Mining Probabilistically Frequent Sequential Patterns in Large Uncertain Databases

Zhou Zhao, Da Yan and Wilfred Ng

Abstract—Data uncertainty is inherent in many real-world applications such as environmental surveillance and mobile tracking. Mining sequential patterns from inaccurate data, such as those data arising from sensor readings and GPS trajectories, is important for discovering hidden knowledge in such applications. In this paper, we propose to measure pattern frequentness based on the *possible world semantics*. We establish two uncertain sequence data models abstracted from many real-life applications involving uncertain sequence data, and formulate the problem of mining *probabilistically frequent sequential patterns* (or *p-FSPs*) from data that conform to our models. However, the number of possible worlds is extremely large, which makes the mining prohibitively expensive. Inspired by the famous *PrefixSpan* algorithm, we develop two new algorithms, collectively called *U-PrefixSpan*, for p-FSP mining. *U-PrefixSpan* effectively avoids the problem of "possible worlds explosion", and when combined with our four pruning and validating methods, achieves even better performance. We also propose a fast validating method to further speed up our *U-PrefixSpan* algorithm. The efficiency and effectiveness of *U-PrefixSpan* are verified through extensive experiments on both real and synthetic datasets.

Index Terms—Frequent patterns, uncertain databases, approximate algorithm, possible world semantics.

1 Introduction

Data uncertainty is inherent in many real-world applications such as sensor data monitoring [13], RFID localization [12] and location-based services [11], due to environmental factors, device limitations, privacy issues, etc. As a result, uncertain data mining has attracted a lot of attention in recent research [19].

The problem of mining *Frequent Sequential Patterns* (*FSPs*) from deterministic databases has attracted a lot of attention in the research community due to its wide spectrum of real life applications [4], [5], [6], [7], [8]. For example, in mobile tracking systems, FSPs can be used to classify or cluster moving objects [2]; and in biological research, FSP mining helps discover correlations among gene sequences [3].

In this paper, we consider the problem of mining FSPs in the context of uncertain sequence data. In contrast to previous work that adopts *expected support* to measure pattern frequentness, we propose to define pattern frequentness based on the *possible world semantics*. This approach leads to more effective mining of high quality patterns with respect to a formal probabilistic data model. We develop two uncertain sequence data models (sequence-level and element-level models) abstracted from many real-life applications involving uncertain sequence data. Based on the models we define the problem of mining *probabilistically frequent sequential patterns* (or *p-FSPs*). We now introduce our data models through the following examples.

• *Zhou Zhao, Da Yan and Wilfred Ng are with the Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Hong Kong, China. E-mail: zhaozhou@cse.ust.hk, yanda@cse.ust.hk, wilfred@cse.ust.hk.*

SID	Sequence Instance	Probability
s_1	$s_{11} = ABC$	1
s_2	$s_{21} = AB$	0.9
	$s_{22} = BC$	0.1

(a)

Possible World	Probability
$pw_1 = \{s_{11}, s_{21}\}$	$1 \times 0.9 = 0.9$
$pw_2 = \{s_{11}, s_{22}\}$	$1 \times 0.1 = 0.1$

(b)

Fig. 1. Sequence-Level Uncertain Data Model

Consider a wireless sensor network (WSN) system, where each sensor continuously collects readings of environmental parameters, such as temperature and humidity, within its detection range. In such a case, the readings are inherently noisy, and can be associated with a confidence value determined by, for example, the stability of the sensor. Figure 1(a) shows a possible set of readings from a WSN application that monitors temperature. Let us assume that each sensor reports temperature ranges A, B and C, (for instance reading A represents $[5°, 7°)$, reading B represents $[7°, 9°)$, and reading C represents $[9°, 11°)$), and a new reading is appended to the sequence of already reported readings whenever the temperature range changes. We also assume that each region is associated with a group of sensors. For example, s_{11} is the reading sequence detected by a sensor in one region within a time period, and s_{21} and s_{22} are the reading sequences detected by two different sensors in another region within that time period.

In Figure 1(a), we assume that the reading sequences detected by different sensors in a region are exclusive to each other, e.g. the temperature sequence in the region represented by s_2 has 90% (or 5%) probability to be $\{A, B\}$ (or $\{B, C\}$). The remaining 5% probability is for the case when there is no new readings reported in that region. Besides, the reading sequences from different regions are

SID	Probabilistic Element
s_1	$s_1[1] = \{(A, 0.95)\}$, $s_1[2] = \{(B, 0.95), (C, 0.05)\}$
s_2	$s_2[1] = \{(A, 1)\}$, $s_2[2] = \{(B, 1)\}$

(a)

Possible World	Probability
$pw_1 = \{B, AB\}$	$(1-0.95) \times 0.95 \times 1 \times 1 = 0.0475$
$pw_2 = \{C, AB\}$	$(1-0.95) \times 0.05 \times 1 \times 1 = 0.0025$
$pw_3 = \{AB, AB\}$	$0.95 \times 0.95 \times 1 \times 1 = 0.9025$
$pw_4 = \{AC, AB\}$	$0.95 \times 0.05 \times 1 \times 1 = 0.0475$

(b)

Fig. 2. Element-Level Uncertain Data Model

assumed to be independent. We call such a data model the *sequence-level uncertain model*. Notably, probabilistic sequences such as s_1 and s_2 are called *x-tuples* in the Trio system [21].

Figure 1(b) shows the set of possible worlds derived from the uncertain sequence data presented in Figure 1(a). Since the occurrences of different probabilistic sequences are mutually independent, the probability of a possible world pw can be computed as the product of the occurrence probability of each sequence in pw. For example, $Pr(pw_1) = Pr(s_{11}) \times Pr(s_{21}) = 0.9$ holds.

To measure the frequentness of patterns, existing studies adopt the notion of expected support, such as frequent itemsets [15], [18] and frequent subsequences [1]. Accordingly, the expected support of a sequential pattern α in an uncertain database can be evaluated as follows: for a sequence-level probabilistic sequence s, if we denote $\alpha \sqsubseteq s$ to be the event that pattern α occurs in s, then the expected support of α in database D is defined as $expSup(\alpha) = \sum_{s \in D} Pr\{\alpha \sqsubseteq s\}$ according to the linearity of expectation.

However, we argue that expected support fails to reflect pattern frequentness in many cases. To illustrate the weakness of $expSup(\alpha)$, we consider $\alpha = AB$ in the dataset shown in Figure 1. The expected support of pattern AB is $Pr(s_{11}) + Pr(s_{21}) = 1.9$, which is not considered as frequent when the minimum support $\tau_{sup} = 2$. Nevertheless, pattern AB occurs twice in pw_1, and once in both pw_2 and pw_3. Thus, if we denote the support of AB in database D as $sup(AB)$, then $Pr\{sup(AB) \geq \tau_{sup}\} = Pr(pw_1) = 90\%$ when $\tau_{sup} = 2$. Therefore, we miss the important sequential pattern AB in this example.

While the *sequence-level uncertain model* is fundamental in a lot of real-life applications, many applications follow a different model. Consider the uncertain sequence database shown in Figure 2(a), where sequences s_1 and s_2 record the tracking paths of two users. Path s_1 contains two uncertain location elements, $s_1[1]$ and $s_1[2]$. The uncertain location $s_1[1]$ has 95% probability to be A and 5% probability to be a misreading (i.e. does not occur), while location $s_1[2]$ has 95% probability to be B and 5% probability to be C. We call such a model the *element-level uncertain model*, where each probabilistic sequence in the database is composed of a sequence of uncertain elements that are mutually independent, and each uncertain element is an x-tuple.

Figure 2(b) shows the possible world space of the dataset shown in Figure 2(a). We can easily compute the probabilities of the possible worlds. For example, $Pr(pw_3) = Pr\{s_1[1] = A\} \times Pr\{s_1[2] = B\} \times Pr\{s_2[1] = A\} \times Pr\{s_2[2] = B\} = 0.9025$.

Note that the expected support of AB is $expSup(AB) = Pr\{s_1 = AB\} + Pr\{s_2 = AB\} = 0.95 \times 0.95 + 1 \times 1 = 1.9025$, and thus AB is not considered as frequent when $\tau_{sup} = 2$. However, $Pr\{sup(AB) \geq \tau_{sup}\} = Pr(pw_3) = 90.25\%$ when $\tau_{sup} = 2$, which is very likely to be frequent in the probabilistic sense.

The above example illustrates that *expected support* fails again to identify some probabilistically frequent patterns. In fact, using expected support may also give rise to some probabilistically infrequent patterns as the result [16]. Intuitively, expected support does not capture the distribution of support. A distribution may be centralized or relatively flat but the expected support does not contain this information. Therefore, we propose to evaluate the frequentness of a sequential pattern by adhering to the probability theory. This gives rise to the idea of *probabilistic frequentness*, which is able to capture the intricate relationships between uncertain sequences.

However, the problem of p-FSP mining is challenging, since each uncertain sequence database D corresponds to many possible deterministic database instances (or *possible worlds*), the number of which is exponential to the number of uncertain sequences in D. To tackle this problem, we propose two new algorithms, collectively called *U-PrefixSpan*, to mine p-FSPs from uncertain data that conform to our two uncertain data models. *U-PrefixSpan* adopts the *prefix-projection* recursion framework of the *PrefixSpan* algorithm [4] in a new algorithmic setting, and effectively avoids the problem of "possible worlds explosion". Our contributions are summarized as follows:

- To our knowledge, this is the first work that attempts to solve the problem of p-FSP mining, the techniques of which are successfully applied in an RFID application for trajectory pattern mining.
- We consider two general uncertain sequence data models that are abstracted from many real-life applications involving uncertain sequence data: the *sequence-level uncertain model*, and the *element-level uncertain model*.
- Based on the *prefix-projection* method of *PrefixSpan*, we design two new *U-PrefixSpan* algorithms that mine p-FSPs from uncertain data conforming to our models.
- Pruning techniques and a fast validating method are developed to further improve the efficiency of *U-PrefixSpan*, which is verified by extensive experiments.

The rest of the paper is organized as follows: Section 2 reviews the related work and introduces the *PrefixSpan* algorithm. Then we provide some preliminaries on mining p-FSPs in Section 3. The *U-PrefixSpan* algorithm for the sequence-level model is presented in Section 4, and the *U-PrefixSpan* algorithm for the element-level model is

SID	Sequence
s_1	$ABCBC$
s_2	$BABC$
s_3	AB
s_4	BC

(a) D

$\xrightarrow{A}$

SID	Sequence
s_1	$\_BCBC$
s_2	$\_BC$
s_3	$\_B$

(b) $D|_A$

$\xrightarrow{B}$

SID	Sequence
s_1	$\_CBC$
s_2	$\_C$
s_3	$\_$

(c) $D|_{AB}$

$\xrightarrow{C}$

SID	Sequence
s_1	$\_BC$
s_2	$\_$

(d) $D|_{ABC}$

Fig. 3. Illustration of PrefixSpan

described in Section 5. In Section 6, we introduce the fast validating method. In Section 7, we verify the efficiency and effectiveness of *U-PrefixSpan* through extensive experiments on both real and synthetic datasets. Finally, we conclude our paper in Section 8.

2 Related Work

A comprehensive survey of traditional data mining problems such as frequent pattern mining in the context of uncertain data can be found in [19]. We only detail some concepts and issues arising from traditional sequential pattern mining and the mining of uncertain data.

2.1 Traditional Sequential Pattern Mining

The problem of sequential pattern mining has been well studied in the literature in the context of deterministic data, and many algorithms have been proposed to solve this problem, including *PrefixSpan* [4], *SPADE* [6], *FreeSpan* [7] and *GSP* [8].

PrefixSpan is demonstrated to be superior to other sequence mining algorithms such as GSP and FreeSpan, due to its *prefix-projection* technique [4]. It has been used successfully in many applications such as trajectory mining [2]. We now review the *prefix-projection* technique of *PrefixSpan*, which is related to our proposed algorithms.

