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.

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 $[5^\circ, 7^\circ]$), reading $B$ represents $[7^\circ, 9^\circ])$, and reading $C$ represents $[9^\circ, 11^\circ])$, 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, $s_{11}$ 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...
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 \( \alpha \) in an uncertain database can be evaluated as follows: for a sequence-level probabilistic sequence \( s \), if we denote \( \alpha \subseteq s \) to be the event that pattern \( \alpha \) occurs in \( s \), then the expected support of \( \alpha \) in database \( D \) is defined as \( \text{expSup}(\alpha) = \sum_{s \in D} Pr(\alpha \subseteq s) \) 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 \( \text{expSup}(\alpha) \), 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, \( s_{11} \) and \( s_{12} \). The uncertain location \( s_{11} \) has 95\% probability to be \( A \) and 5\% probability to be a misreading (i.e. does not occur), while location \( s_{12} \) 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, \( Pr(pw_3) = Pr(s_{11}) = A \} \times Pr(s_{22}) = B \} \times Pr(s_{21}) = A \} \times Pr(s_{22}) = B \} = 0.9025 \).

Note that the expected support of \( AB \) is \( \text{expSup}(AB) = Pr\{s_{11} = AB \} + Pr\{s_{22} = AB \} = 0.95 \times 0.95 + 1 \times 1 = 1.9025 \), 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
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 \( \beta \) with sequence \( \alpha \). As mentioned in Section 1, \( \alpha \subseteq s \) corresponds to the event that sequence \( \alpha \) occurs as a subsequence of \( s \). We now present some concepts that are necessary for understanding PrefixSpan.

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

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

Definition 2: Given a sequential pattern \( \alpha \) and a sequence database \( D \), the \( \alpha \)-projected database \( D|_\alpha \) is defined to be the set \( \{s|_\alpha \mid s \in D \land \alpha \subseteq s \} \).

Note that if \( \alpha \nsubseteq s \), then the minimal prefix \( \beta \) of \( s \) satisfying \( \alpha \subseteq \beta \) 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 \( \tau_{sup} \)) by recursively checking the frequency of patterns with growing lengths. In each iteration, if the current pattern \( \alpha \) is found to be frequent, it will recur on all the possible patterns \( \alpha' \) constructed by appending one more element to \( \alpha \). PrefixSpan checks whether a pattern \( \alpha \) 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 pseudo-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 similar 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 $\alpha$ in a probabilistic sequence $s$ is given by

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

(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 $\alpha$, denoted by $expSup(\alpha)$, is defined as the sum of the expected probabilities of the presence of $\alpha$ in each of the sequences in the databases.

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

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

![Fig. 4. Probability Distribution of $sup(AB)$](image)

Given a sequence-level or an element-level uncertain sequence database $D$, we denote its possible world space as $PW = \{pw_1, pw_2, \ldots, pw_{|PW|}\}$. We also denote by $sup(\alpha)$ the support of pattern $\alpha$ in a possible world $pw_i \in PW$. Since $pw_i$ is a deterministic instance, $sup(\alpha)$ is simply a count that is equal to $|\{s \in pw_i| \alpha \subseteq s\}|$. Note that each possible world $pw_i$ is associated with an occurrence probability $Pr(pw_i)$, and therefore, given a pattern $\alpha$, each possible world $pw_i$ contributes to a pair $(sup(\alpha), Pr(pw_i))$. 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 $\alpha$, 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:

$$f_\alpha(c) = \sum_{pw_i \in PW \text{ s.t. } sup(\alpha)|c} Pr(pw_i).$$

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 $\tau_{prob}$ and a support threshold $\tau_{sup}$, pattern $\alpha$ is probabilistically frequent (or $(\tau_{sup}, \tau_{prob})$-frequent) iff

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

(2)

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

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

(3)

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

- **R1** CatPrune. Let us define $cnt(\alpha) = |\{s \in D| \alpha \subseteq s\}|$, then pattern $\alpha$ 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 $\alpha$ is not $(\tau_{sup}, \tau_{prob})$-frequent if $expSup(\alpha) < \tau_{sup} \times \tau_{prob}$.

