Mining Probabilistically Frequent Sequential Patterns in Uncertain Databases

Zhou Zhao, Da Yan and Wilfred Ng

Department of Computer Science and Engineering
The Hong Kong University of Science and Technology
Hong Kong
{zhaozhou,yanda,wilfred}@cse.ust.hk

ABSTRACT

Data uncertainty is inherent in many real-world applications such as environmental surveillance and mobile tracking. As a result, mining sequential patterns from inaccurate data, such as sensor readings and GPS trajectories, is important for discovering hidden knowledge in such applications. Previous work uses *expected support* as the measurement of *pattern frequentness*, which has inherent weaknesses with respect to the underlying probability model, and is therefore ineffective for mining high-quality sequential patterns from uncertain sequence databases.

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. Based on the *prefix-projection* strategy of 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 world explosion", and when combined with our three pruning techniques and one validating technique, achieves good performance. The efficiency and effectiveness of *U-PrefixSpan* are verified through extensive experiments on both real and synthetic datasets.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications—*Data mining*

General Terms

Algorithms, Design, Performance

1. INTRODUCTION

The problem of mining *frequent sequential patterns* (*FSPs*) from deterministic databases has attracted a lot of attention in the research community [4, 5, 6, 7, 8] due to its wide spectrum of real life applications. For example, in mobile tracking systems, FSPs are useful in the classification or clustering of moving objects [2]; and in biological research, FSP mining can help discover correlations among gene sequences [3].

SID	Sequence Instances	Probability
s_1	$s_{11} = ABC$	1
s_2	$s_{21} = AB$	0.9
	$s_{22} = BC$	0.05

(a) Uncertain Data Model

Possible World	Probability
$pw_1 = \{s_{11}, s_{21}\}$	$1 \times 0.9 = 0.9$
$pw_2 = \{s_{11}, s_{22}\}$	$1 \times 0.05 = 0.05$
$pw_3 = \{s_{11}\}$	$1 \times 0.05 = 0.05$

(b) Possible World Space

Figure 1: Sequence-Level Uncertain Data Model

Data uncertainty is inherent in many real-world applications such as sensor data monitoring [11], RFID localization [10] and location-based services [9], due to environmental factors, device limitations, privacy issues, etc. As a result, uncertain data mining has attracted a lot of attention in recent research. A comprehensive survey of the techniques on uncertain data mining can be found in [17].

To our knowledge, [1] is the only work that studies mining sequential patterns from uncertain data. However, this work adopts *expected support* as the measurement of pattern frequentness, which has some inherent weaknesses with respect to the fundamental probability model, and is therefore ineffective for mining high-quality sequential patterns from uncertain sequence databases. We will illustrate this point in our later examples.

In this paper, we propose to define pattern frequentness based on the *possible world semantics*. 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 which we define the problem of mining *probabilistically frequent sequential patterns* (or *p-FSPs*). We now introduce our data models through the following examples.

Consider a wireless sensor network (WSN) system, where each sensor continuously collects readings about environmental conditions, such as temperature and humidity, within its detection range. In such a case, the readings of a sensor 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, (e.g. reading A represents $[5^\circ, 7^\circ)$ and reading B represents $[7^\circ, 9^\circ)$), 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_1 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 sensors in another region within that time period.

In Figure 1(a), we assume that the reading sequences detected

EDBT 2012, March 26–30, 2012, Berlin, Germany.

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

(a) Uncertain Data Model

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) Possible World Space

Figure 2: Element-Level Uncertain Data Model

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 assumed to be independent. We call such a data model the *sequence-level uncertain model*. In the Trio [19] system, probabilistic sequences such as s_1 and s_2 are also called *x-tuples*.

Figure 1(b) shows the set of possible worlds derived from the uncertain dataset of 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 in Figure 1.

The existing approach to evaluating the frequentness of a sequential pattern α in an uncertain database is to compute its expected support: 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.

Expected support is used by some existing studies to measure the frequentness of patterns such as frequent itemsets [13, 16] and frequent subsequences [1] on uncertain data. However, the notion of *expected support fails to reflect pattern frequentness in many cases*. To illustrate this weakness, we now consider pattern $\alpha = AB$ 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, in this example, using expected support leads to the missing of the important sequential pattern AB.

While the *sequence-level uncertain model* is fundamental for a lot of real-life applications, many applications follow a different model. Consider an RFID location tracking system, where a set of RF readers are deployed in an indoor environment, and a user may be detected by several near-by readers simultaneously. In this application, user locations are uncertain. This kind of uncertainty is common in many related applications such as PEEX[20].

Consider the uncertain sequence database shown in Figure 2(a), where sequences s_1 and s_2 record the tracking paths of two users, respectively. 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 presented 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.

From the above example, we can see 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, an example of which can be found in [14]. Therefore, we propose to evaluate the frequentness of a sequential pattern more adhering to the probability theory. This gives rise to the idea of *probabilistic frequentness*, which is able to capture the intricate relationship 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.

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 world explosion". Our contributions are summarized as follows:

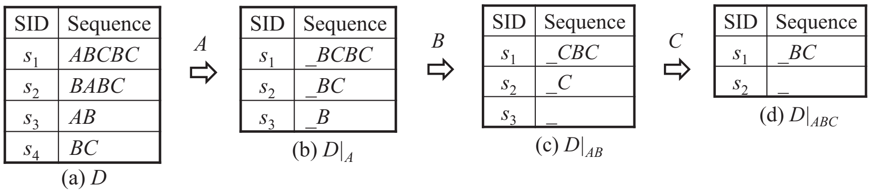
- To our knowledge, this is the first work that attempts to solve the problem of p-FSP mining from uncertain data, the techniques of which are successfully applied in an RFID application for trajectory pattern mining.
- We consider two general uncertain sequence data models that are abstracted from many real-life applications involving unceratin 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 to mine p-FSPs from uncertain data conforming to our models.
- Several pruning techniques 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 described in Section 5. In Section 6, 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 7.

2. RELATED WORK

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].



