2. Recall and Precision of APM**

| minsup | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 |
|---|---|---|---|---|---|
| Recall | 1 | 1 | 1 | 1 | 1 |
| Precision | 0.997 | 1 | 1 | 1 | 1 |

(a) Recall and Precision vs. minsup

| minprob | 0.1 | 0.3 | 0.5 | 0.7 | 0.9 |
|---|---|---|---|---|---|
| Recall | 1 | 1 | 1 | 1 | 1 |
| Precision | 0.986 | 1 | 0.985 | 1 | 1 |

(b) Recall and Precision vs. minprob

| n | 1k | 4k | 10k | 50k | 100k |
|---|---|---|---|---|---|
| Recall | 1 | 1 | 1 | 1 | 1 |
| Precision | 0.987 | 0.988 | 1 | 1 | 1 |

(c) Recall and Precision vs. n

**8.1 Efficiency of PA vs. APM .**

next compare PA and APM, which both yield PFIs. The given Figure 3(a) shows that PA is faster than APM over different minsup values. minsup is taken on the X-axis and runtime is taken on the Y-axis. On increasing the minsup value the runtime decreases. Fig.3(b) examines the algorithms under a wide range of minprob values. Again, PA runs faster than APM. Fig. 3(b) examines the effect of using different minprob values. Minprob is taken on X-axis and running time is taken on the Y-axis. The details are of these distributions are listed in Table 3.

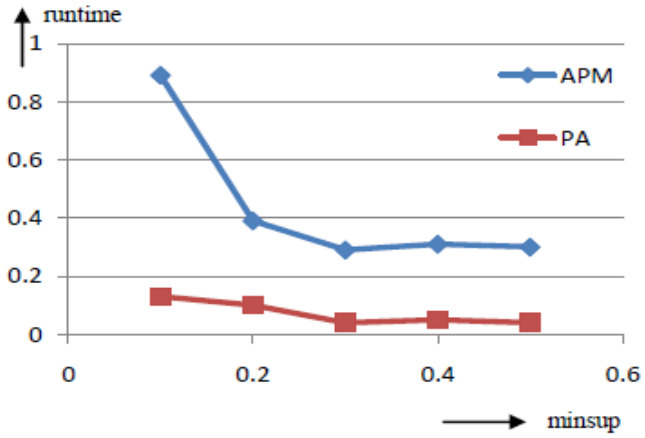


**Fig.10(a) minsup vs. Runtime**

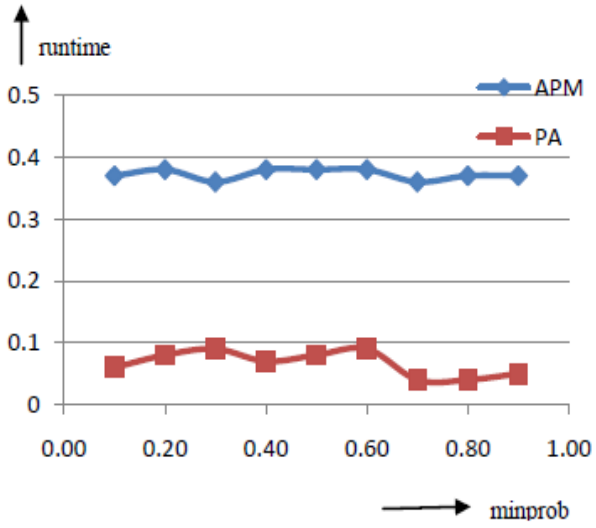


**Fig.3(b) minprob vs. Runtime**

It is observed that the algorithm PA performs better than APM over different types of distributions. The consistently high performance gain demonstrated by AP can be explained. On observing the figure 3(a) the existing algorithm takes longer time than the proposed algorithm(PA), since the on increasing the minsup value against the runtime the curve gradually decreases in existing algorithm and also maintains constant behaviour in the proposed algorithm. The graph given in the figure 3(b) depicts the difference in performs between the existing algorithm APM and the proposed algorithm PA. In this graph also the developed or proposed algorithm is faster than the existing algorithm. The below table 3 provides the details of the recall and precision of PA algorithm.