**Proof:** According to Markov’s inequality, $expSup(\alpha) < \frac{\tau_{sup}}{\tau_{prob}}$ implies

$$Pr\{sup(\alpha) \geq \tau_{sup}\} \leq \frac{expSup(\alpha)}{\tau_{sup}} < \tau_{prob}.$$  

- **R3** ExpPrune. Let $\mu = expSup(\alpha)$ and $\delta = \frac{\tau_{sup} - \mu - 1}{\tau_{prob}}$. When $\delta > 0$, pattern $\alpha$ is not $(\tau_{sup}, \tau_{prob})$-frequent if

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

**Proof:** According to Chernoff Bound, we have

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

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

CatPrune 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 $\alpha$ 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 $\alpha$-projected (uncertain) database $D|\alpha$ is equivalent to evaluating $f_\alpha(c)$ on $D$, since $\forall s \notin D|\alpha$, $Pr(\alpha \subseteq s) = 0$. 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 \( f_c^{D(\alpha)}(c) \).

We now consider how to obtain \( f_S^\alpha(c) \) from \( f_S^{\alpha_1}(c) \) and \( f_S^{\alpha_2}(c) \). Let us denote \( sup\sup(\alpha) \) to be the support of \( \alpha \) on \( S \). Note that \( sup\sup(\alpha) \) is a random variable, and \( sup\sup(\alpha) \) and \( sup\sup(\alpha) \) are independent. Obviously, \( sup\sup(\alpha) = sup\sup(\alpha) + sup\sup(\alpha) \), and \( f_S^\alpha(c) \) can be computed by the following formula:

\[
f_S^\alpha(c) = \sum_{i=0}^{c} f_S^{\alpha_1}(c) \times f_S^{\alpha_2}(c - i).
\] (4)

According to Equation (4), \( f_S^\alpha \) is the convolution of \( f_S^{\alpha_1} \) and \( f_S^{\alpha_2} \). Thus, \( f_S^\alpha \) can be computed from \( f_S^{\alpha_1} \) and \( f_S^{\alpha_2} \) 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 naively evaluating Equation (4) for all \( c \), which takes \( O(n^2) \) time.

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

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

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

\[
1 - F_{\alpha}^{S_1}(\tau_{sup} - 1) = Pr\{sup\sup_1(\alpha) \geq \tau_{sup}\} \geq \tau_{prob}. \tag{5}
\]

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

\[
F_{\alpha}^{S}(\tau_{sup} - 1) = \sum_{i=0}^{\tau_{sup}} f_{\alpha}^{S}(i) \times F_{\alpha}^{S}(j).
\]

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

\[
\text{Algorithm 1 PMFCheck(vec}_\alpha) \text{.}
\]

\textbf{Input:} probability vector: \( vec_\alpha \)

\textbf{Output:} mark of frequentness: \( tag \); pmf: \( f_\alpha \)

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 \( |vec_\alpha^1| = \lfloor n/2 \rfloor \) and \( |vec_\alpha^2| = \lceil n/2 \rceil \).
5: if \( tag_1 = \text{TRUE} \) then
6: return \( (\text{TRUE}, \emptyset) \)
7: \( (tag_2, f_\alpha^1) \leftarrow \text{PMFCheck(vec}_\alpha^1) \)
8: if \( tag_2 = \text{TRUE} \) then
9: return \( (\text{TRUE}, \emptyset) \)
10: \( f_\alpha \leftarrow \text{convolution}(f_\alpha^1, f_\alpha^2) \)
11: return \( (1 - F_{\alpha}(\tau_{sup} - 1) \geq \tau_{prob}, f_\alpha) \)