PrefixSpan. For ease of presentation, we denote $\alpha\beta$ to be the sequence resulted from appending sequence β with sequence α. As mentioned in Section 1, $\alpha \sqsubseteq s$ corresponds to the event that sequence α occurs as a subsequence of s. We now present some concepts that are necessarily for understanding *PrefixSpan*.

Definition 1: Given a sequential pattern α and a sequence s, the α-projected sequence $s|_\alpha$ is defined to be the suffix γ of s such that $s = \beta\gamma$ with β being the minimal prefix of s satisfying $\alpha \sqsubseteq s$.

To highlight the fact that γ is a suffix, we write it as "$\_\gamma$". As an illustration of Definition 1, when $\alpha = BC$ and $s = ABCBC$, we have $\beta = ABC$ and $s|_\alpha = \_\gamma = \_BC$.

Definition 2: Given a sequential pattern α and a sequence database D, the α-projected database $D|_\alpha$ is defined to be the set $\{s|_\alpha \mid s \in D \wedge \alpha \sqsubseteq s\}$.

Note that if $\alpha \not\sqsubseteq s$, then the minimal prefix β of s satisfying $\alpha \sqsubseteq \beta$ does not exist, and therefore s is not considered in $D|_\alpha$.

Consider the sequence database D shown in Figure 3(a). The projected databases $D|_A$, $D|_{AB}$ and $D|_{ABC}$ are shown in Figures 3(b), (c) and (d), respectively.

PrefixSpan finds the frequent patterns (with support of at least τ_{sup}) by recursively checking the frequentness of patterns with growing lengths. In each iteration, if the current pattern α is found to be frequent, it will recurse on all the possible patterns α' constructed by appending one more element to α. *PrefixSpan* checks whether a pattern α is frequent using the projected database $D|_\alpha$, which can be constructed from the projected database of the previous iteration. Figure 3 presents one recursion path when $\tau_{sup} = 2$, where, for example, $s_1|_{ABC}$ in $D|_{ABC}$ is obtained by removing the element C (above the third arrow) from $s_1|_{AB}$ in $D|_{AB}$. The *bi-level* projection technique of *PrefixSpan* is a disk-based algorithm which reduces the *IO* cost using S-matrix. In this paper, we focus on single-level projection, since the advantage of bi-level projection may not be significant when the pesudo-projected database is stored in main memory.

2.2 Pattern Mining on Uncertain Data

Frequent itemset mining, graph pattern mining and sequential pattern mining are important pattern mining problems that have been studied in the context of uncertain data. For the problem of frequent pattern mining, earlier work commonly uses expected support to measure pattern frequentness [15], [18], [10]. However, some have found that the use of expected support may render important patterns missing [16], [17]. As a result, recent research focuses more on using probabilistic support, such as [17], [14], [24], [25], [26], [27], [28]. The work mainly utilizes algorithms based on dynamic programming and divide-and-conquer in order to validate the probabilistic frequentness of an itemset pattern or a subgraph pattern. However, these techniques cannot be directly applied for checking the probabilistic frequentness of a sequential pattern. This is because the projection of a frequent sequential pattern on uncertain databases is fundamentally different from the projections of an frequent itemset or a frequent subgraph.

As for the problem of sequential pattern mining on uncertain data, [1] is the only existing work we are aware of. However, all the models proposed by [1] are merely variations of the sequence-level model in essence, and the work evaluates the frequentness of a pattern based on its expected support. The problem of mining long sequential patterns in a noisy environment has also been studied in [20]. However, their *compatibility matrix* model of uncertainty is very different from, and not as general as, our uncertain sequence data models. It is worth mentioning that models similarly to our probabilistic sequence models have been used in studies concerning similarity join [22], [23].

3 Preliminaries

In this section we discuss several fundamental concepts.

Presence Probability. The probability of the presence of a pattern α in a probabilistic sequence s is given by

$$Pr\{\alpha \subseteq s\} = \sum_{\alpha \subseteq s_i} Pr(pw_i) \tag{1}$$

where s_i is a deterministic instance of probabilistic sequence s in the possible word pw_i. $Pr(pw_i)$ is the existence probability of possible world pw_i.

Expected Support. Formally, the concept of *expected support* is as follows.

Definition 3 (Expected Support): The expected support of a pattern α, denoted by $expSup(\alpha)$, is defined as the sum of the expected probabilities of the presence of α in each of the sequences in the databases.

The pattern α is said to be *expectably frequent* if $expSup(\alpha)$ is greater than specified support threshold τ_{sup}.

Support as a random variable. We use $sup(\alpha)$ as a random variable in the context of uncertain databases.

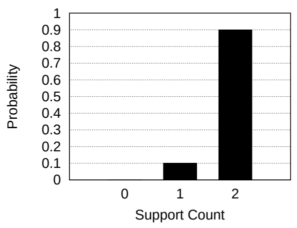


Fig. 4. Probability Distribution of $sup(AB)$

Given a sequence-level or an element-level uncertain sequence database D, we denote its possible world space as $\mathcal{PW} = \{pw_1, pw_2, \ldots, pw_{|\mathcal{PW}|}\}$. We also denote by $sup_i(\alpha)$ the support of pattern α in a possible world $pw_i \in \mathcal{PW}$. Since pw_i is a deterministic database instance, $sup_i(\alpha)$ is simply a count that is equal to $|\{s \in pw_i | \alpha \sqsubseteq s\}|$. Note that each possible world pw_i is associated with an occurrence probability $Pr(pw_i)$, and therefore, given a pattern α, each possible world pw_i corresponds to a pair $(sup_i(\alpha), Pr(pw_i))$. In the example presented in Figure 1, given pattern AB, the possible worlds pw_1, pw_2 and pw_3 correspond to pairs $(2, 0, 9)$, $(1, 0.05)$ and $(1, 0.05)$, respectively. Therefore, we have

- $Pr\{sup(AB) = 2\} = Pr(pw_1) = 0.9$;
- $Pr\{sup(AB) = 1\} = Pr(pw_2) + Pr(pw_3) = 0.1$;
- $Pr\{sup(AB) = 0\} = 0$.

Note that $sup(AB)$ is a random variable whose probability distribution is depicted in Figure 4. Generally, for any pattern α, its support $sup(\alpha)$ can be represented by (1) a *probability mass function* (pmf), denoted as $f_\alpha(c)$ where c is a count, and (2) a *cumulative distribution function* (cdf), denoted as $F_\alpha(c) = \sum_{i=0}^{c} f_\alpha(i)$. For a database with n probabilistic sequences (i.e. $|D| = n$), $sup(\alpha)$ can be at most n, and therefore the domain of c is $\{0, 1, \ldots, n\}$.

Formally, $f_\alpha(c)$ is given by the following formula:

$$f_\alpha(c) = \sum_{pw_i \in \mathcal{PW} \ s.t. \ sup_i(\alpha) = c} Pr(pw_i).$$

Probabilistic frequentness. We now introduce the concept of *probabilistic frequentness* (or simply *$(\tau_{sup}, \tau_{prob})$-frequentness*):

Definition 4 (Probabilistic Frequentness): Given a probability threshold τ_{prob} and a support threshold τ_{sup}, pattern α is probabilistically frequent (or $(\tau_{sup}, \tau_{prob})$-frequent) iff

$$Pr\{sup(\alpha) \geq \tau_{sup}\} \ \geq \ \tau_{prob}. \tag{2}$$

The L.H.S. of Equation 2 can be represented as

$$Pr\{sup(\alpha) \geq \tau_{sup}\} = \sum_{c=\tau_{sup}}^{n} f_\alpha(c) = 1 - F_\alpha(\tau_{sup} - 1). \tag{3}$$

Pruning infrequent patterns. Next, we present our three pruning rules for pruning probabilistically infrequent patterns:

- **R1 CntPrune.** Let us define $cnt(\alpha) = |\{s \in D \,|\, Pr\{\alpha \sqsubseteq s\} > 0\}|$, then pattern α is not *$(\tau_{sup}, \tau_{prob})$-frequent* if $cnt(\alpha) < \tau_{sup}$.

  Proof: When $cnt(\alpha) < \tau_{sup}$, $Pr\{sup(\alpha) \geq \tau_{sup}\} \leq Pr\{sup(\alpha) > cnt(\alpha)\} = 0$. □
- **R2 MarkovPrune.** Pattern α is not *$(\tau_{sup}, \tau_{prob})$-frequent* if $expSup(\alpha) < \tau_{sup} \times \tau_{prob}$.

  Proof: According to Markov's inequality, $expSup(\alpha) < \tau_{sup} \times \tau_{prob}$ implies $Pr\{sup(\alpha) \geq \tau_{sup}\} \leq \frac{expSup(\alpha)}{\tau_{sup}} < \tau_{prob}$. □
- **R3 ExpPrune.** Let $\mu = expSup(\alpha)$ and $\delta = \frac{\tau_{sup} - \mu - 1}{\mu}$. When $\delta > 0$, pattern α is not *$(\tau_{sup}, \tau_{prob})$-frequent* if

$$\begin{cases} \delta \geq 2e - 1, & 2^{-\delta\mu} < \tau_{prob}; \\ 0 < \delta < 2e - 1, & e^{-\frac{\delta^2 \mu}{4}} < \tau_{prob}. \end{cases}$$

  Proof: According to Chernoff Bound, we have

$$Pr\{sup(\alpha) > (1+\delta)\mu\} < \begin{cases} 2^{-\delta\mu}, & \delta \geq 2e - 1 \\ e^{-\frac{\delta^2 \mu}{4}}, & 0 < \delta < 2e - 1 \end{cases},$$

  and if we set $\delta = \frac{\tau_{sup} - \mu - 1}{\mu}$, i.e. $(1+\delta)\mu = \tau_{sup} - 1$, we have $Pr\{sup(\alpha) > (1+\delta)\mu\} = Pr\{sup(\alpha) \geq \tau_{sup}\}$ □

CntPrune and *ExpPrune* are also used in [14] to prune infrequent itemsets. Note that these pruning rules only require one pass of the database to determine whether a pattern can be pruned.

Frequentness validating. If α cannot be pruned, we have to check whether Equation (2) holds. According to Equation (3), this is equivalent to computing $f_\alpha(c)$.

In fact, *evaluating $f_\alpha(c)$ on α-projected (uncertain) database $D|_\alpha$ is equivalent to evaluating $f_\alpha(c)$ on D*, since $\forall s \notin D|_\alpha$, $Pr\{\alpha \sqsubseteq s\} = 0$. Thus, we always compute $f_\alpha(c)$ on the smaller projected database $D|_\alpha$. We will discuss how to perform sequence projection in our sequence-level (and element-level) uncertain model in Section 4 (and Section 5).

We compute $f_\alpha(c)$ on $D|_\alpha$ by using the divide-and-conquer strategy. Given a set S of probabilistic sequences, we divide it into two partitions S_1 and S_2. Let $f_\alpha^S(c)$ be the pmf of $sup(\alpha)$ on S. Then our ultimate goal is to compute $f_\alpha^{D|_\alpha}(c)$.

We now consider how to obtain $f_\alpha^S(c)$ from $f_\alpha^{S_1}(c)$ and $f_\alpha^{S_2}(c)$. Let us denote $sup^S(\alpha)$ to be the support of α on S. Note that $sup^S(\alpha)$ is a random variable, and $sup^{S_1}(\alpha)$ and $sup^{S_2}(\alpha)$ are independent. Obviously, $sup^S(\alpha) = sup^{S_1}(\alpha) + sup^{S_2}(\alpha)$, and $f_\alpha^S(c)$ can be computed by the following formula:

$$f_\alpha^S(c) = \sum_{i=0}^{c} f_\alpha^{S_1}(i) \times f_\alpha^{S_2}(c-i). \quad (4)$$

According to Equation (4), f_α^S is the convolution of $f_\alpha^{S_1}$ and $f_\alpha^{S_2}$. Thus, f_α^S can be computed from $f_\alpha^{S_1}$ and $f_\alpha^{S_2}$ in $O(n \log n)$ time using the *Fast Fourier Transform* (FFT) algorithm, where $n = |S|$. When S is large, this approach is much better than naïvely evaluating Equation (4) for all c, which takes $O(n^2)$ time.