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

(a) *D*

A ⇨

SID	Sequence
s_1	_*BCBC*
s_2	_*BC*
s_3	_*B*

(b) $D|_A$

B ⇨

SID	Sequence
s_1	_*CBC*
s_2	_*C*
s_3	_

(c) $D|_{AB}$

C ⇨

SID	Sequence
s_1	_*BC*
s_2	_

(d) $D|_{ABC}$

Figure 3: Illustration of PrefixSpan

PrefixSpan is demonstrated in [4] to be superior than the other sequence mining algorithms such GSP and FreeSpan, due to its *prefix-projection* technique. *PrefixSpan* has been used in a lot of applications such as trajectory mining [2], and is also the foundation of this work. We review the *prefix-projection* technique of *PrefixSpan* next.

PrefixSpan. For ease of presentation, we denote $\alpha\beta$ to be the sequence resulted from concatenating sequence α with sequence β. Also recall (from Section 1) that $\alpha \sqsubseteq s$ corresponds to the event that sequence α occurs as a subsequence of s. We now define some concepts that are necessary 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 Figure 3(b), (c) and (d), respectively.

PrefixSpan finds the frequent patterns (with support at least τ_{sup}) by recursively checking the frequentness of patterns with growing lengths. In each iteration, if the current pattern α is checked to be frequent, it will recurse on all the possible patterns α' constructed by appending α with one more element. *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 $D|_{AB}$.

2.2 Pattern Mining on Uncertain Data

Frequent itemset mining and sequential pattern mining are two of the most important pattern mining problems studied in the context of uncertain data. For the problem of frequent itemset mining, earlier work usually uses expected support to measure pattern frequentness, such as [13, 16]. However, [14] and [15] find that the use of expected support may render important patterns missing. As a result, recent research focuses more on using probability measurements, such as [15, 12].

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. [18] studies the problem of mining long sequential patterns in a noisy environment. 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 similar to our probabilistic sequence models have been used to model probabilistic set [21] and probabilistic string [22] in researches on similarity join.

3. PRELIMINARIES

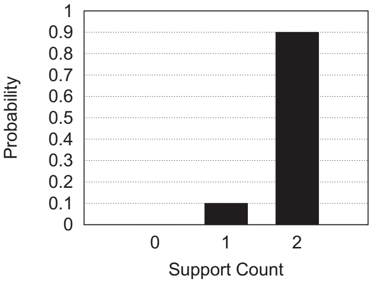


Figure 4: Probability Distribution of $sup(AB)$

Support as a random variable. While pattern support $sup(\alpha)$ is just a count for deterministic databases, in the context of uncertain databases $sup(\alpha)$ becomes a random variable. We clarify this point next.

Given a sequence-level or an element-level uncertain sequence database D, let us denote its possible world space as $\mathcal{PW} = \{pw_1, pw_2, \ldots, pw_{|\mathcal{PW}|}\}$. We also denote $sup_i(\alpha)$ to be the support of pattern α in possible world $pw_i \in \mathcal{PW}$. Since pw_i is a deterministic database instance, $sup_i(\alpha)$ is just a count (equal to $|\{s \in pw_i | \alpha \sqsubseteq s\}|$). Note that each possible world pw_i is also associated with an occurrence probability $Pr(pw_i)$, and therefore, given pattern α, each possible world pw_i corresponds to a pair $(sup_i(\alpha), Pr(pw_i))$. In the example dataset of Figure 1, given pattern AB, 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 random variable whose probability distribution is given by 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, $f_\alpha(c)$ can be represented as the following formula:

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

Probabilistic frequentness. Now we are ready to introduce the concept of *probabilistic frequentness* (or simply *p-frequentness*):

DEFINITION 3 (PROBABILISTIC FREQUENTNESS). *Given a probability threshold τ_{prob} and a support threshold τ_{sup}, pattern α is probabilistically frequent (or p-frequent) iff*

$$Pr\{sup(\alpha) \geq \tau_{sup}\} \ \geq \ \tau_{prob}. \quad (1)$$

The L.H.S. of Equation 1 can be represented as

$$Pr\{sup(\alpha) \geq \tau_{sup}\} = \sum_{c=\tau_{sup}}^{n} f_\alpha(c) = 1 - F_\alpha(\tau_{sup} - 1). \quad (2)$$

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

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

  PROOF. When $cnt(\alpha) < \tau_{sup}$, $Pr\{sup(\alpha) \geq \tau_{sup}\} \leq Pr\{sup(\alpha) > cnt(\alpha)\} = 0$. □

- **(2) MarkovPrune.** Pattern α is not *p-frequent* if $expSup(\alpha) < \tau_{sup} \times \tau_{prob}$.

  PROOF. According to Markov's inequality, $expSup(\alpha) < \tau_{sup} \times \tau_{prob}$ implies $Pr\{sup(\alpha) \geq \tau_{sup}\} \leq expSup(\alpha)/\tau_{sup} < \tau_{prob}$. □

- **(3) ExpPrune.** Let $\mu = expSup(\alpha)$ and $\delta = \frac{\tau_{sup} - \mu - 1}{\mu}$. When $\delta > 0$, pattern α is not *p-frequent* if

  - $\delta \geq 2e - 1$, and $2^{-\delta\mu} < \tau_{prob}$, or
  - $0 < \delta < 2e - 1$, and $e^{-\frac{\delta^2\mu}{4}} < \tau_{prob}$.

  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 [12] 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 pattern α can not be pruned, we have to check whether Equation (1) holds. According to Equation (2), this is equivalent to computing the pmf $f_\alpha(c)$.

In fact, *evaluating $f_\alpha(c)$ on α-projected (uncertain) database $D|_\alpha$ is equivalent to evaluating $f_\alpha(c)$ on the original database D*, since $\forall s \notin D|_\alpha$, $Pr\{\alpha \sqsubseteq s\} = 0$. Therefore, 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 the divide-and-conquer strategy. Given a set S of probabilistic sequences, we divide it into two partitions S_1 and S_2. Let us define $f^S_\alpha(c)$ as the pmf of $sup(\alpha)$ on sequence set S, then our ultimate goal is to compute $f^{D|_\alpha}_\alpha(c)$.