PMFCheck partitions \( vec_\alpha \) into two halves: \( vec_\alpha^1 \) and \( vec_\alpha^2 \) respectively as the first half \( S_1 \) and the second half \( S_2 \) of \( S \) (Line 4). If \( \alpha \) is found to be \((\tau_{sup}, \tau_{prob})\)-frequent in either half (Lines 6 and 9), PMFCheck returns \( \text{TRUE} \) 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. \( f_\alpha^1 \) and \( f_\alpha^2 \)), to compute the pmf of \( \alpha \) in \( S \) in Line 11. After obtaining \( f_\alpha \), we can check whether \( \alpha \) 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 \( f_\alpha(0) = Pr\{sup(\alpha) = 0\} = Pr(\alpha \not\subseteq s_1) \) and \( f_\alpha(1) = Pr\{sup(\alpha) = 1\} = Pr(\alpha \subseteq s_1) \).

\textbf{Complexity Analysis:} Let \( T(n) \) be the running time of PMFCheck on input \( vec_\alpha \) 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 \( T(n) = O(n\log^2 n) \).

\textbf{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.

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

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

According to Property 1, we can stop growing \( \alpha \), once we find that \( \alpha \) is probabilistically infrequent.

\section{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.
<table>
<thead>
<tr>
<th>Seq. Instance</th>
<th>Prob.</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s_i = ABCBC$</td>
<td>0.3</td>
</tr>
<tr>
<td>$s_2 = BABCB$</td>
<td>0.2</td>
</tr>
<tr>
<td>$s_3 = AB$</td>
<td>0.4</td>
</tr>
<tr>
<td>$s_4 = BC$</td>
<td>0.1</td>
</tr>
</tbody>
</table>

\[ A \iff \begin{align*}
    s_i &= _ABCBC \\
    s_i &= _BC \\
    s_i &= _B \\
    s_i &= _
\end{align*} \]

(b) $s_j$

Pr \{ $A \subseteq s_j$ \} = Pr \{ $AB \subseteq s_j$ \} = 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 $\alpha$, we now discuss how to obtain the $\alpha$-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 $\alpha \nsubseteq s_{ij}$), 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 $s_{ij}|\alpha$ as 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 $s_{ij}|\alpha$ 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 $s_i|AB = \{(s_1|AB, 0.3), (s_2|AB, 0.2), (s_3|AB, 0.4)\}$ where, for example, $s_1|AB = <2, s_{i1}>$.

Conceptually, the $\alpha$-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 $\alpha$ by (1) computing $vec_{\alpha}(T) = Pr\{\alpha \subseteq s_i\}$ for each projected probabilistic sequence $s_i|\alpha \in D|\alpha$, and then (2) determining the result by invoking PMFCheck(vec) (Algorithm 1).

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

\[
Pr\{\alpha \subseteq s_i\} = \sum_{s_{ij} \subseteq s_i} Pr\{\alpha \subseteq s_{ij} | s_{ij} \text{ occurs as } s_{ij}\} \times Pr(s_{ij}) = \sum_{s_{ij}|\alpha \in s_i|\alpha} Pr(s_{ij}).
\] (6)

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

**Algorithm 2 Prune($T|\alpha, D|\alpha$).**

Input: element table $T|\alpha$, projected probabilistic database $D|\alpha$  Output: element table $T|\alpha$

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

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

Recall that in PrefixSpan, in each recursive iteration, if the current pattern $\alpha$ is frequent, we grow $\alpha$ by appending to it one element $e$ to obtain a new pattern $\alpha e$, and then recursively checking the frequentness of $\alpha e$. 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 $\alpha e$ $(\tau_{sup}, \tau_{prob})$-frequent.