Theorem 1 (Early Validating): Suppose that pattern α is *$(\tau_{sup}, \tau_{prob})$-frequent* in $S' \subseteq S$, then α is also *$(\tau_{sup}, \tau_{prob})$-frequent* in S.

Proof: Suppose that probabilistic sequence set S is divided into two partitions S_1 and S_2. It is sufficient to prove that, when α is *$(\tau_{sup}, \tau_{prob})$-frequent* in S_1, it is also *$(\tau_{sup}, \tau_{prob})$-frequent* in S.

When α is *$(\tau_{sup}, \tau_{prob})$-frequent* in S_1, according to Equation (3), we have

$$1 - F_\alpha^{S_1}(\tau_{sup} - 1) = Pr\{sup^{S_1}(\alpha) \geq \tau_{sup}\} \geq \tau_{prob}. \quad (5)$$

According to Equation (5), $F_\alpha^{S_1}(\tau_{sup} - 1) \leq 1 - \tau_{prob}$. If we can prove $F_\alpha^S(\tau_{sup} - 1) \leq F_\alpha^{S_1}(\tau_{sup} - 1)$, then we are done since this implies $F_\alpha^S(\tau_{sup} - 1) \leq F_\alpha^{S_1}(\tau_{sup} - 1) \leq 1 - \tau_{prob}$, or equivalently, $Pr\{sup^S(\alpha) \geq \tau_{sup}\} = 1 - F_\alpha^S(\tau_{sup} - 1) \geq \tau_{prob}$.

We now prove $F_\alpha^S(\tau_{sup} - 1) \leq F_\alpha^{S_1}(\tau_{sup} - 1)$. Let us denote $\tau'_{sup} = \tau_{sup} - 1$. Then, we obtain

$$\begin{aligned}
F_\alpha^S(\tau'_{sup}) &= \sum_{i+j=0}^{\tau'_{sup}} f_\alpha^{S_1}(i) \times f_\alpha^{S_2}(j) \\
&= \sum_{i=0}^{\tau'_{sup}} \sum_{j=0}^{\tau'_{sup}-i} f_\alpha^{S_1}(i) \times f_\alpha^{S_2}(j) \\
&= \sum_{i=0}^{\tau'_{sup}} f_\alpha^{S_1}(i) \times \sum_{j=0}^{\tau'_{sup}-i} f_\alpha^{S_2}(j) \\
&= \sum_{i=0}^{\tau'_{sup}} f_\alpha^{S_1}(i) \times F_\alpha^{S_2}(\tau'_{sup} - i) \\
&\leq \sum_{i=0}^{\tau'_{sup}} f_\alpha^{S_1}(i) = F_\alpha^{S_1}(\tau'_{sup}).
\end{aligned}$$

□

Algorithm 1 shows our divide-and-conquer algorithm (*PMFCheck*) which determines the *$(\tau_{sup}, \tau_{prob})$-frequentness* of pattern α in an uncertain sequence set $S = \{s_1, s_2, \ldots, s_n\}$. The input to *PMFCheck* is a vector vec_α where each element $vec_\alpha[i] = Pr\{\alpha \sqsubseteq s_i\}$.

Algorithm 1 *PMFCheck*(vec_α)

Input: probability vector: vec_α
Output: mark of frequentness: tag; pmf: f_α

```
1: if $|vec_\alpha|$=1 then
2:     $f_\alpha(0) \leftarrow 1 - vec_\alpha[1]$, $f_\alpha(1) \leftarrow vec_\alpha[1]$
3:     return $(1 - F_\alpha(\tau_{sup} - 1) \geq \tau_{prob},\ f_\alpha)$
4: Partition $vec_\alpha$ into $vec_\alpha^1$ and $vec_\alpha^2$, where $|vec_\alpha^1| = \lfloor \frac{n}{2} \rfloor$
   and $|vec_\alpha^2| = \lceil \frac{n}{2} \rceil$
5: $(tag_1,\ f_\alpha^1) \leftarrow$PMFCheck$(vec_\alpha^1)$
6: if $tag_1$ =TRUE then
7:     return $(TRUE, \emptyset)$
8: $(tag_2,\ f_\alpha^2) \leftarrow$PMFCheck$(vec_\alpha^2)$
9: if $tag_2$ =TRUE then
10:    return $(TRUE, \emptyset)$
11: $f_\alpha \leftarrow$convolution$(f_\alpha^1,\ f_\alpha^2)$
12: return $(1 - F_\alpha(\tau_{sup} - 1) \geq \tau_{prob},\ f_\alpha)$
```

PMFCheck partitions vec_α into two halves: vec_α^1 and vec_α^2 respectively as the first half S_1 and the second half S_2 of S (Line 4). If α is found to be *$(\tau_{sup}, \tau_{prob})$-frequent* in either half (Lines 6 and 9), *PMFCheck* returns *TRUE* directly (which is propagated upwards through the recursions in Lines 5 and 8). Otherwise, *PMFCheck* uses the pmfs obtained from recursion in S_1 and S_2 (i.e. f_α^1 and f_α^2), to compute the pmf of α in S in Line 11. After obtaining f_α, we can check whether α is *$(\tau_{sup}, \tau_{prob})$-frequent* in S by Equations (2) and (3) (Line 12).

The degenerated case of $S = \{s_1\}$ is handled in Lines 1–3, where $f_\alpha(0) = Pr\{sup(\alpha) = 0\} = Pr\{\alpha \not\sqsubseteq s_1\}$ and $f_\alpha(1) = Pr(sup(\alpha) = 1) = Pr(\alpha \sqsubseteq s_1)$.

Complexity Analysis: Let $T(n)$ be the running time of *PMFCheck* on input vec_α with $|vec_\alpha| = n$. Then the time costs in Lines 5 and 8 are both $T(n/2)$. Since Line 11 can be done in $O(n \log n)$ time, we have $T(n) = 2T(n/2) + O(n \log n)$, which yields $T(n) = O(n \log^2 n)$.

Pattern anti-monotonicity. Finally, we present the *pattern anti-monotonicity* property that allows us to use the *PrefixSpan*-style pattern-growth method for mining p-FSPs:

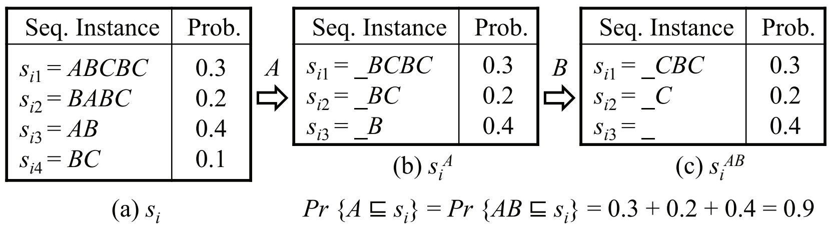
Property 1 (Pattern Anti-Monotonicity): If a pattern α is not *$(\tau_{sup}, \tau_{prob})$-frequent*, then any pattern β satisfying $\alpha \sqsubseteq \beta$ is not *$(\tau_{sup}, \tau_{prob})$-frequent.*

The proof follows from the fact that in any possible world pw where β is frequent, α must also be frequent, since for each sequence $s \in pw$, $\beta \sqsubseteq s$ implies $\alpha \sqsubseteq s$.

According to Property 1, we can stop growing α, once we find that α is probabilistically infrequent.

4 Sequence-Level U-PrefixSpan

In this section, we address the problem of p-FSP mining on data that conform to the sequence-level uncertain model. We propose a pattern-growth algorithm, called *SeqU-PrefixSpan*, to tackle this problem. Compared with *PrefixSpan*, the *SeqU-PrefixSpan* algorithm needs to address the following additional issues arising from the sequence-level uncertain model.



Seq. Instance	Prob.
$s_{i1} = ABCBC$	0.3
$s_{i2} = BABC$	0.2
$s_{i3} = AB$	0.4
$s_{i4} = BC$	0.1

(a) s_i

$\xrightarrow{A}$

Seq. Instance	Prob.
$s_{i1} = \_BCBC$	0.3
$s_{i2} = \_BC$	0.2
$s_{i3} = \_B$	0.4

(b) s_i^A

$\xrightarrow{B}$

Seq. Instance	Prob.
$s_{i1} = \_CBC$	0.3
$s_{i2} = \_C$	0.2
$s_{i3} = \_$	0.4

(c) s_i^{AB}

$Pr\ \{A \sqsubseteq s_i\} = Pr\ \{AB \sqsubseteq s_i\} = 0.3 + 0.2 + 0.4 = 0.9$

Fig. 5. Sequence Projection in Sequence-Level Model

Sequence Projection. Given a sequence-level probabilistic sequence s_i and a pattern α, we now discuss how to obtain the α-projected probabilistic sequence $s_i|_\alpha$.

Figure 5(a) shows a sequence-level probabilistic sequence s_i with four sequence instances, and Figures 5(b) and (c) present the projected sequences $s_i|_A$ and $s_i|_{AB}$, respectively. In general, $s_i|_\alpha$ is obtained by projecting each deterministic sequence instance s_{ij} of sequence s_i (denoted $s_{ij} \in s_i$) onto $s_{ij}|_\alpha$, excluding those instances that cannot be projected (due to $\alpha \not\sqsubseteq s_{ij}$), such as s_{i4} in Figure 5.

In order to achieve high space utility, we do not store $s_{ij}|_\alpha$ as a suffix sequence of s_{ij}. In fact, it is sufficient to represent $s_{ij}|_\alpha$ with a pointer to s_{ij} and the starting position of suffix $s_{ij}|_\alpha$ in s_{ij}. In our algorithm, each projected sequence instance $s_{ij}|_\alpha$ is represented as a pair $<pos, s_{ij}>$, where pos denotes the position before the starting position of suffix $s_{ij}|_\alpha$ in s_{ij}. Besides, each $s_i|_\alpha$ is represented as a list of pairs, where each pair corresponds to an instance s_{ij} and the format $(s_{ij}|_\alpha, Pr(s_{ij}))$. We illustrate our representation in Figure 5(c), which shows that $s_i|_{AB} = \{(s_{i1}|_{AB}, 0.3),\ (s_{i2}|_{AB}, 0.2),\ (s_{i3}|_{AB}, 0.4)\}$ where, for example, $s_{i1}|_{AB} = <2, s_{i1}>$.

Conceptually, the α-projected database $D|_\alpha$ is constructed by projecting each probabilistic sequence $s_i \in D$ onto $s_i|_\alpha$.

Pattern Frequentness Checking. Recall that given a projected database $D|_\alpha$, we check the *$(\tau_{sup}, \tau_{prob})$-frequentness* of pattern α by (1) computing $vec_\alpha[i] = Pr\{\alpha \sqsubseteq s_i\}$ for each projected probabilistic sequence $s_i|_\alpha \in D|_\alpha$, and then (2) determining the result by invoking *PMFCheck*(vec_α) (Algorithm 1).

Thus, the key to checking pattern frequentness is the computation of $Pr\{\alpha \sqsubseteq s_i\}$. According to the *law of total probability*, we can compute $Pr\{\alpha \sqsubseteq s_i\}$ using the following formula:

$$\begin{aligned} & Pr\{\alpha \sqsubseteq s_i\} \\ = & \sum_{s_{ij} \in s_i} Pr\{\alpha \sqsubseteq s_{ij} \mid s_i \text{ occurs as } s_{ij}\} \times Pr(s_{ij}) \\ = & \sum_{s_{ij}|_\alpha\ \in\ s_i|_\alpha} Pr(s_{ij}). \end{aligned} \tag{6}$$

In a nutshell, $Pr\{\alpha \sqsubseteq s_i\}$ is equal to the sum of the occurrence probabilities of all sequence instances whose α-projected instances belong to $s_i|_\alpha$. For example, we can check that in Figure 5(c), $Pr\{AB \sqsubseteq s_i\} = Pr(s_{i1}) + Pr(s_{i2}) + Pr(s_{i3}) = 0.9$.