Now, let us consider how to obtain $f^S_\alpha(c)$ from $f^{S_1}_\alpha(c)$ and $f^{S_2}_\alpha(c)$. Let us denote $sup^S(\alpha)$ to be the support of α on uncertain sequence set 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 $f^S_\alpha(c)$ can be computed by the following formula:

$$f^S_\alpha(c) = \sum_{i=0}^{c} f^{S_1}_\alpha(i) \times f^{S_2}_\alpha(c-i). \quad (3)$$

Note that according to Equation (3), f^S_α is the convolution of $f^{S_1}_\alpha$ and $f^{S_2}_\alpha$. Therefore, f^S_α can be computed from $f^{S_1}_\alpha$ and $f^{S_2}_\alpha$ in $O(n \log n)$ time using the *Fast Fourier Transform* (FFT) algorithm [23], where $n = |S|$. When S is large, this approach is much better than naïvely evaluating Equation (3) for all c, which takes $O(n^2)$ time.

It is not always necessary to compute $f^{D|_\alpha}_\alpha(c)$ to the end, in order to determine whether pattern α is p-frequent. In fact, we can conclude that a pattern α is *p-frequent* on $D|_\alpha$, as long as α is found to be *p-frequent* on a subset of $D|_\alpha$, which is formalized by the following theorem.

THEOREM 1 (EARLY VALIDATING). *Suppose that pattern α is p-frequent on $S' \subseteq S$, then α is also p-frequent on S.*

PROOF. Suppose that probabilistic sequence set S is divided it into two partitions S_1 and S_2. It is sufficient to prove that, when α is *p-frequent* on S_1, it is also *p-frequent* on S.

When α is *p-frequent* on S_1, according to Equation (2), we have

$$1 - F^{S_1}_\alpha(\tau_{sup} - 1) = Pr\{sup^{S_1}(\alpha) \geq \tau_{sup}\} \geq \tau_{prob}. \quad (4)$$

According to Equation (4), $F^{S_1}_\alpha(\tau_{sup} - 1) \leq 1 - \tau_{prob}$. If we can prove $F^S_\alpha(\tau_{sup} - 1) \leq F^{S_1}_\alpha(\tau_{sup} - 1)$, then we are done since this implies $F^S_\alpha(\tau_{sup} - 1) \leq F^{S_1}_\alpha(\tau_{sup} - 1) \leq 1 - \tau_{prob}$, or equivalently, $Pr\{sup^S(\alpha) \geq \tau_{sup}\} = 1 - F^S_\alpha(\tau_{sup} - 1) \geq \tau_{prob}$.

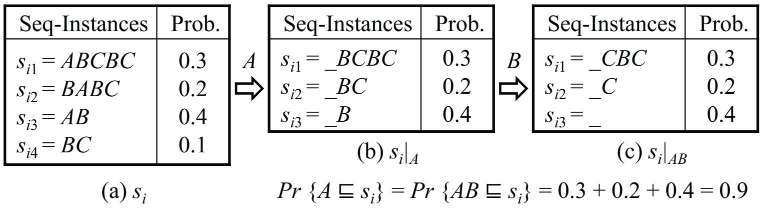
We now prove $F^S_\alpha(\tau_{sup} - 1) \leq F^{S_1}_\alpha(\tau_{sup} - 1)$. Let us denote $\tau'_{sup} = \tau_{sup} - 1$. Then, we obtain

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

□

Algorithm 1 shows our divide-and-conquer algorithm (*PMFCheck*) to determine the *p-frequentness* of pattern α on an uncertain sequence set $S = \{s_1, s_2, \ldots, s_n\}$. The input to *PMFCheck* is a vector vec_α where each element $vec_\alpha[i] = Pr\{\alpha \sqsubseteq s_i\}$. Note that the terminology "vector" here actually refers to a set (of probability values) where element order is not important.

Since the order of sequences in S has no influence on the pmf $f^S_\alpha(c)$, and sequence s_i does not contribute to $f^S_\alpha(c)$ if $Pr\{\alpha \sqsubseteq$



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

(b) $s_i|_A$

$\xrightarrow{B}$

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

(c) $s_i|_{AB}$

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

Figure 5: Sequence Projection in Sequence-Level Model

Algorithm 1 *PMFCheck*(vec_α)

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

```
1: if |vec_α|=1 then
2:     $f_\alpha(0) \leftarrow 1 - vec_\alpha[1]$, $f_\alpha(1) \leftarrow vec_\alpha[1]$
3:     return $(1 - F_\alpha(\tau_{sup}) \geq \tau_{prob}, f_\alpha)$
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 $(1 - F_\alpha(\tau_{sup}) \geq \tau_{prob}, f_\alpha)$
```

$s_i\} = 0$, we always exclude the elements with value 0 from vec_α before each invocation of *PMFCheck*.

PMFCheck partitions vec_α into two halves: vec_α^1 (and vec_α^2) for the first (and the second) half S_1 (and S_2) of S (Line 4). If α is found to be *p-frequent* on 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 on S_1 and S_2 (i.e.f_α^1 and f_α^2), to compute the pmf of pattern α on S in Line 11. After obtaining f_α, we can check whether α is *p-frequent* on S by Equations (1) and (2) (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 us denote $T(n)$ as 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 sequential pattern α is not p-frequent, then any pattern β satisfying $\alpha \sqsubseteq \beta$ is not p-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 a pattern α for examination, 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 issues arising from the sequence-level uncertain 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 $s_{ij} \in s_i$) to $s_{ij}|_\alpha$, excluding those instances that cannot be projected (due to $\alpha \not\sqsubseteq s_{ij}$), like 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 (1) a pointer to s_{ij} and (2) 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 projected probabilistic sequence $s_i|_\alpha$ is represented as a list of pairs, where each pair correponds to an instance s_{ij} and has format $(s_{ij}|_\alpha, Pr(s_{ij}))$. To illustrate our represention format, in Figure 5(c), we have $s_i|_{AB} = \{(s_{i1}|_{AB}, 0.3), (s_{i2}|_{AB}, 0.2), (s_{i3}|_{AB}, 0.4)\}$ where, for example, $s_{i1}|_{AB} = <2, s_{i1}>$.