**TABLE 3. Recall and Precision of PA**

| minsup | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 |
|---|---|---|---|---|---|
| Recall | 1 | 1 | 1 | 1 | 1 |
| Precision | 0.998 | 1 | 1 | 1 | 1 |

(a) Recall and Precision vs. minsup

| minprob | 0.1 | 0.3 | 0.5 | 0.7 | 0.9 |
|---|---|---|---|---|---|
| Recall | 1 | 1 | 1 | 0.970 | 1 |
| Precision | 0.986 | 1 | 1 | 1 | 1 |

(b) Recall and Precision vs. minprob

| n | 1k | 5k | 10k | 50k | 100k |
|---|---|---|---|---|---|
| Recall | 1 | 1 | 1 | 1 | 1 |
| Precision | 1 | 1 | 1 | 1 | 0.985 |

(c) Recall and Precision vs. n

 

The experiments on the tuple uncertainty model and on the synthetic data set were also performed. Since they are similar to the results presented above, The most representative ones. For the accuracy aspect, the recall and precision values of approximate results on these data sets are still higher than 98 percent. Thus, the proposed algorithm can return accurate results.

## IX. CONCLUSIONS

In this paper, we propose a approach to extract threshold-based PFIs from large uncertain databases. Its main idea is to the s-pmf of a PFI by some common probability model, so that a PFI can be verified quickly. This approach supports retrieving PFIs from evolving databases. The experimental results show that the proposed algorithm is highly efficient and accurate. It supports both attribute- and tuple uncertain data approach to develop other mining algorithm (e.g., clustering and classification) on uncertain data. It is also interesting to study efficient mining algorithm for handling tuple. Another interesting work is to investigate PFI mining algorithm for probability models that capture correlation among attributes and tuples.

## X. REFERENCES

[1] A. Veloso, W. Meira Jr., M. de Carvalho, B. Poˆ ssas, S. Parthasarathy, and M.J. Zaki, "Mining Frequent Itemsets in Evolving Databases," Proc. Second SIAM Int‟l Conf. Data Mining (SDM), 2002.

[2] C. Aggarwal, Y. Li, J. Wang, and J. Wang, "Frequent Pattern Mining with Uncertain Data, ", Proc. 15th ACM SIGKDD Int‟l Conf. Knowledge Discovery and Data Mining (KDD), 2009.

[3] C. Aggarwal and P. Yu, "A Survey of Uncertain Data Algorithm and Applications," IEEE Trans Knowledge and Data Eng., vol. 21, no. 5, pp. 609-623, May 2009.

[4] R. Agrawal, T. Imieliμnski, and A. Swami, "Mining Association Rules between Sets of Items in Large Databases," Proc.ACM SIGMOD Int‟l Conf. Management of Data, 1993.

[5 ] O. Benjelloun, A.D. Sarma, A. Halevy, and J. Widom, "ULDBs: Databases with Uncertainty and Lineage," Proc. 32ndInt‟l Conf. Very Large Data Bases (VLDB), 2006.

[6] T. Bernecker, H. Kriegel, M. Renz, F. Verhein, and A. Zuefle,"Probabilistic Frequent Itemset Mining in Uncertain Databases," Proc. 15th ACM SIGKDD Int‟lConf. Knowledge Discovery and Data Mining (KDD), 2009.

[7] C.J. van Rijsbergen, Information Retrieval. Butterworth, 1979.

[8] L.L. Cam, "An Approximation Theorem for the Poisson Binomial Distribution," Pacific J. Math., vol. 10, pp. 1181-1197,1960.

[9] H. Cheng, P. Yu, and J. Han, " Frequent Itemset Mining in the Presence of Random Noise," Proc. Soft Computing for Knowledge Discovery and Data Mining, pp. 363-389, 2008.

[10] R. Cheng, D. Kalashnikov, and S. Prabhakar, "Evaluating Probabilistic Queries over Imprecise Data," Proc. ACM SIGMOD Int‟l Conf. Management of Data, 2003.

[11] D. Cheung, J. Han, V. Ng, and C. Wong, "Maintenance of Discovered Association Rules in Large Databases: An Updating Technique," Proc. 12th Int‟l Conf. Data Eng. (ICDE), 1996.

[12] D. Cheung, S.D. Lee, and B. Kao, "A General Technique for Maintaining Discovered Association Rules," Proc. fifth Int‟l Conf. Database Systems for Advanced Applications (DASFAA), 1997.

[13] W. Cheung and O.R. Zaı¨ane, " Mining of Frequent Patterns without Candidate Generation or Support Constraint," Proc. Seventh Int‟l Database Eng. and Applications Symp. (IDEAS), 2003.

[14] C.K. Chui, B. Kao, and E. Hung, "Mining Frequent Itemsets from Uncertain Data," Proc. 11th Pacific-Asia Conf. advancesin Knowledge Discovery and Data Mining (PAKDD), 2007.

[15] G. Cormode and M. Garofalakis, "Sketching Probabilistic Data Streams," Proc. ACM SIGMOD Int‟l Conf. Management of Data, 2007.