Algorithm 2 *Prune*$(T|_\alpha, D|_{\alpha e})$

Input: element table $T|_\alpha$, projected probabilistic database $D|_{\alpha e}$ **Output:** element table $T|_{\alpha e}$

```
1: $T|_{\alpha e} \leftarrow \emptyset$
2: for each element $\ell \in T|_\alpha$ do
3:     Check CntPrune with pattern $\ell$ on $D|_{\alpha e}$
4:     if $\ell$ is not pruned then
5:         Check MarkovPrune with pattern $\ell$ on $D|_{\alpha e}$
6:         if $\ell$ is not pruned then
7:             Check ExpPrune with pattern $\ell$ on $D|_{\alpha e}$
8:     if $\ell$ is not pruned then
9:         $T|_{\alpha e} \leftarrow T|_{\alpha e} \cup \{\ell\}$
```

Candidate Elements for Pattern Growth. Given a pattern α, we need to examine whether another pattern β grown from α such that $\alpha \sqsubseteq \beta$ is *$(\tau_{sup}, \tau_{prob})$-frequent*.

Recall that in *PrefixSpan*, in each recursive iteration, if the current pattern α is frequent, we grow α by appending to it one element e to obtain a new pattern αe, and then recursively checking the frequentness of αe. To keep the number of such new patterns small in each growing step, we maintain an element table $T|_\alpha$ that stores only those elements e that still have a chance of making αe *$(\tau_{sup}, \tau_{prob})$-frequent*.

We now present an important property of $T|_\alpha$:

Property 2: If β is grown from α, $T|_\beta \subseteq T|_\alpha$.

Proof: Let $\beta = \alpha\gamma$. For any element $e \notin T|_\alpha$, αe is not *$(\tau_{sup}, \tau_{prob})$-frequent*, and since $\alpha e \sqsubseteq \alpha\gamma e = \beta e$, βe is also not *$(\tau_{sup}, \tau_{prob})$-frequent* according to pattern anti-monotonicity, which implies $e \notin T|_\beta$. □

As a special case of Property 2, we have $T|_{\alpha e} \subseteq T|_\alpha$. Property 2 guarantees that an element pruned from $T|_\alpha$ does not need to be considered when checking a pattern grown from α later.

We construct $T|_{\alpha e}$ from $T|_\alpha$ during the pattern growth in Algorithm 2. Note that checking our three pruning rules with element ℓ on $D|_{\alpha e}$ is equivalent to checking them with pattern $\alpha e \ell$ on D, since for any probabilistic sequence s_i whose αe-projected sequence does not exist in $D|_{\alpha e}$, $Pr\{\alpha e \ell \sqsubseteq s_i\} = 0$.

***SeqU-PrefixSpan* Algorithm.** We now present Algorithm 3 for growing patterns. Given a sequence-level probabilistic database $D = \{s_1, \ldots, s_n\}$, we grow patterns starting from $\alpha = \emptyset$. Thus, our projected sequence/instance is $D|_\emptyset = \{s_1|_\emptyset, s_2|_\emptyset, \ldots, s_n|_\emptyset\}$, where for each sequence

$s_i|_\emptyset$, its instance $s_{ij}|_\emptyset = <0, Pr(s_{ij})>$. Here the "pos" field is the one before the first position, which is 0. Let T_0 be the table of all possible elements in D. The mining algorithm begins by invoking the following functions:

- $T|_\emptyset \leftarrow$*Prune*$(T_0, D|_\emptyset)$;
- For each element $e \in T|_\emptyset$, call *SeqU-PrefixSpan*$(e, D|_\emptyset, T|_\emptyset)$.

Essentially, *SeqU-PrefixSpan* recursively performs pattern growth from the previous pattern α to the current $\beta = \alpha e$, by appending an element $e \in T|_\alpha$. In Lines 2–12, we construct the current projected probabilistic database $D|_\beta$ using the previous projected probabilistic database $D|_\alpha$. Specifically, for each projected probabilistic sequence $s_i|_\alpha \in D|_\alpha$, we compute $Pr\{\beta \sqsubseteq s_i\}$ as $pr(s_i|_{\alpha e})$ in Lines 3–9, and if $Pr\{\beta \sqsubseteq s_i\} > 0$, we add $s_i|_\beta$ (constructed from $s_i|_\alpha$) into $D|_\beta$ and append this probability to vec_β (Lines 10–12), which is used to determine whether β is *$(\tau_{sup}, \tau_{prob})$-frequent* by invoking *PMFCheck*(vec_β) in Line 13.

To compute $Pr\{\beta \sqsubseteq s_i\}$ using Equation (6), we first initialize $pr(s_i|_{\alpha e})$ to 0 (Line 3). Whenever we find that $s_{ij} \in s_i|_{\alpha e}$, which can be checked by examining whether e is in the suffix $s_{ij}|_\alpha$ in Line 6, we add $Pr(s_{ij})$ to $pr(s_i|_{\alpha e})$, and construct the new projected instance of s_{ij}, i.e. $s_{ij}|_\beta$, for the new projected probabilistic sequence $s_i|_\beta$ in Lines 8–9.

If β is found to be *$(\tau_{sup}, \tau_{prob})$-frequent* (Lines 13 and 14), we first output β in Line 15 and use Algorithm 2 to prune the candidate elements in the previous element table $T|_\alpha$, in order to obtain the current truncated element table $T|_\beta$. Finally, we check the patterns grown from β by running the recursion on $D|_\beta$ and $T|_\beta$ in Lines 17–18.

Algorithm 3 *SeqU-PrefixSpan*$(\alpha e, D|_\alpha, T|_\alpha)$

Input: current pattern αe, projected probabilistic database $D|_\alpha$, element table $T|_\alpha$

```
1: $vec_{\alpha e} \leftarrow \emptyset$
2: for each projected sequence $s_i|_\alpha \in D|_\alpha$ do
3:     $pr(s_i|_{\alpha e}) \leftarrow 0$
4:     for each instance $s_{ij}|_\alpha = <pos, Pr(s_{ij})> \in s_i|_\alpha$ do
5:         Find its corresponding sequence $s_{ij} \in D$
6:         if $e \in s_{ij}[pos+1, \ldots, len(s_{ij})]$ then
7:             $pr(s_i|_{\alpha e}) \leftarrow pr(s_i|_{\alpha e}) + Pr(s_{ij})$
8:             $c' \leftarrow \min_{c \geq pos+1}\{s_{ij}[c] = e\}$
9:             Append $(c', pr(s_{ij}))$ to $s_i|_{\alpha e}$
10:    if $pr(s_i|_{\alpha e}) > 0$ then
11:        Append $s_i|_{\alpha e}$ to $D|_{\alpha e}$
12:        Append $pr(s_i|_{\alpha e})$ to $vec_{\alpha e}$
13: $(tag, f_{\alpha e}) \leftarrow$ PMFCheck$(vec_{\alpha e})$
14: if $tag$ =TRUE then
15:     output $\alpha e$
16:     $T|_{\alpha e} \leftarrow$Prune$(T|_\alpha, D|_{\alpha e})$
17:     for each element $\ell \in T|_{\alpha e}$ do
18:         SeqU-PrefixSpan$(\alpha e \ell, D|_{\alpha e}, T|_{\alpha e})$
```

5 Element-Level U-PrefixSpan

In this section, we present our *ElemU-PrefixSpan* algorithm which mines p-FSPs from data conforming to the element-level uncertain model. Compared with *SeqU-PrefixSpan* discussed in the previous section, we need to consider additional issues arising from sequence projection.

An interesting observation is that the possible world space of s_i is exactly the sequence-level representation of s_i. Therefore, a naïve method to implement *ElemU-PrefixSpan* is to expand each element-level probabilistic sequence in database D into its sequence-level representation, and then solve the problem by *SeqU-PrefixSpan*. However, this approach is intractable due to the following fact:

"Each element-level probabilistic sequence of length ℓ has many sequence instances, the number of which is exponential to ℓ."

Instead of using the full-expansion approach mentioned above, we only expand the probabilistic sequence when it is necessary. For example, for pattern BA in the probabilistic sequence s_i in Figure 6, the expansion related to C is completely unnecessary, since whether C occurs in s_i or not has no influence on $Pr\{BA \sqsubseteq s_i\}$.

The differences between *ElemU-PrefixSpan* and *SeqU-PrefixSpan* mainly lie in two aspects: (1) sequence projection from s_i onto $s_i|_\alpha$, and (2) the computation of $Pr\{\alpha \sqsubseteq s_i\}$. We discuss them next.

Sequence Projection. Given an element-level probabilistic sequence s_i and a pattern α, we now explain how to obtain the projected probabilistic sequence $s_i|_\alpha$.

Definition 5: Event $e_{pos}(s_i, \alpha) = \{\alpha \sqsubseteq s_i[1, \ldots, pos] \wedge \alpha \not\sqsubseteq s_i[1, \ldots, pos-1]\}$.

In Definition 5, $s_i[1, \ldots, pos]$ is the *minimal* prefix of s_i that contains pattern α. Event $e_{pos}(s_i, \alpha)$ can be recursively constructed in the following way:

(1) **Base Case.** When pattern $\alpha = \emptyset$, we have $Pr(e_0(s_i, \alpha)) = 1$ and $Pr(e_{pos}(s_i, \alpha)) = 0$ for any $pos > 0$. This is because $\alpha \sqsubseteq \emptyset$, or equivalently, the *minimal* prefix $s_i[1, \ldots, pos]$ in Definition 5 should be $\emptyset$, which implies $pos = |s_i[1, \ldots, pos]| = |\emptyset| = 0$.

(2) **Recursive Rule.** When $\beta = \alpha e$,

$$e_{pos}(s_i, \beta) = \bigcup_{k<pos} \{ e_k(s_i, \alpha) \wedge s_i[pos] = e \wedge s_i[j] \neq e, \forall k < j < pos \}. \quad (7)$$

The above method works because $s_i[1, \ldots, pos]$ is the minimal prefix containing $\beta = \alpha e$, iff (1) $s_i[1, \ldots, k]$ is the minimal prefix containing α for some $k < pos$, (2) $s_i[pos] = e$, and (3) $s_i[j] \neq e$ for all positions j between k and pos.

The events $e_k(s_i, \alpha)$ on the R.H.S. of Equation (7) with different k values are disjoint due to the *minimality requirement* stated in Definition 5. Note that the events involved in the *UNION* operator of Equation (7) are subsets of the events $e_k(s_i, \alpha)$ (due to the *AND* operator), and thus they are also disjoint.