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

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

**Proof:** Let $\beta = \alpha \gamma$. For any element $e \notin T|\alpha$, $\alpha e$ is not $(\tau_{sup}, \tau_{prob})$-frequent, and since $\alpha e \subseteq \alpha \gamma e = \beta e$, $\beta e$ 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 $\alpha$ 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 $\ell$ on $D|\alpha e$ is equivalent to checking them with pattern $\alpha e \ell$ on $D$, since for any probabilistic sequence $s_i$, whose $\alpha e$-projected sequence does not exist in $D|\alpha e$, $Pr\{\alpha e \ell \subseteq s_i\} = 0$.

**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
Not has no influence on completely unnecessary, since whether $C$ sequence
is in the suffix $s\ell$, PrefixSpan mainly lie in two aspects: (1) sequence pro-
earrow
s\ell, T\ell|\alpha\rangle$.

PrefixSpan recursively performs pat-
tern growth from the previous pattern $\alpha$, 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 $s|\alpha \in D|\alpha$, we compute $Pr\{\beta \subseteq s\}$ as $pr(s|\alpha)$ in Lines 3–9, and if $Pr\{\beta \subseteq s\} > 0$, we add $s|\beta$ (con-
ructed from $s|\alpha$) into $D|\beta$ and append this probability to vec$\alpha$ (Lines 10–12), which is used to determine whether $\beta$ is $(\tau_{sup}, \tau_{prob})$-frequent by invoking PMFCheck(vec$\beta$) in Line 13.

To compute $Pr\{\beta \subseteq s\}$ using Equation (6), we first initialize $pr(s|\alpha)$ to 0 (Line 3). Whenever we find that $s|\beta$ is recursively performed pat-
tern growth from the previous pattern $\alpha$, 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 $s|\alpha \in D|\alpha$, we compute $Pr\{\beta \subseteq s\}$ as $pr(s|\alpha)$ in Lines 3–9, and if $Pr\{\beta \subseteq s\} > 0$, we add $s|\beta$ (con-
ructed from $s|\alpha$) into $D|\beta$ and append this probability to vec$\alpha$ (Lines 10–12), which is used to determine whether $\beta$ is $(\tau_{sup}, \tau_{prob})$-frequent by invoking PMFCheck(vec$\beta$) in Line 13.

To compute $Pr\{\beta \subseteq s\}$ using Equation (6), we first initialize $pr(s|\alpha)$ to 0 (Line 3). Whenever we find that $s|\beta$ is recursively performed pat-
tern growth from the previous pattern $\alpha$, 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 $s|\alpha \in D|\alpha$, we compute $Pr\{\beta \subseteq s\}$ as $pr(s|\alpha)$ in Lines 3–9, and if $Pr\{\beta \subseteq s\} > 0$, we add $s|\beta$ (con-
ructed from $s|\alpha$) into $D|\beta$ and append this probability to vec$\alpha$ (Lines 10–12), which is used to determine whether $\beta$ is $(\tau_{sup}, \tau_{prob})$-frequent by invoking PMFCheck(vec$\beta$) in Line 13.

To compute $Pr\{\beta \subseteq s\}$ using Equation (6), we first initialize $pr(s|\alpha)$ to 0 (Line 3). Whenever we find that $s|\beta$ is recursively performed pat-
tern growth from the previous pattern $\alpha$, 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 $s|\alpha \in D|\alpha$, we compute $Pr\{\beta \subseteq s\}$ as $pr(s|\alpha)$ in Lines 3–9, and if $Pr\{\beta \subseteq s\} > 0$, we add $s|\beta$ (con-
ructed from $s|\alpha$) into $D|\beta$ and append this probability to vec$\alpha$ (Lines 10–12), which is used to determine whether $\beta$ is $(\tau_{sup}, \tau_{prob})$-frequent by invoking PMFCheck(vec$\beta$) in Line 13.