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

Pattern Frequentness Checking. Recall that given a projected database $D|_\alpha$, we check the *p-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 pattern frequentness checking 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:

$$
\begin{aligned}
& Pr\{\alpha \sqsubseteq s_i\} \\
= & \sum_{s_{ij} \in s_i} Pr\{\alpha \sqsubseteq s_{ij} \mid s_i \text{ occurs as } s_{ij}\} \times Pr(s_{ij}) \\
= & \sum_{s_{ij}|_\alpha \,\in\, s_i|_\alpha} Pr(s_{ij}). \qquad (5)
\end{aligned}
$$

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, in Figure 5(c), $Pr\{AB \sqsubseteq s_i\} = Pr(s_{i1}) + Pr(s_{i2}) + Pr(s_{i3}) = 0.9$.

Candidate Elements for Pattern Growth. Given a p-FSP α, we need to examine whether the patterns β grown from α are *p-frequent*. Note that $\alpha \sqsubseteq \beta$. In fact, α is a prefix of β.

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

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

```
$T|_{\alpha e} \leftarrow \emptyset$
for each element $\ell \in T|_\alpha$ do
    Check CntPrune with pattern $\ell$ on $D|_{\alpha e}$
    if $\ell$ is not pruned then
        Check MarkovPrune with pattern $\ell$ on $D|_{\alpha e}$
        if $\ell$ is not pruned then
            Check ExpPrune with pattern $\ell$ on $D|_{\alpha e}$
    if $\ell$ is not pruned then
        $T|_{\alpha e} \leftarrow T|_{\alpha e} \cup \{\ell\}$
```

Recall that in *PrefixSpan*, in each recursion iteration, if the current pattern α is frequent, we grow α by one element e to obtain new patterns αe, and recurcively check the frequentness of αe. To keep the number of new patterns small in each growth step, we maintain an element table $T|_\alpha$ that stores only those elements e still having chance to make pattern αe *p-frequent*.

We present an important property for elements tables:

PROPERTY 2. *For a pattern β grown from α, $T|_\beta \subseteq T|_\alpha$.*

PROOF. Let us denote $\beta = \alpha\gamma$. For any element $e \notin T|_\alpha$, αe is not *p-frequent*, and since $\alpha e \sqsubseteq \alpha\gamma e = \beta e$, βe is not *p-frequent* according to pattern anti-monotonicity, which implies that $e \notin T|_\beta$ either. □

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

We construct $T|_{\alpha e}$ from $T|_\alpha$ during pattern growth using Algorithm 2, which is quite self-explanatory. 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 present our *SeqU-PrefixSpan* algorithm in Algorithm 3. Given a sequence-level probabilistic database $D = \{s_1, \ldots, s_n\}$, we grow patterns starting from $\alpha = \emptyset$. Recall our projected sequence/instance format, we have $D|_\emptyset = \{s_1|_\emptyset, s_2|_\emptyset, \ldots, s_n|_\emptyset\}$, where for each sequence $s_i|_\emptyset$, its instance $s_{ij}|_\emptyset = <0, Pr(s_{ij})>$ (the "pos" field is the one before the first position of s_{ij}, which is 0). Let us denote T_0 to be the table of all possible elements in D, then 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*$(e, D|_\emptyset, T|_\emptyset)$.

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 *p-frequent* by invoking *PMFCheck*(vec_β) in Line 13.