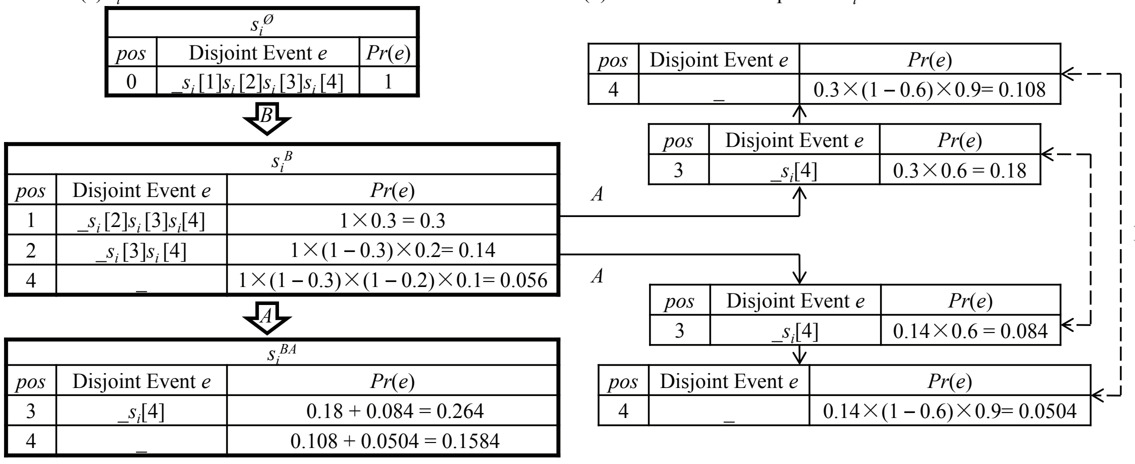
As a result, we can use the principle of additivity to

Probabilistic Element
$s_i[1] = \{(A, 0.7), (B, 0.3)\}$
$s_i[2] = \{(B, 0.2), (C, 0.8)\}$
$s_i[3] = \{(C, 0.4), (A, 0.6)\}$
$s_i[4] = \{(B, 0.1), (A, 0.9)\}$

(a) s_i

Seq. Instance	Prob.	Seq. Instance	Prob.	Seq. Instance	Prob.	Seq. Instance	Prob.
$pw_1(s_i) = ABCB$	0.0056	$pw_5(s_i) = ACCB$	0.0224	$pw_9(s_i) = BBCB$	0.0024	$pw_{13}(s_i) = BCCB$	0.0096
$pw_2(s_i) = ABCA$	0.0504	$pw_6(s_i) = ACCA$	0.2016	$pw_{10}(s_i) = BBCA$	0.0216	$pw_{14}(s_i) = BCCA$	0.0864
$pw_3(s_i) = ABAB$	0.0084	$pw_7(s_i) = ACAB$	0.0336	$pw_{11}(s_i) = BBAB$	0.0036	$pw_{15}(s_i) = BCAB$	0.0144
$pw_4(s_i) = ABAA$	0.0756	$pw_8(s_i) = ACAA$	0.3024	$pw_{12}(s_i) = BBAA$	0.0324	$pw_{16}(s_i) = BCAA$	0.1296

(b) Possible Instance Space of s_i



$s_i^{\emptyset}$

pos	Disjoint Event e	Pr(e)
0	$\_s_i[1]s_i[2]s_i[3]s_i[4]$	1

s_i^{B}

pos	Disjoint Event e	Pr(e)
1	$\_s_i[2]s_i[3]s_i[4]$	$1\times 0.3 = 0.3$
2	$\_s_i[3]s_i[4]$	$1\times(1-0.3)\times 0.2 = 0.14$
4	$\_$	$1\times(1-0.3)\times(1-0.2)\times 0.1 = 0.056$

s_i^{BA}

pos	Disjoint Event e	Pr(e)
3	$\_s_i[4]$	$0.18 + 0.084 = 0.264$
4	$\_$	$0.108 + 0.0504 = 0.1584$

pos	Disjoint Event e	Pr(e)
4	$\_$	$0.3\times(1-0.6)\times 0.9 = 0.108$

pos	Disjoint Event e	Pr(e)
3	$\_s_i[4]$	$0.3\times 0.6 = 0.18$

pos	Disjoint Event e	Pr(e)
3	$\_s_i[4]$	$0.14\times 0.6 = 0.084$

pos	Disjoint Event e	Pr(e)
4	$\_$	$0.14\times(1-0.6)\times 0.9 = 0.0504$

(c) Projection

Fig. 6. Illustration of *ElemU-PrefixSpan*

compute $Pr(e_{pos}(s_i, X))$ according to Equation (7):

$$\begin{aligned} & Pr(e_{pos}(s_i, \alpha e)) \\ = & \sum_{k<pos} [\ Pr(e_k(s_i, \alpha)) \times Pr\{s_i[pos] = e\} \\ & \times \prod_{k<j<pos} (1 - Pr\{s_i[j] = e\})\ ]. \end{aligned} \tag{8}$$

Equation (8) is a recursive formula where the computation of $Pr(e_{pos}(s_i, \alpha e))$ requires the values of $Pr(e_k(s_i, \alpha))$ for all $k < pos$.

Recall that in the prefix-projection method of *PrefixSpan*, the projected sequence $s|_\alpha$ of a deterministic sequence s is obtained by removing from s its minimal prefix containing α. Therefore, the projected sequence $s_i|_\alpha$ of an element-level probabilistic sequence s_i can be represented by a set of disjoint events $e_k(s_i, \alpha), 0 < k \le len(s_i)$, where $len(s_i)$ is the number of probabilistic elements in s_i. The first (top) table in Figure 6(c) gives the *event representation* of $s_i|_\emptyset$ for the probabilistic sequence s_i shown in Figure 6(a).

Next, let us consider the case when α grows from $\emptyset$ to B. For ease of presentation, we use $s_i|_{e_{pos}(s_i,\alpha)}$ to denote the suffix of s_i given event $e_{pos}(s_i, \alpha)$. Since $s_i|_{e_0(s_i,\emptyset)} = s_i[1]s_i[2]s_i[3]s_i[4]$, and B can occur in any of $s_i[1]$, $s_i[2]$ and $s_i[4]$, we can derive from $e_0(s_i, \emptyset)$ altogether three disjoint events that correspond to B occurring in $s_i|_{e_0(s_i,\emptyset)}$, as shown in the second (middle) table in Figure 6(c):

- $e_1(s_i, B) = \{s_i[1] = B\}$. In this case, $Pr(e_1(s_i, B)) = Pr(e_0(s_i, \emptyset)) \times Pr\{s_i[1] = B\} = 0.3$.
- $e_2(s_i, B) = \{s_i[1] \ne B \wedge s_i[2] = B\}$. In this case, $Pr(e_2(s_i, B)) = Pr(e_0(s_i, \emptyset)) \times (1 - Pr\{s_i[1] = B\}) \times Pr\{s_i[2] = B\} = 0.14$.
- $e_4(s_i, B) = \{s_i[1] \ne B \wedge s_i[2] \ne B \wedge s_i[4] = B\}$. In this case, $Pr(e_4(s_i, B)) = Pr(e_0(s_i, \emptyset)) \times (1 - Pr\{s_i[1] = B\}) \times (1 - Pr\{s_i[2] = B\}) \times Pr\{s_i[4] = B\} = 0.0056$.

For the case when α grows from B to BA, we focus on the event $e_2(s_i, B)$ of $s_i|_B$. Since $s_i|_{e_2(s_i,B)} = s_i[3]s_i[4]$, and A may occur in either $s_i[3]$ or $s_i[4]$, we can derive two sub-events from $e_2(s_i, B)$ as shown in Figure 6(c). For example, the probability of the sub-event at the bottom on the right of Figure 6(c) is computed as $Pr(e_2(s_i, B)) \times (1 - Pr\{s_i[3] = A\}) \times Pr\{s_i[4] = A\} = 0.0504$.

Note that we do not obtain any sequence containing pattern BA from $e_4(s_i, B)$. After all the sub-events are obtained, we merge those with the same pos value into $e_{pos}(s_i, BA)$, where $Pr(e_{pos}(s_i, BA))$ is computed as the summation of the probabilities of the sub-events (see the bottom table in Figure 6(c)), which is based on Equation (8).

Algorithm 4 shows our algorithm which constructs $D|_\beta$ $(\beta = \alpha e)$ from the old projected database $D|_\alpha$. For each projected probabilistic sequence $s|_\alpha$, we project it onto a new projected sequence $s|_\beta \in D|_\beta$ in Lines 2–19. In our algorithm, each projected probabilistic sequence represents a set of events (recall Figure 6(c)), and each event $r = e_{pos}(s_i, \alpha)$ represents a pair $<pos_r, Pr(r)>$, where $pos_r = pos$ and $Pr(r) = Pr(e_{pos}(s_i, \alpha))$.

To obtain $\beta = \alpha e$, we need to find element e from the suffix of s starting from $pos_r + 1$ (Line 4), i.e. $s|_r$. We also attach a variable $accum_r$ to each event r to record the value of the product term on the R.H.S. of Equation (8) and the term is initialized to 1.

To construct $s|_\beta$ from $s|_\alpha$, we check all the events $r = <pos_r, Pr(r)>$ of $s|_\alpha$, and in each iteration, we pick

Algorithm 4 *Project*$(D|_\alpha, e)$

Input: projected probabilistic database $D|_\alpha$, element e
Output: $D|_{\alpha e}$

```
1: for each projected sequence $s|_\alpha \in D|_\alpha$ do
2:   Find its corresponding sequence $s \in D$
3:   for each event $r = (pos_r, pr_r) \in s|_\alpha$ do
4:     $pivot_r \leftarrow pos_r + 1$
5:     $accum_r \leftarrow 1$
6:   $s|_{\alpha e} \leftarrow \emptyset$
7:   while $\exists r, pivot_r < len(s)$ do
8:     $r' \leftarrow \arg\min_r pivot_r$
9:     if $Pr\{s[pivot_{r'}] = e\} > 0$ then
10:      $\{\exists$ probabilistic element $(e, p_e) \in s[pivot_{r'}]\}$
11:      $\Delta \leftarrow pr_{r'} \times accum_{r'} \times p_e$
12:      $accum_{r'} \leftarrow accum_{r'} \times (1 - p_e)$
13:      $(pos_{last}, pr_{last}) \leftarrow$the last element in $s|_{\alpha e}$
14:      if $pos_{last} = pivot_{r'}$ then
15:        $pr_{last} \leftarrow pr_{last} + \Delta$
16:      else
17:        Append $(pivot_{r'}, \Delta)$ to $s|_{\alpha e}$
18:    $pivot_{r'} \leftarrow pivot_{r'} + 1$
19:  Append $s|_{\alpha e}$ to $D|_{\alpha e}$
20: return $D|_{\alpha e}$
```

the event with the minimum position value $pivot_r$ to be scanned next (Line 8) and denote this event of r'. If the probabilistic element in the current position $pivot_{r'}$ can take value e with probability p_e (Line 9), then we can compute the probability of the sub-event derived from r' as Δ using Equation (8) (Line 11), and update the product value $accum_r$ in Line 12 to reflect the event that $\{s[pivot_{r'}] \neq e\}$, since $pos > pivot_{r'}$ for later sub-events.

Since we choose the event with the minimum position value in each iteration, the sub-events are constructed with the **non-decreasing** values of pos. According to Equation (8), we can sum the probabilities of the sub-events with the same new value of pos. Therefore, if the newly constructed sub-event has the same value of pos as the last sub-event already constructed, we simply add its probability Δ to that of the last sub-event (Lines 14–15). Otherwise, we create a new event for $s|_{\alpha e}$ with the new value of pos, and the probability is initialized to Δ (Lines 16–17).

When $s|_\alpha$ has k events, and each event e_i has suffix of length ℓ_i, then it takes $O(k \times \sum_i \ell_i)$ time to construct $s|_\beta$ from $s|_\alpha$. This is because, in each iteration of the while loop, Line 8 takes $O(k)$ time, and there are $O(\sum_i \ell_i)$ iterations (see Lines 7 and 18).

Recall that each element-level projected sequence is represented by a set of events, and each value of pos corresponds to one event. Thus, we have the following interesting observation:

"Each element-level projected probabilistic sequence $s|_\alpha$ of length ℓ can have no more than ℓ events."