[16] N. Dalvi and D. Suciu, "Efficient Query Evaluation on Probabilistic Databases," Proc. 13th Int‟l Conf. Very Large Data Bases (VLDB), 2004.

[17] A. Deshpande, C. Guestrin, S.Madden, J. Hellerstein, and W. Hong, "Model-Driven Data Acquisition in Sensor Networks," Proc. 13th Int‟l Conf. Very Large Data Bases (VLDB), 2004.

[18] J. Han, J. Pei, and Y. Yin,"Mining Frequent Patterns without Candidate Generation," Proc. ACM SIGMOD Int‟l Conf. Management of Data, 2000.

[19] J. Huang, "MayBMS: A Probabilistic Database Management System," Proc. 35th ACM SIGMOD Int‟l Conf. Management of Data, 2009.

[20] R. Jampani, L. Perez, M. Wu, F. Xu, C. Jermaine, and P. Haas, "MCDB: A Monte Carlo Approach to Managing Uncertain Data," Proc. ACM SIGMOD Int‟l Conf. Management of Data 2008.

[21] J. Ren, S.D. Lee, X. Chen, B. Kao, R. Cheng, and D.W. Cheung" Naive Bayes Classification of Uncertain Data," Proc. IEEE Ninth Int‟l Conf. Data Mining (ICDM), 2009.

[22] N. Khoussainova, M. Balazinska, and D. Suciu, "Towards Correcting Input Data Errors Probabilistically Using Integrity Constraints," Proc. Fifth ACM Int‟l Workshop Data Eng. For Wireless and Mobile Access (MobiDE), 2006.

[23] H. Kriegel and M. Pfeifle, "Density-Based Clustering of Uncertain Data," Proc. ACM SIGKDD Int‟l Conf. Knowledge Discovery in Data Mining (KDD), 2005.

[24] C. Kuok, A. Fu, and M. Wong, "Mining Fuzzy Association Rules in Databases," SIGMOD Record, vol. 27, no. 1, pp. 41-46, 1998. Proc. IEEE Fifth Int‟l Conf. Data Mining (ICDM), 2005

[25] C.K.-S. Leung, Q.I. Khan, and T. Hoque, "Cantree: A Tree Structure for Efficient Mining of Frequent Patterns,". Proc. IEEE Fifth Int‟l Conf. Data Mining (ICDM), 2005.

[26] A.Lu, Y. Ke, J. Cheng, and W. Ng, "Mining Vague Association Rules," Proc. 12th Int‟l Conf. Database Systems for Advanced Applications (DASFAA), 2007.

[27] M. Mutsuzaki, "Trio-One: Layering Uncertainty and Lineage on a Conventional DBMS," Proc. Third Biennial Conf. Innovative Data Systems Research (CIDR), 2007.

[28] P. Sistla, O. Wolfson, S. Chamberlain, and S. Dao, "Querying the Uncertain Position of Moving Objects,"

 

Temporal Databases: Research and Practice, Springer Verlag, 1998.
[29] C. Stein, Computation of Expectations, Lecture Notes Monograph Series, vol. 7, Inst. of Math. Statistics, 1986.
[30] L. Sun, R. Cheng, D.W. Cheung, and J. Cheng, "Mining Uncertain Data with Probabilistic Guarantees," Proc. 16th ACM SIGKDD Int‟l Conf. Knowledge Discovery and Data Mining, 2010.
[31] T. Jayram et al., "Avatar Information Extraction System," IEEE Data Eng. Bull., vol. 29, no. 1, pp. 40-48, Mar. 2006.
[32] S. Tsang, B. Kao, K.Y. Yip, W.-S. Ho, and S.D. Lee., "Decision Trees for Uncertain Data," Proc. IEEE Int‟l Conf. Data Eng. (ICDE), 2009.
[33] L. Wang, R. Cheng, S.D. Lee, and D. Cheung, "Accelerating Probabilistic Frequent Itemset Mining: A Approach," Proc.19th ACM Int‟l Conf. Information and Knowledge Management (CIKM), 2010.
[34] M. Yiu, N. Mamoulis, X. Dai, Y. Tao, and M. Vaitis, "Efficient Evaluation of Probabilistic Advanced Spatial Queries on existentially Uncertain Data," IEEE Trans Knowledge and Data Eng., vol. 21, no. 9, pp. 108-122, Jan. 2009.
[35] Q. Zhang, F. Li, and K. Yi, "Finding Frequent Items in Probabilistic Data," Proc. ACM SIGMOD Int‟l Conf. Management of Data, 2008.

 