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$(\alpha e\ell, D|_{\alpha e}, T|_{\alpha e})$
19: Free $D|_{\alpha e}$ and $T|_{\alpha e}$ from memory
```

To compute $Pr\{\beta \sqsubseteq s_i\}$ using Equation (5), we first initialize $pr(s_i|_{\alpha e})$ to 0 (Line 3), and whenever we find that $s_{ij} \in s_i|_{\alpha e}$ which is 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 *p-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 $T|_\alpha$, so as to obtain the current truncated element table $T|_\beta$. Then, we check the patterns grown from β by running the recursion on $D|_\beta$ and $T|_\beta$ in Lines 17–18.

5. ELEMENT-LEVEL U-PREFIXSPAN

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

Consider the element-level probabilistic sequence s_i shown in Figure 6(a) that have four probabilistic elements. Figure 6(b) shows the possible world space of s_i, where each possible world corresponds to a sequence that s_i may occur to be.

An interesting observation is that the possible world space of s_i is exactly the sequence-level representation of s_i. Therefore, one 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 adopt a more efficient approach that expands the probabilistic sequence only when necessary. For example, when we consider pattern BA on the probabilistic sequence s_i in Figure 6, the expansions related to C are totally unnecessary, since whether C occurs in s_i or not has no influence on $Pr\{BA \sqsubseteq s_i\}$.

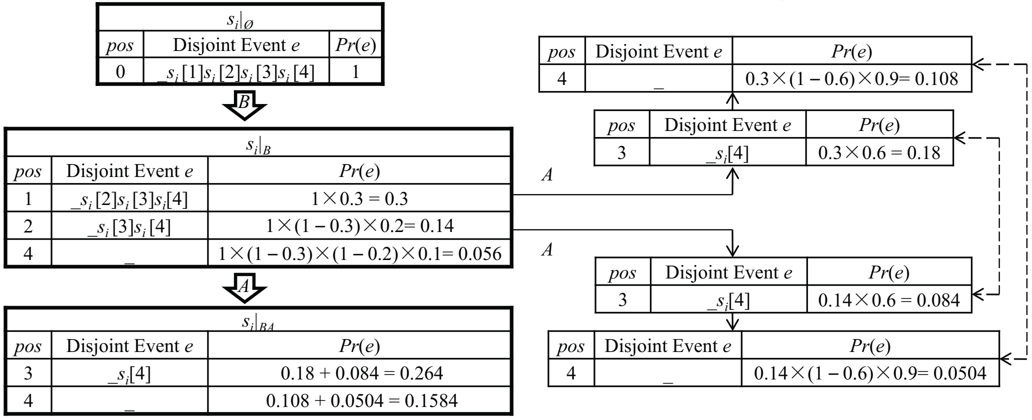
The differences between *ElemU-PrefixSpan* and *SeqU-PrefixSpan*

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

(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	$\_s_i[2]s_i[3]s_i[4]$	1×0.3 = 0.3
2	$\_s_i[3]s_i[4]$	1×(1 – 0.3)×0.2= 0.14
4	_	1×(1 – 0.3)×(1 – 0.2)×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×(1 – 0.6)×0.9= 0.108

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

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

pos	Disjoint Event *e*	*Pr*(*e*)
4	_	0.14×(1 – 0.6)×0.9= 0.0504

(c) Projection

Figure 6: Illustration of *ElemU-PrefixSpan*

mainly lie in two aspects: (1) sequence projection from s_i to $s_i|_\alpha$, 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 discuss how to obtain the projected probabilistic sequence $s_i|_\alpha$.

DEFINITION 4. *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 4, $s_i[1, \ldots, pos]$ is the *minimal* prefix of s_i that contains pattern α. Event $e_{pos}(s_i, \alpha)$ can be recursively constructed in the following manner:

(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 4 should be $\emptyset$, which implies $pos = |s_i[1, \ldots, pos]| = |\emptyset| = 0$.

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

$$e_{pos}(s_i, \beta) \;= \bigcup_{k<pos} \{\, e_k(s_i, \alpha) \;\wedge\; s_i[pos] = e \;\wedge\; s_i[j] \neq e, \forall k < j < pos \,\}. \quad (6)$$

This is 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 (6) with different k are disjoint to each other due to the *minimality requirement* of Definition 4. Note that the events in the *UNION* operator of Equation (6) 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 compute $Pr(e_{pos}(s_i, X))$ according to Equation (6):

$$\begin{aligned} & Pr(e_{pos}(s_i, \alpha e)) \\ = & \sum_{k<pos} [\, Pr(e_k(s_i, \alpha)) \times Pr\{s_i[pos] = e\} \\ & \times \prod_{k<j<pos} (1 - Pr\{s_i[j] = e\}) \,]. \end{aligned} \quad (7)$$

Equation (7) 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 $e_k(s_i, \alpha), 0 < k \leq len(s_i)$, 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, $Pr(e_1(s_i, B)) = Pr(e_0(s_i, \emptyset)) \times Pr\{s_i[1] = B\} = 0.3$.
- $e_2(s_i, B) = \{s_i[1] \neq B \;\wedge\; s_i[2] = B\}$. In this case, $Pr(e_2(s_i, B)) = Pr(e_0(s_i, \emptyset)) \times (1 - Pr\{s_i[1] = B\}) \times Pr\{s_i[2] = B\} = 0.14$.
- $e_4(s_i, B) = \{s_i[1] \neq B \wedge s_i[2] \neq B \wedge s_i[4] = B\}$. In this case, $Pr(e_4(s_i, B)) = Pr(e_0(s_i, \emptyset)) \times (1 - Pr\{s_i[1] = B\}) \times (1 - Pr\{s_i[2] = B\}) \times Pr\{s_i[4] = B\} = 0.0056$.

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 $D|_{\alpha e}$
```

For the case when α grows from B to BA, let us 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 can occur in any of $s_i[3]$ and $s_i[4]$, 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 in 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 cannot 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 third/bottom table in Figure 6(c)), which is based on Equation (7).

Algorithm 4 shows our algorithm that constructs $D|_\beta$ ($\beta = \alpha e$) from the old projected database $D|_\alpha$. For each projected probabilistic sequence $s|_\alpha$, we project it into a new projected sequence $s|_\beta \in D|_\beta$ in Lines 2–19. In our algorithm, each projected probabilistic sequence is represented as a set of events (recall Figure 6(c)), and each event $r = e_{pos}(s_i, \alpha)$ is represented as 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 $accum_r$ with each event r, which is used to record the value of the product term on the R.H.S. of Equation (7) and 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 the event with minimum position value $pivot_r$ to be scanned next (Line 8), denoted as r'. If the probabilistic element in the current position $pivot_{r'}$ can take value e (Line 9) (with probability p_e), then we can compute the probability of the sub-event derived from r' as Δ using Equation (7) (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 minimum position value in each iteration, the sub-events are constructed with **non-decreasing** values of pos. According to Equation (7), we know that we can sum up 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 with that of 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).

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

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

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

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 is much better than the full-expansion approach mentioned in the beginning of this section, where each $s|_\alpha$ is expanded to many sequence instances, the number of which 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. $s_i|_{e_{pos}(s_i,\alpha)}$). Therefore, we have

$$\begin{aligned} & Pr\{\beta \not\sqsubseteq s_i\} \\ = & \sum_{pos} Pr\{\beta \not\sqsubseteq s_i | e_{pos}(s_i, \alpha)\} \times Pr(e_{pos}(s_i, \alpha)) \\ = & \sum_{pos} Pr\{s_i[j] \neq e, \forall j > pos\} \times Pr(e_{pos}(s_i, \alpha)) \\ = & \sum_{pos} \left(Pr(e_{pos}(s_i, \alpha)) \times \prod_{i > pos} (1 - Pr\{s_i[pos] = e\}) \right), \quad (8) \end{aligned}$$

and thus

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

Algorithm 5 shows how we compute the factor in the last line of Equation (9). Algorithm 6 shows our *ElemU-PrefixSpan* algo-