The correctness of this statement is established by the fact that there are at most ℓ values for pos. This result

Algorithm 5 *ElemProb*(s, pos, e)

Input: probabilistic sequence s, position pos, element e
Output: $Pr\{\, e \in s[\,pos + 1, len(s)\,]\,\}$

```
1: $accum \leftarrow 1$
2: for each $k \in [\,pos + 1, len(s)\,]$ do
3:   if $(e, Pr\{s[k] = e\}) \in s[k]$ then
4:     $accum \leftarrow accum \cdot (1 - Pr\{s[k] = e\})$
5: $Pr\{\, e \in s[\,pos + 1, len(s)\,]\,\} \leftarrow 1 - accum$
6: return $Pr\{\, e \in s[\,pos + 1, len(s)\,]\,\}$
```

can be utilized to solve the problem arising from the full-expansion approach, in which each $s|_\alpha$ is expanded to many sequence instances and the number of such instances number is exponential to ℓ.

Computation of $Pr\{\alpha \sqsubseteq s_i\}$. Consider pattern $\beta = \alpha e$. Suppose that the projected probabilistic sequence $s_i|_\alpha$ has k events $e_{pos}(s_i, \alpha)$, $pos = i_1, i_2, \ldots, i_k$. Then, for each event $e_{pos}(s_i, \alpha)$ which implies $\alpha \sqsubseteq s_i$, it follows that $\beta \not\sqsubseteq s_i$ if and only if e does not occur in any of the elements in the suffix $s_i[pos{+}1, \ldots, len(s_i)]$ (i.e. $s_i|_{e_{pos}(s_i,\alpha)}$). Thus, it follows that

$$\begin{aligned} & Pr\{\beta \not\sqsubseteq s_i\} \\ = & \sum_{pos} Pr\{\beta \not\sqsubseteq s_i | e_{pos}(s_i, \alpha)\} \times Pr(e_{pos}(s_i, \alpha)) \\ = & \sum_{pos} Pr\{s_i[j] \neq e, \forall j > pos\} \times Pr(e_{pos}(s_i, \alpha)) \\ = & \sum_{pos} \left(Pr(e_{pos}(s_i, \alpha)) \times \prod_{i > pos} (1 - Pr\{s_i[pos] = e\}) \right) \end{aligned} \quad (9)$$

So we now have

$$\begin{aligned} Pr\{\beta \sqsubseteq s_i\} &= 1 - Pr\{\beta \not\sqsubseteq s_i\} \\ &= \left(\sum_{pos} Pr(e_{pos}(s_i, \alpha)) \right) - Pr\{\beta \not\sqsubseteq s_i\} \\ &= \sum_{pos} \Bigg[Pr(e_{pos}(s_i, \alpha)) \times \\ &\qquad \left(1 - \prod_{i > pos} (1 - Pr\{s_i[pos] = e\}) \right) \Bigg] \end{aligned} \quad (10)$$

Algorithm 5 shows how we compute the factor in the last line of Equation (10). Algorithm 6 shows our *ElemU-PrefixSpan* algorithm, where Line 6 computes Equation (10) as $accum$ using Algorithm 5. After obtaining $Pr\{\beta \sqsubseteq s_i\}$ for all $s_i|_\beta \in D|_\beta$, we check the *$(\tau_{sup}, \tau_{prob})$-frequentness* of β and prune the element table similarly to Algorithm 3.

6 Fast Validating Method

In this section, we present a fast validation method that further speeds up the *U-PrefixSpan* algorithm. The method involves two approximation techniques that check the probabilistic frequentness of patterns, reducing the time

Algorithm 6 *ElemU-PrefixSpan*(αe, $D|_\alpha$, $T|_\alpha$)

Input: pattern αe, projected probabilistic database $D|_\alpha$, element table $T|_\alpha$

```
1: $vec_\alpha \leftarrow \emptyset$
2: for each projected sequence $s|_\alpha \in D|_\alpha$ do
3:     Find its corresponding sequence $s \in D$
4:     $accum \leftarrow 0$
5:     for each $(pos, pr) \in s|_\alpha$ do
6:         $accum \leftarrow accum + pr \times ElemProb(s, pos, e)$
7:     Append $accum$ to $vec_{\alpha e}$
8: $(tag, f_{\alpha e}) \leftarrow PMFCheck(vec_{\alpha e})$
9: if $tag = TRUE$ then
10:    output $\alpha e$
11:    $D|_{\alpha e} \leftarrow Project(D|_\alpha, e)$
12:    $T|_{\alpha e} \leftarrow Prune(T|_\alpha, D|_{\alpha e})$
13:    for each element $\ell \in T|_{\alpha e}$ do
14:        $ElemU\text{-}PrefixSpan(\alpha e\ell, D|_{\alpha e}, T|_{\alpha e})$
15: Free $D|_{\alpha e}$ and $T|_{\alpha e}$ from memory
```

complexity from $O(n \log^2 n)$ to $O(n)$. The underlying idea of our method is to approximate the probabilistic frequentness of patterns by applying some probability model (e.g. a Poisson or Normal distribution), so that p-FSPs can be verified quickly.

Given an uncertain database of size n, each sequential pattern α is associated with n probabilities $Pr\{\alpha \sqsubseteq s_i\}$ $(i = 1, \dots, n)$, where each probability $Pr\{\alpha \sqsubseteq s_i\}$ conforms to an independent Bernoulli distribution representing the existence of pattern α in s_i. Since the sequences s_i $(i = 1, \dots, n)$ are independent of each other, the events $\{\alpha \sqsubseteq s_i\}$ represent n Poisson trials.Therefore, the random variable $sup(\alpha)$ follows a Poisson-binomial distribution. In both the sequence-level and element-level models, the verification of probabilistic frequentness of α is given by

$$Pr\{sup(\alpha) \geq \tau_{sup}\} = 1 - Pr\{sup(\alpha) \leq \tau_{sup} - 1\}, \quad (11)$$

where $Pr\{sup(\alpha) \leq \tau_{sup} - 1\}$ is a Poisson-binomial cumulative distribution of random variable $sup(\alpha)$. The Poisson binomial distribution can be approximated by the Poisson distribution and the performance has been validated in [27].

Let us denote the Possion distribution by $f(k, \lambda) = \frac{\lambda^k e^{-\lambda}}{k!}$, and denote its cumulative distribution by $F(k, \lambda)$. We propose an approximation algorithm (i.e. *PA*) based on Possion cumulative distribution $F(\mu, \tau_{sup} - 1)$. This algorithm checks α in the projection database by

$$Pr\{sup(\alpha) \geq \tau_{sup}\} \approx 1 - F(\mu, \tau_{sup} - 1) \geq \tau_{prob}, \quad (12)$$

where $F(\mu, \tau_{sup} - 1)$ monotonically decreases w.r.t. μ, as shown in [27] and μ is the expected support of α given by

$$\mu = \sum_{i=1}^{n_\alpha} Pr\{\alpha \sqsubseteq s_i\}, \quad (13)$$

with n_α being the size of $D|_\alpha$.

Based on the property of $F(\mu, \tau_{sup} - 1)$ and Equation 12, the value of α estimated by *PA* monotonically increases w.r.t μ. We compute the minimum expected support threshold μ_m by

$$1 - F(\mu_m, \tau_{sup} - 1) = \tau_{prob}. \quad (14)$$

The underlying idea of Equation 14 is to use numerical methods and grows the patterns whose expected support μ is greater than μ_m.

The *PA* method utilizes the expected support to approximate the probabilistic frequentness of patterns. However, the *PA* method only works well when the expected support of α (i.e. $expSup(\alpha)$) is very small, as stated in [31].

As a result, we propose another method, Normal approximation (i.e. *NA*), to check the probabilistic frequentness of patterns based on the *Central Limiting Theorem.* The *NA* method is more robust, since it verifies the probabilistic frequentness of pattern using both the expected support and the standard variance. The computation of standard variance δ of α in its projected database is given by

$$\delta = \sqrt{\sum_{i=1}^{n_\alpha} Pr\{\alpha \sqsubseteq s_i\}(1 - Pr\{\alpha \sqsubseteq s_i\})}, \quad (15)$$

and, therefore the *NA* approximation of the probabilistic frequentness of α is given by

$$Pr\{sup(\alpha) \geq \tau_{sup}\} \approx 1 - G(\frac{\tau_{sup} - \frac{1}{2} - \mu}{\delta}) \quad (16)$$

where $G(t) = \int_{-\infty}^{t} e^{-\frac{x^2}{2}} dx$ and $\frac{\tau_{sup} - \frac{1}{2} - \mu}{\delta}$ is the normalization of the parameter for the probability distribution $G(t)$. The *NA* method has a good approximate ratio whose upper error bound [30] is given by

$$sup_{\tau_{sup}} \left\{ |Pr\{sup(\alpha) \leq \tau_{sup} - 1\} - G(\frac{\tau_{sup} - \frac{1}{2} - \mu}{\delta}))| \right\} \leq c\delta^{-2}, \quad (17)$$

where c is a constant and its proof can be found in [30]. The approximate ratio of *NA* method is tighter for larger uncertain databases.

The formula of the *NA* method is monotonically decreasing as t increases, since we have the following derivation

$$\begin{aligned} \frac{\partial}{\partial t}(1 - G(t)) &= -\frac{\partial}{\partial t} G(t) \\ &= -\frac{\partial}{\partial t} \int_{-\infty}^{t} e^{-\frac{x^2}{2}} dx \\ &= -e^{-\frac{t^2}{2}} \\ &\leq 0, \end{aligned}$$

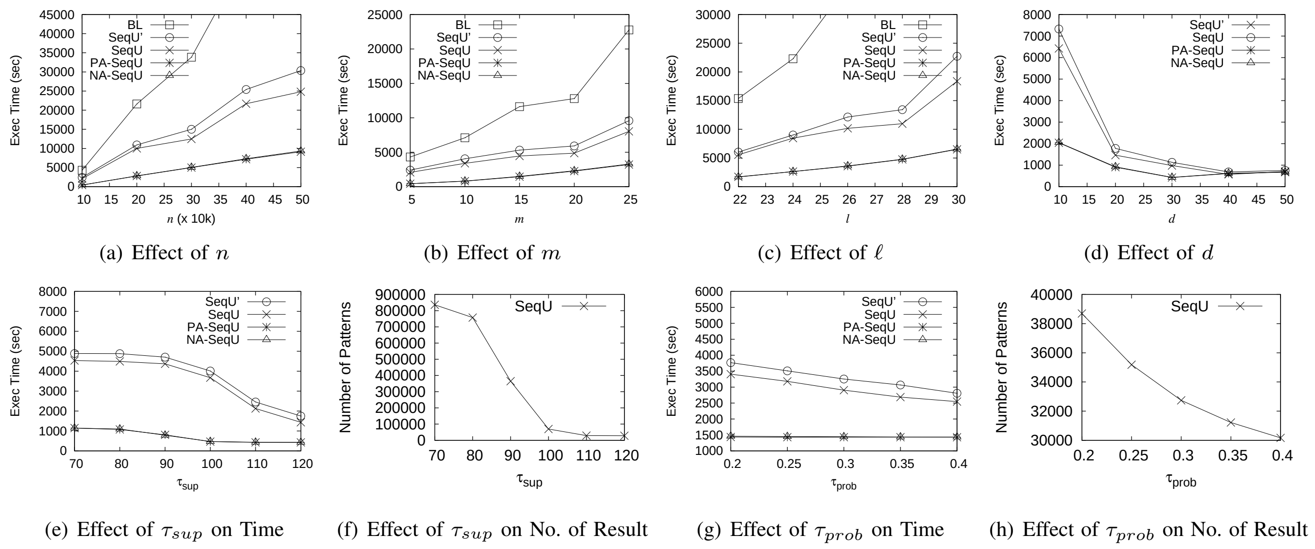
where t is the parameter of Normal distribution (i.e. $t = \frac{\tau_{sup} - \frac{1}{2} - \mu}{\delta}$). We compute the maximum t (i.e. t_m) as the verification threshold for the p-FSP, and the formula is given by

$$1 - G(t_m) = \tau_{prob}. \quad (18)$$

We compute t_m by numerical methods. We also compute μ and δ by scanning the projected database $D|_\alpha$ and grow the pattern α when $t = \frac{\tau_{sup} - \frac{1}{2} - \mu}{\delta} \leq t_m$.