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
```

rithm, where Line 6 computes Equation (9) as $accum$ by using Algorithm 5. After obtaining $Pr\{\beta \sqsubseteq s_i\}$ for all $s_i|_\beta \in D|_\beta$, we can check the *p-frequentness* of β and prune the element table in a similar manner as in Algorithm 3.

6. EXPERIMENTS

In this section, we evaluate the performance of our two *U-PrefixSpan* algorithms using both real and synthetic datasets. Specifically, we test the scalability of *SeqU-PrefixSpan* and *ElemU-PrefixSpan* on large synthetic datasets in Sections 6.1 and 6.2, respectively. In Section 6.3, we compare *ElemU-PrefixSpan* with the naïve 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, we successfully apply *ElemU-PrefixSpan* in an RFID application for trajectory pattern mining, and the results confirm that *probabilistic frequentness* is more accurate than *expected support* for evaluating pattern frequentness on uncertain data.

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.

6.1 SeqU-PrefixSpan Scalability Results

Synthetic Data Generation. To test the scalability of *SeqU-PrefixSpan*, we implemented a data generator to generate datasets that conform to the sequence-level uncertain model. Given the configuration parameter set (n, m, ℓ, d), our generator generates n probabilistic sequences. For each probabilistic sequence, the number of sequence instances is randomly chosen from range $[1, m]$, and therefore the expected number of sequence instances for each sequence is $m/2$. The length of a sequence instance is randomly chosen from range $[1, \ell]$, and each element in the sequence instance is randomly picked from an element table with d elements.

For a probabilistic sequence $s_i = \{s_{i1}, s_{i2}, \ldots, s_{im_i}\}$, we generate the probabilities $Pr(s_{ij})$ as follows: we first add a new dummy instance s_{i0} to s_i (s_{i0} corresponds to case where s_i does not occur), and then for each sequence instance s_{ij} $(j = 0, 1, \ldots, m_i)$, we generate a value w_{ij} following uniform distribution in range $(0, 1)$. Finally, we normalize the probability values to be $Pr(s_{ij}) = w_{ij} / \sum_{j=0}^{m_i} w_{ij}$ $(j = 1, 2, \ldots, m_i)$. Note that $\sum_{j=1}^{m_i} Pr(s_{ij})$ never exceeds 1 due to the introduction of the dummy instance s_{i0}.

Experimental Setting. In addition to the four dataset configuration parameters n, m, ℓ and d, we have two threshold parameters: the support threshold τ_{sup} and the probability threshold τ_{prob}. Therefore, we have six parameters in total. We generate five datasets for each dataset configuration (n, m, ℓ, d), and the reported results are averaged on the corresponding five runs.

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

Effect of n, m, ℓ and d on Execution Time. The experimental results are summarized as follows:

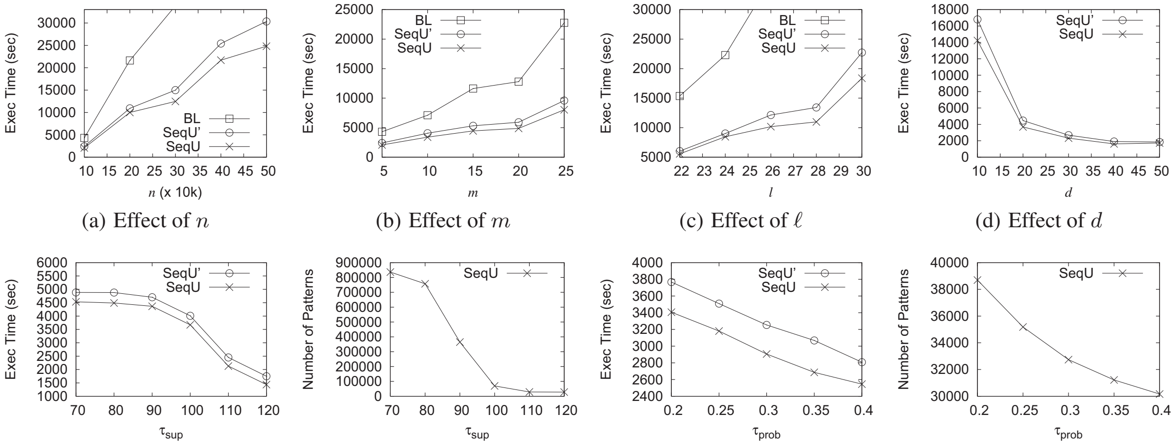
- Figure 7(a) shows the execution time of *SeqU-PrefixSpan* variations when n varies from 100,000 to 500,000, where we fix $m = 5$, $\ell = 20$, $d = 30$, $\tau_{sup} = 12$ and $\tau_{prob} = 0.4$.
- Figure 7(b) shows the execution time of *SeqU-PrefixSpan* variations when m varies from 5 to 25, where we fix $n =$ 100k, $\ell = 20$, $d = 30$, $\tau_{sup} = 12$ and $\tau_{prob} = 0.4$.
- Figure 7(c) shows the execution time of *SeqU-PrefixSpan* variations when ℓ from 22 to 30, where we fix $n =$ 100k, $m = 5$, $d = 30$, $\tau_{sup} = 12$ and $\tau_{prob} = 0.4$.
- Figure 7(d) shows the execution time of *SeqU* and *SeqU'* when d varies from 10 to 50, where we fix $n =$ 100k, $m = 5$, $\ell = 20$, $\tau_{sup} = 12$ and $\tau_{prob} = 0.4$.

From these results, we observe the following trends:

- In all the experiments, *BL* is around 2–3 times slower than *SeqU'*, which verifies the effectiveness of the pruning methods. *SeqU'* is around 10%–20% slower than *SeqU*, which verifies the effectiveness of the validating method.
- The running time of all the algorithms increase with the increment of parameters n, m and ℓ. This trend is intuitive since larger n, m and ℓ implies larger data size. In particular, the running time of all the algorithms increases almost linearly with the increment of n.
- The running time of *SeqU* and *SeqU'* decreases with the increment of d. This is mainly because, 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 means that *SeqU-PrefixSpan* does not have to recurse to deep levels.

Effect of τ_{sup} and τ_{prob} on Execution Time and Number of Results. The experimental results are summarized as follows:

- Figures 7(e) and 7(f) show the running time and result size of *SeqU* and *SeqU'* when the support threshold τ_{sup} varies from 70 to 120, where we fix $n =$ 100k, $m = 5$, $\ell = 20$, $d = 30$ and $\tau_{prob} = 0.4$.