TABLE 1
APPROXIMATE PRECISION ON SEQUENCE-LEVEL UNCERTAIN MODEL

n	*PA*	*NA*	m	*PA*	*NA*	l	*PA*	*NA*	d	*PA*	*NA*	τ_{sup}	*PA*	*NA*	τ_{prob}	*PA*	*NA*
10	0.86	0.95	5	0.98	0.99	22	0.99	0.99	10	1	1	70	0.99	0.99	0.2	0.67	0.99
20	1	1	10	0.99	0.99	24	1	1	20	1	1	80	0.99	0.998	0.25	0.78	0.99
30	1	1	15	0.99	0.99	26	1	1	30	0.989	0.999	90	0.98	0.99	0.3	0.83	0.99
40	1	1	20	1	0.99	28	1	1	40	1	1	100	0.991	0.998	0.35	0.92	0.99
50	1	1	25	1	1	30	1	1	50	1	1	110	0.98	0.99	0.4	0.95	0.99



(a) Effect of n (b) Effect of m (c) Effect of ℓ (d) Effect of d

(e) Effect of τ_{sup} on Time (f) Effect of τ_{sup} on No. of Result (g) Effect of τ_{prob} on Time (h) Effect of τ_{prob} on No. of Result

Fig. 7. Scalability Results on Sequence-Level Uncertain Model

7 EXPERIMENTS

In this section, we study the performance of our two *U-PrefixSpan* algorithms using both real and synthetic datasets. Specifically, we test the performance of *U-PrefixSpan* algorithms and their approximation algorithms , using large synthetic datasets in Sections 7.1 and 7.2. We define *recall* and *precision* to measure the accuracy of the approximation methods as

$$precision = \frac{|FSP_{app} \bigcap FSP|}{|FSP_{app}|}, \tag{19}$$

$$recall = \frac{|FSP_{app} \bigcap FSP|}{|FSP|}, \tag{20}$$

where FSP is the set of patterns obtained by *U-PrefixSpan* algorithms, the patterns in FSP are taken as the ground truth, and FSP_{app} is the set of patterns obtained by the approximation methods. For brevity, we only report the approximate precision in this paper, since the approximate recall reaches 1 in all cases. In Section 7.3, we compare *ElemU-PrefixSpan* with the full expansion approach for mining data that conform to the element-level uncertain model, where the results show that *ElemU-PrefixSpan* effectively avoids the problem of "possible world explosion". Finally, in Section 7.4, we successfully apply *ElemU-PrefixSpan* in an RFID application for trajectory pattern mining, and the result validates the performance of the approximation algorithms.

All the experiments were run on a computer with Intel(R) Core(TM) i5 CPU and 4GB memory. The algorithms were implemented in C++, and run in Eclipse on Windows 7 Enterprise.

7.1 SeqU-PrefixSpan Experimental Results

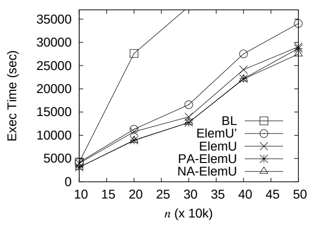
Synthetic Data Generation. To test the performance of *SeqU-PrefixSpan*, we implement a data generator to generate datasets that conform to the sequence-level uncertain model. Given the configuration (n, m, ℓ, d), our generator generates n probabilistic sequences. For each probabilistic sequence, the number of sequence instances is randomly chosen from the range $[1, m]$.The length of a sequence instance is randomly chosen from the range $[1, \ell]$, and each element in the sequence instance is randomly picked from an element table with d elements.

Experimental Setting. In addition to the four dataset configuration parameters n, m, ℓ and d, we also have two threshold parameters: the support threshold τ_{sup} and the probability threshold τ_{prob}.

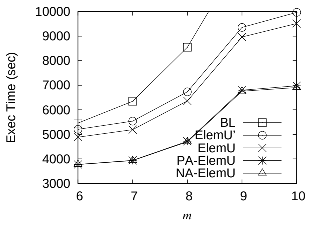
To study the effectiveness of our three pruning rules (*CntPrune*, *MarkovPrune* and *ExpPrune*) and early validating method (cf. Theorem 1), we also carry out experiments on the algorithm version without them. This serves as the *baseline*. From now on, we abbreviate our *SeqU-PrefixSpan* algorithm to *SeqU*, our *ElemU-PrefixSpan* algorithm to *ElemU*, and their baseline algorithm version without the pruning and validating methods for *BL*. We also name the algorithm version that uses only the pruning methods by appending an apostrophe to the original algorithm names, e.g. *SeqU* becomes *SeqU'*. The *SeqU-PrefixSpan* algorithms

TABLE 2
APPROXIMATION RESULTS ON ELEMENT-LEVEL UNCERTAIN MODEL

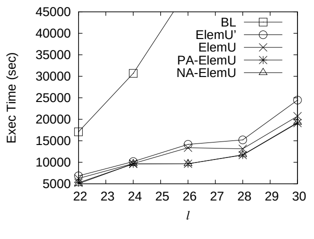
n	*PA*	*NA*	m	*PA*	*NA*	l	*PA*	*NA*	d	*PA*	*NA*	τ_{sup}	*PA*	*NA*	τ_{prob}	*PA*	*NA*
10	0.86	0.95	6	0.80	0.927	22	0.942	0.973	10	1	1	15	1	1	0.2	0.506	0.889
20	1	1	7	0.74	0.904	24	1	1	20	1	1	18	1	1	0.25	0.536	0.881
30	1	1	8	0.71	0.881	26	1	1	30	0.86	0.95	21	1	1	0.3	0.659	0.894
40	1	1	9	0.685	0.85	28	1	1	40	1	1	24	1	1	0.35	0.809	0.92
50	1	1	10	0.663	0.849	30	1	1	50	1	1	27	1	1	0.4	0.865	0.962



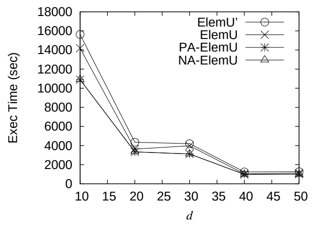
(a) Effect of n



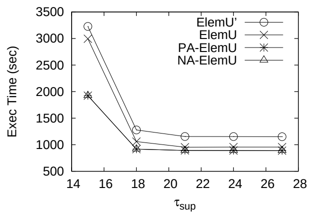
(b) Effect of m



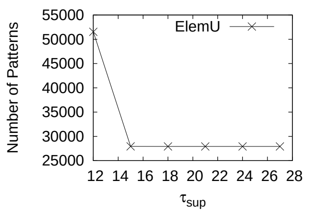
(c) Effect of ℓ



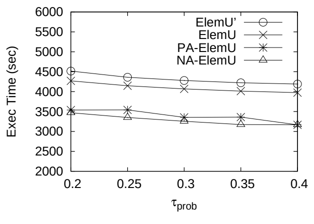
(d) Effect of d



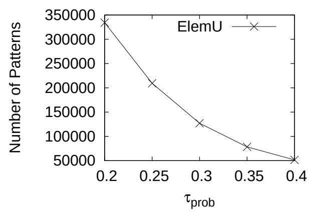
(e) Effect of τ_{sup} on Time



(f) Effect of τ_{sup} on No. of Result



(g) Effect of τ_{prob} on Time



(h) Effect of τ_{prob} on No. of Result

Fig. 8. Scalability Results on Element-Level Uncertain Model

based on Poisson approximation and Normal approximation are called *PA-SeqU* and *NA-SeqU*, respectively.

Effect of n, m, ℓ and d on Execution Time. The experimental results are presented in Figures 7(a) to 7(d). From these results, we summarize some interesting observations as follows:

- In all the experiments, *BL* is around 2 to 3 times slower than *SeqU'*, which verifies the effectiveness of the pruning methods. *SeqU'* is around 10% to 20% slower than *SeqU*, which verifies the effectiveness of the validating method.
- The running time of all the algorithms increase as n, m and ℓ increase. In particular, the running time of all the algorithms increases almost linearly with n.
- The running time of *SeqU* and *SeqU'* decreases as d increases. This can be explained as follows. When the data size is fixed, a larger pool of elements implies that the length of the patterns found by *SeqU-PrefixSpan* tends to be smaller, which further shows that *SeqU-PrefixSpan* does not have to recurse to deep levels.
- *PA-SeqU* and *NA-SeqU* are more efficient than *Seq-U*. They increase linearly as n, m and l increase, as shown in Figures 7(a) to 7(c). The performance of *PA-SeqU* and *NA-SeqU* is far better than that of *SeqU* when the uncertainty of data is high, as shown in Figure 7(b). We also find that their precision is very high, almost reaching 1 in all the configurations in Table 1.

Effect of τ_{sup} and τ_{prob} on Execution Time and Number of Results. The experimental results are presented in Figures 7(e) to 7(h). From these figures, we observe that both *PA-SeqU* and *NA-SeqU* algorithms have good approximate precision on τ_{sup} varies. The *NA-SeqU* algorithm has good approximate precision on τ_{tau} varies while the *PA-SeqU* does not have high precision, as shown in Table 1. The *NA-SeqU* algorithm is more robust than *PA-SeqU* on estimating Poisson-binomial cumulative distribution of random variable $sup(\alpha)$ of pattern α.

7.2 ElemU-PrefixSpan Experimental Results

Synthetic Data Generation. Similarly to the study of *SeqU-PrefixSpan*, we generate datasets that conform to the element-level uncertain model to test the scalability of *ElemU-PrefixSpan*. Using the configuration (n, m, ℓ, d), our generator generates n probabilistic sequences.In each probabilistic sequence, 20% of the elements are sampled to be uncertain. We generate a value w_{ij} following uniform distribution in the range $(0, 1)$ for each instance j of a probabilistic element i, then normalize the value as its probability.

Similarly to the sequence-level case presented in Section 7.1, we have altogether six parameters of n, m, ℓ, d, τ_{sup} and τ_{prob}. For each dataset configuration, we generate five datasets and the results are averaged on the five runs before they are reported. The experimental results are in Figures 8(a) to 8(g).

The trends observed from these results are similar to those observed from the scalability test of *SeqU-PrefixSpan* in Section 7.1, and thus a similar analysis can also be applied. The precision of *PA-ElemU* and *NA-ElemU* can be found in Table 2.

7.3 ElemU-PrefixSpan v.s. Full Expansion

Recall from Section 5 that a naïve method to mine p-FSPs from data that conform to the element-level uncertain model, is to first expand each element-level probabilistic sequence into all its possible sequence instances, and then mine p-FSPs from the expanded sequences using *SeqU-PrefixSpan*.

In this subsection, we empirically compare this naïve method with our *ElemU-PrefixSpan* algorithm. We use the same data generator as the one described in Section 7.2 to generate experimental data, with the default setting $(n, m, \ell, d) = (10\text{k}, 5, 20, 30)$. Figures 9(a) to 9(d) show the running time of both algorithms with mining parameters $\tau_{sup} = 16$ and $\tau_{prob} = 0.7$, where one data parameter is varied and the other three are fixed to the default values. Note that for the naïve method, we do not include the time required for sequence expansion (i.e. We only count the mining time of *SeqU-PrefixSpan*).

In Figures 9(a), 9(c) and 9(d), *ElemU-PrefixSpan* is around 20 to 50 times faster than the naïve method, and this performance ratio is relatively insensitive to parameters n, ℓ and d. On the other hand, as shown in Figure 9(b), the performance ratio increases sharply as m increases: 2.6 times when $m = 2$, 22 times when $m = 5$ and 119 times when $m = 6$. This trend is intuitive, since m controls the number of element instances in a probabilistic element, which has a big influence on the number of expanded sequence instances. All results show that *ElemU-PrefixSpan* effectively avoids the problem of "possible world explosion" associated with the naïve method.