- Figures 7(g) and 7(h) show the running time and result size of *SeqU* and *SeqU'* when the probability threshold τ_{prob} varies from 0.2 to 0.4, where we fix $n =$ 100k, $m = 5$, $\ell = 20$, $d = 30$ and $\tau_{sup} = 12$.



(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 Result # (g) Effect of τ_{prob} on Time (h) Effect of τ_{prob} on Result #

Figure 7: Scalability Results on Sequence-Level Uncertain Model

From these results, we observe the following trend: both the running time and result number *SeqU* and *SeqU'* decrease with the increment of parameters τ_{sup} and τ_{prob}. This trend is intuitive since larger τ_{sup} and τ_{prob} implies higher requirement on pattern frequentness, which further implies less valid patterns.

6.2 ElemU-PrefixSpan Scalability Results

Synthetic Data Generation. Similarly to the study of *SeqU-PrefixSpan*, to test the scalability of *ElemU-PrefixSpan*, we generate datasets that conform to the element-level uncertain model. Given the configuration parameter set (n, m, ℓ, d), our generator generates n probabilistic sequences. For each probabilistic sequence, its length is randomly chosen from range $[1, \ell]$, and therefore the expected number of probabilistic elements for each sequence is $\ell/2$. In each probabilistic sequence, 20% of the elements are sampled to be uncertain. The number of element instances of a probabilistic element is randomly chosen from range $[1, m]$, and therefore, each probabilistic element has $m/2$ element instances on average. Each element instance is randomly picked from an element table with d elements.

For each probabilistic element $e_i = \{e_{i1}, e_{i2}, \ldots, e_{im_i}\}$, we generate the probabilities $Pr(e_{ij})$ as follows: we first add a new dummy element e_{i0} to e_i (e_{i0} corresponds to case where e_i does not occur), and then for each element instance e_{ij} $(j = 0, 1, \cdots, m_i)$, we generate a value w_{ij} following uniform distribution in range $(0, 1)$. Finally, we normalize the probability values to be $Pr(e_{ij}) = w_{ij}/\sum_{j=0}^{m_i} w_{ij}$ $(j = 1, 2, \cdots, m_i)$. Note that $\sum_{j=1}^{m_i} Pr(e_{ij})$ never exceeds 1 due to the introduction of the dummy element e_{i0}.

Similar to the sequence-level case, we have altogether six parameters, and for each dataset configuration, we generate five datasets and the reported results are averaged on the five runs. We detail the experimental results as follows:

- Figure 8(a) shows the running time of *ElemU-PrefixSpan* variations when n varies from 100,000 to 500,000, where we fix $m = 5$, $\ell = 20$, $d = 30$, $\tau_{sup} = 110$ and $\tau_{prob} = 0.5$.
- Figure 8(b) shows the running time of *ElemU-PrefixSpan* variations when m varies from 6 to 10, where we fix $n =$ 100k, $\ell = 20$, $d = 30$, $\tau_{sup} = 110$ and $\tau_{prob} = 0.5$.
- Figure 8(c) shows the running time of *ElemU-PrefixSpan* variations when ℓ varies from 22 to 30, where we fix $n =$ 100k, $m = 5$, $d = 30$, $\tau_{sup} = 110$ and $\tau_{prob} = 0.5$.
- Figure 8(d) shows the running time of *ElemU* and *ElemU'* when d varies from 10 to 50, where we fix $n =$ 100k, $m = 5$, $\ell = 20$, $\tau_{sup} = 110$ and $\tau_{prob} = 0.5$.
- Figures 8(e) and 8(f) show the running time and result size of *ElemU* and *ElemU'* when the support threshold τ_{sup} varies from 12 to 28, where we fix $n =$ 100k, $m = 5$, $\ell = 20$, $d = 30$ and $\tau_{prob} = 0.5$.
- Figures 8(g) and 8(h) show the running time and result size of *ElemU* and *ElemU'* when the probability threshold τ_{prob} varies from 0.2 to 0.4, where we fix $n =$ 100k, $m = 5$, $\ell = 20$, $d = 30$ and $\tau_{sup} = 110$.

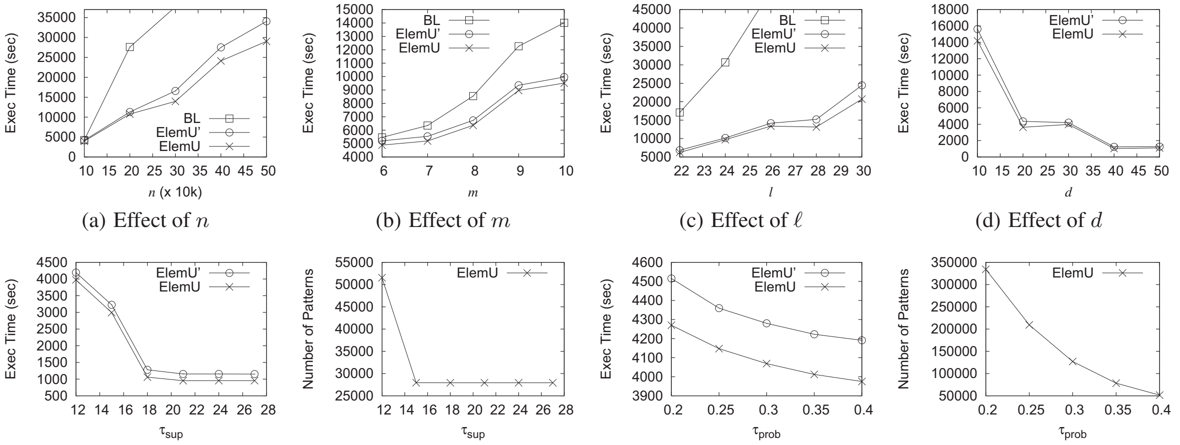
The trends observed from these results are similar to those observed from the scalability test of *SeqU-PrefixSpan* in Section 6.1, and so is the corresponding analysis.

6.3 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 6.2 to generate experimental data, with the default setting $(n, m, \ell, d) = (10k, 5, 20, 30)$. Figures 9(a)–(d) shows 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 as the default ones. Note that for the naïve method, we do not include the time for sequence expansion (i.e. we only count the mining time of *SeqU-PrefixSpan*).

In Figures 9(a), (c) and (d), *ElemU-PrefixSpan* is around 20–50 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 with



(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 Result # (g) Effect of τ_{prob} on Time (h) Effect of τ_{prob} on Result #

Figure 8: Scalability Results on Element-Level Uncertain Model

the increment of parameter m: 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" faced by the naïve method.

6.4 Effectiveness in RFID Trajectory Mining