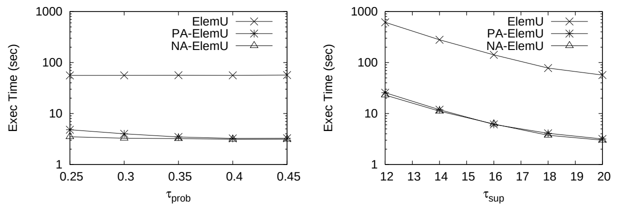
7.4 A Case Study of RFID Trajectory Mining

In this subsection, we evaluate the effectiveness of *ElemU-Prefix-Span* by using the real RFID datasets obtained from the Lahar project [32]. The data were collected in an RFID deployment with nearly 150 RFID antennae spread throughout the hallways of all six floors of a building. These antennae detect RFID tags that pass by, and log the sightings along with their timestamp in a database. In our experiment, we use a database of 213 probabilistic sequences with an average of 10 instances.

We test the performance of our approximation methods on τ_{sup} and τ_{prob}. We find that the approximation methods *NA-ElemU* and *PA-ElemU* are an order of magnitude faster than *ElemU* as shown in Figures 10(a) and 10(b). This result shows that the approximation methods *NA-ElemU* and *PA-ElemU* perform better for more uncertain datasets. The underlying reason is that the size of possible projections of some pattern α becomes larger as the uncertainty of the data (i.e. m) grows. Compared with the approximation methods, the *ElemU* algorithm needs more time to validate the patterns, as shown in the time complexity analysis. We also conclude that *NA-ElemU* performs better than *PA-ElemU*, since *NA-ElemU* is more robust in probabilistic frequentness estimation, as shown in Table 3.

TABLE 3
APPROXIMATION RESULTS ON REAL DATASET

τ_{sup}	*PA*	*NA*	τ_{prob}	*PA*	*NA*
20	0.924138	1	0.45	0.924138	0.943662
18	0.854911	0.994805	0.4	0.85623	0.924138
16	0.995868	0.995868	0.35	0.748603	0.884488
14	0.911961	0.99867	0.3	0.598214	0.848101
12	0.870491	0.989465	0.25	0.467714	0.752809



(a) Effect of τ_{prob} on Time (b) Effect of τ_{sup} on Time

Fig. 10. Scalability Results on Real Dataset

Figure 11 shows a sample result trajectory pattern with support threshold equal to 3, whose probability of being frequent is 91.4%. The blue lines correspond to the connectivity graph, the red rectangles correspond to the RFID antennae, and the green points correspond to the locations in the trajectory pattern, the orders of which are marked by the numbers near them. We also compute the expected support of this sample trajectory pattern, which is 2.95. Thus, this pattern cannot be found if expected support is adopted to measure pattern frequentness.

8 CONCLUSIONS

In this paper, we study the problem of mining probabilistically frequent sequential patterns (*p-FSPs*) in uncertain databases. Our study is founded on two uncertain sequence data models that are fundamental to many real-life applications. We propose two new *U-PrefixSpan* algorithms to mine p-FSPs from data that conform to our sequence-level and element-level uncertain sequence models. We also design three pruning rules and one early validating method to speed up pattern frequentness checking. These rules are able to improve the mining efficiency. To further enhance the algorithmic efficiency, we devise two approximation methods to verify the probabilistic frequentness of the patterns based on Poisson and Normal distributions. The experiments conducted on synthetic and real datasets show that our two *U-PrefixSpan* algorithms effectively avoid the problem of "possible world explosion" and the approximation methods *PA* and *NA* are very efficient and accurate.

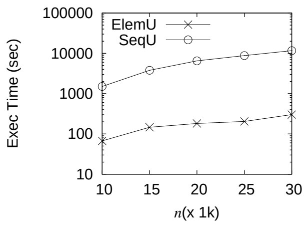
ACKNOWLEDGMENT

We would like to express our thanks to the editor and the reviewers for their careful revisions and insightful suggestions.

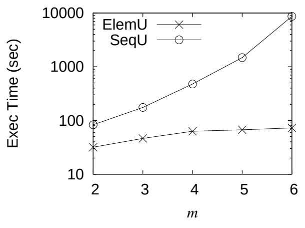
REFERENCES

[1] M. Muzammal and R. Raman. "Mining Sequential Patterns from Probabilistic Databases". In *PAKDD*, 2011.

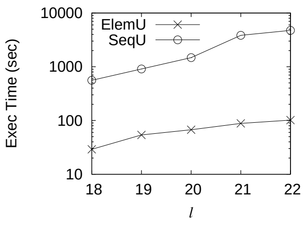
[2] F. Giannotti, M. Nanni, F. Pinelli and D. Pedreschi. "Trajectory Pattern Mining". In *SIGKDD*, 2007.



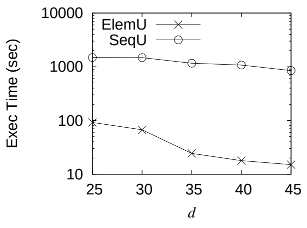
(a) Effect of n



(b) Effect of m



(c) Effect of ℓ



(d) Effect of d

Fig. 9. ElemU-PrefixSpan v.s. Full Expansion on Element-Level Uncertain Model

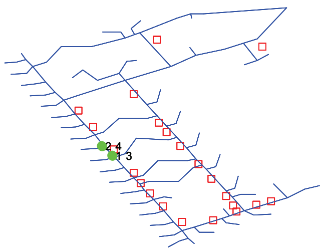


Fig. 11. Sample Result of a RFID Trajectory Pattern – Blue Lines Corresponding to the Connectivity Graph. Red Rectangles Corresponding to the RFID Antennae. Green Points Corresponding to the Locations

[3] D. Tanasa, J. A. López and B. Trousse. “Extracting Sequential Patterns for Gene Regulatory Expressions Profiles”. In *Knowledge Exploration in Life Science Informatics*, 2004.

[4] J. Pei, J. Han, B. Mortazavi-Asl, H. Pinto, Q. Chen, U. Dayal and M. C. Hsu. “PrefixSpan: Mining Sequential Patterns Efficiently by Prefix-Projected Pattern Growth”. In *ICDE*, 2001.

[5] R. Agrawal and R. Srikant. “Mining Sequential Patterns”. In *ICDE*, 1995.

[6] M. J. Zaki. “SPADE: An Efficient Algorithm for Mining Frequent Sequences”. In *Machine Learning*, 2001.

[7] J. Han, J. Pei, B. Mortazavi-Asl, Q. Chen, U. Dayal and M. C. Hsu. “FreeSpan: Frequent Pattern-Projected Sequential Pattern Mining”. In *SIGKDD*, 2000.

[8] R. Srikant and R. Agrawal. “Mining Sequential Patterns: Generalizations and Performance Improvements”. In *EDBT*, 1996.

[9] Z. Zhao, D. Yan and W. Ng. ”Mining probabilistically frequent sequential patterns in uncertain databases”. In *EDBT*, 2012

[10] C. Gao and J. Wang. ”Direct mining of discriminative patterns for classifying uncertain data”. In *SIGKDD*, 2010

[11] N. Pelekis, I. Kopanakis, E. E. Kotsifakos, E. Frentzos and Y. Theodoridis. “Clustering Uncertain Trajectories”. In *Knowledge and Information Systems*, 2010.

[12] H. Chen, W. S. Ku, H. Wang and M. T. Sun. “Leveraging Spatio-Temporal Redundancy for RFID Data Cleansing”. In *SIGMOD*, 2010.

[13] A. Deshpande, C. Guestrin, S. R. Madden, J. M. Hellerstein and W. Hong. “Model-Driven Data Acquisition in Sensor Networks”. In *VLDB*, 2004.

[14] L. Sun, R. Cheng, D. W. Cheung and J. Cheng. “Mining Uncertain Data with Probabilistic Guarantees”. In *SIGKDD*, 2010.

[15] C. C. Aggarwal, Y. Li, J. Wang and J. Wang. “Frequent Pattern Mining with Uncertain Data”. In *SIGKDD*, 2009.

[16] Q. Zhang, F. Li and K. Yi “Finding Frequent Items in Probabilistic Data”. In *SIGMOD*, 2008.

[17] T. Bernecker, H. P. Kriegel, M. Renz, F. Verhein and A. Zuefle. “Probabilistic Frequent Itemset Mining in Uncertain Databases”. In *SIGKDD*, 2009.

[18] C. K. Chui, B. Kao and E. Hung “Mining Frequent Itemsets from Uncertain Data”. In *PAKDD*, 2007.

[19] C. C. Aggarwal and P. S. Yu. “A Survey of Uncertain Data Algorithms and Applications”. In *TKDE*, 2008.

[20] J. Yang, W. Wang, P. S. Yu, and J. Han. “Mining Long Sequential Patterns in a Noisy Environment”. In *SIGMOD*, 2002.

[21] P. Agrawal, O. Benjelloun, A. D. Sarma, C. Hayworth, S. Nabar, T. Sugihara and J. Widom. “Trio: A System for Data, Uncertainty, and Lineage”. In *VLDB*, 2006.

[22] X. Lian and L. Chen. “Set Similarity Join on Probabilistic Data”. In *VLDB*, 2010.

[23] J. Jestes, F. Li, Z. Yan and K. Yi. “Probabilistic String Similarity Joins”. In *SIGMOD*, 2010.

[24] Y. Tong, L. Chen, and B. Ding. ”Discovering threshold-based frequent closed itemsets over probabilistic data”. In *ICDE*, 2012.

[25] L. Wang, R. Cheng, and D. Lee, and D. Cheung. ”Accelerating probabilistic frequent itemset mining: A model-based approach”. In *CIKM*, 2010.

[26] Z. Zou, J. Li, and H. Gao. ”Discovering frequent subgraphs over uncertain graph databases under probabilistic semantics”. In *SIGKDD*, 2010.

[27] L. Wang, D. Cheung, R. Cheng, and S. Lee, and X. Yang. ”Efficient Mining of Frequent Itemsets on Large Uncertain Databases”. In *TKDE*, 2011.

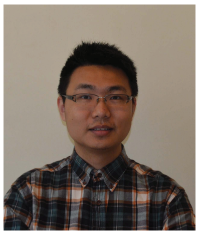
[28] Y. Tong, L. Chen, Y. Cheng and P.S. Yu ”Mining frequent itemsets over uncertain databases”. In *VLDB*, 2012

[29] L. Le Cam. ”An approximation theorem for the Poisson binomial distribution”. In *Pacific Journal of Mathematics*, 1960.

[30] A. Volkova. ”A refinement of the central limit theorem for sums of independent random indicators”. In *Theory of Probability and its Applications*, 1995.

[31] Y. Hong. ”On computing the distribution function for the sum of independent and non-identical random indicators”. In *Technical Report, Department of Statitics, Virginia Tech, Blacksburg, VA*

[32] The Lahar Project: http://lahar.cs.washington.edu/displayPage.php?path=./content/Download/RFIDData/rfidData.html



Zhou Zhao received his BS degree in computer science from the Hong Kong University of Science and Technology (HKUST), in 2010. He is currently a PhD student in the Department of Computer Science and Engineering, HKUST. His research interests include data cleansing and data mining.

Da Yan received his BS degree in computer science from Fudan University, Shanghai, in 2009. He is currently a PhD student in the Department of Computer Science and Engineering, Hong Kong University of Science and Technology. His research interests include spatial data management, uncertain data management and data mining.

Wilfred Ng received his MSc (Distinction) and PhD in Computer Science from the University of London. Currently he is an Associate Professor of Computer Science and Engineering at the Hong Kong University of Science and Technology, where he is a member of the database research group. His research interests are in the areas of databases, data mining and information Systems, which include Web data management and XML searching. Further Information can be found at the following URL: http://www.cs.ust.hk/faculty/wilfred/index.html.