In this subsection, we evaluate the effectiveness of *ElemU-PrefixSpan* by applying it on the real RFID datasets obtained from the Lahar project [24]. 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 near them, and log the sightings along with their timestamp in a database.

Figure 10(a) shows the structure of the $6th$ floor of the building, where the light blue vertices and edges define connectivity graph of the building, with the vertices being the discrete locations and the edges indicating the location connectivity. We use the Markovian stream data which were inferred from the raw readings. In each time step, each data item is in the form (location, probability). For example, the locations marked by blue circles in Figure 10(a) are those with probability smaller than 5%, while the locations marked by yellow or red circles are those with probability at least 5%. Therefore, the location of a tracking tag in a time step can be regarded as a probabilistic element, and the whole trajectory of the tracking tag can be regarded as an element-level probabilistic sequence.

In our experiments, we use 6 trace sequences on the $3rd$ floor for trajectory pattern mining. The average number of possible locations in a time step is $m = 10$ and the average length of the traces is $\ell = 690$. It takes 9276.81 seconds for *ElemU-PrefixSpan* to run on the 6 traces with $\tau_{sup} = 6$ and $\tau_{prob} = 90\%$. Note that the long running time is mainly due to the large length of the traces. For each result trajectory pattern, if an element occurs in several consecutive positions in the pattern, we replace the subsequence with just one occurrence of the element.

Figure 10(b) shows a sample result trajectory pattern with 7 elements, whose probability of being frequent is 90.67%. In Figure 10(b), 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 numbers near them. After checking the ground-truth location labels of all the 6 traces, we find that all the traces contain this pattern without Location 3, which is a room entrance/exit event.

We find that the patterns found by *ElemU-PrefixSpan* is accurate in terms of locations on the hallways, although they may not be accurate enough for detecting local events such as entering/exiting a room. The experiments verify that the patterns found by *ElemU-PrefixSpan* is useful for trajectory mining tasks in RFID applications.

We expect that our *U-PrefixSpan* algorithms would also be useful for frequent sequential pattern mining in many other real world applications involving uncertain data.

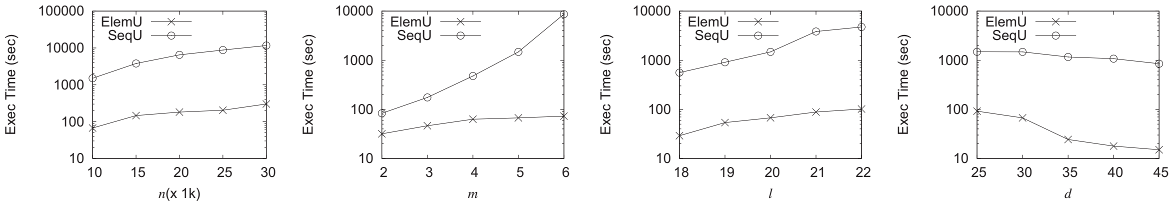
7. CONCLUSION

In this paper, we formulate and study the problem of mining probabilistically frequent sequential patterns (or *p-FSPs*) in uncertain databases. Our study is founded on two uncertain sequence data models that are fundamental for many real-life applications involving uncertain sequence data. 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, which further improve the mining efficiency. Experiments show that our algorithms effectively avoid the problem of "possible world explosion", and the trajectory patterns found by *ElemU-PrefixSpan* in an RFID tracking application are shown to be accurate and useful.

Acknowledgements. This work is partially supported by RGC GRF under grant number HKUST 617610.

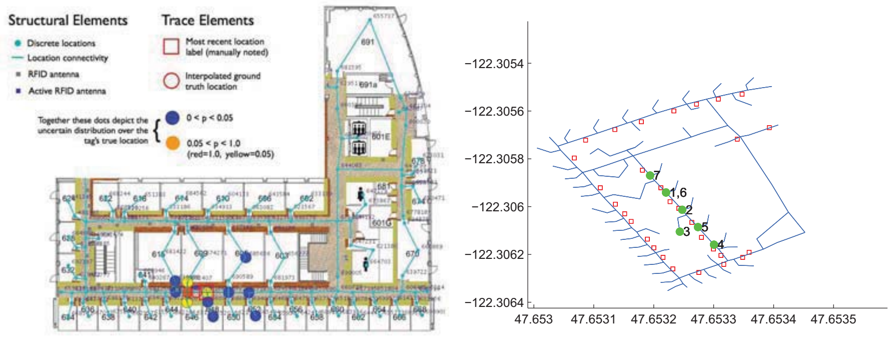
8. REFERENCES

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



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

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



(a) Building Structure on the $6th$ Floor (b) Sample Result Trajectory Pattern on the $3th$ Floor

Figure 10: Results on Real RFID Datasets

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

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

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

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

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

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

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

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

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

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

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

[19] 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.

[20] N. Khoussainova, M. Balazinska and D. Suciu. "PEEX: Extracting Probabilistic Events from RFID Data". In *ICDE*, 2008.

[21] X. Lian and L. Chen. "Set Similarity Join on Probabilistic Data". In *VLDB*, 2010.

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

[23] T. H. Cormen, C. E. Leiseron, R. L. Rivest and C. Stein. "Introduction to Algorithms, 2nd Edition". The MIT Press.

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