To compute $Pr\{\beta \subseteq s\}$ using Equation (6), we first initialize $pr(s|\alpha)$ to 0 (Line 3). Whenever we find that $s|\beta$ is recursively performed pat-
tern growth from the previous pattern $\alpha$, 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 $s|\alpha \in D|\alpha$, we compute $Pr\{\beta \subseteq s\}$ as $pr(s|\alpha)$ in Lines 3–9, and if $Pr\{\beta \subseteq s\} > 0$, we add $s|\beta$ (con-
ructed from $s|\alpha$) into $D|\beta$ and append this probability to vec$\alpha$ (Lines 10–12), which is used to determine whether $\beta$ is $(\tau_{sup}, \tau_{prob})$-frequent by invoking PMFCheck(vec$\beta$) in Line 13.

To compute $Pr\{\beta \subseteq s\}$ using Equation (6), we first initialize $pr(s|\alpha)$ to 0 (Line 3). Whenever we find that $s|\beta$ is recursively performed pat-
tern growth from the previous pattern $\alpha$, 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 $s|\alpha \in D|\alpha$, we compute $Pr\{\beta \subseteq s\}$ as $pr(s|\alpha)$ in Lines 3–9, and if $Pr\{\beta \subseteq s\} > 0$, we add $s|\beta$ (con-
ructed from $s|\alpha$) into $D|\beta$ and append this probability to vec$\alpha$ (Lines 10–12), which is used to determine whether $\beta$ is $(\tau_{sup}, \tau_{prob})$-frequent by invoking PMFCheck(vec$\beta$) in Line 13.
compute $Pr_{\text{pos}}(s_i, X)$ according to Equation (7):

$$Pr_{\text{pos}}(s_i, \alpha e) = \sum_{k < \text{pos}} \left[ Pr(ek(s_i, \alpha)) \times Pr\{s_i[\text{pos}] = e\} \right] \prod_{k < j < \text{pos}} \left(1 - Pr\{s_i[j] = e\}\right).$$  \tag{8}$$

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

Recall that in the prefix-projection method of PrefixSpan, the projected sequence $s_i|\alpha$ of a deterministic sequence $s$ is obtained by removing from $s$ its minimal prefix containing $\alpha$. Therefore, the projected sequence $s_i|\alpha$ of an element-level probabilistic sequence $s_i$ can be represented by a set of disjoint events $e_k(s_i, \alpha), 0 < k \leq \text{len}(s_i)$, where $\text{len}(s_i)$ 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 $\alpha$ grows from $\emptyset$ to $B$. For ease of presentation, we use $e_{pos}(s_i, \alpha)$ to denote the suffix of $s_i$ given event $e_{pos}(s_i, \alpha)$. Since $e_{pos}(s_i, \emptyset) = s_i[1]s_i[2]s_i[3]s_i[4]$, 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 $s_i|\emptyset$, as shown in the second (middle) table in Figure 6(c):

- $e_1(s_i, B) = \{s_i[1] \neq B \wedge s_i[2] \neq B \wedge s_i[4] = B\}$.
  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, $Pr(e_2(s_i, B)) = Pr(e_0(s_i, \emptyset)) \times (1 - Pr\{s_i[1] = B\}) \times Pr\{s_i[2] = B\} = 0.14$.

- $e_3(s_i, B) = \{s_i[1] \neq B \wedge s_i[2] \neq B \wedge s_i[4] = B\}$. In this case, $Pr(e_3(s_i, B)) = Pr(e_0(s_i, \emptyset)) \times (1 - Pr\{s_i[1] = B\}) \times Pr\{s_i[2] = B\} = 0.0504$.

For the case when $\alpha$ grows from $B$ to $BA$, we focus on the event $e_2(s_i, B)$ of $s_i|B$. Since $e_2(s_i, B) = s_i[3]s_i[4]$, 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 $s_i|\alpha$, we project it onto a new projected sequence $s_i|\beta$ 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 $pos_s + 1$ (Line 4), i.e. $s_r$. We also attach a variable $\text{accumulate}$ 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 $s_i|\beta$ from $s_i|\alpha$, we check all the events $r = <pos_r, Pr(r)>$ of $s_i|\alpha$, 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 $r = (pos_r, pr_r) \in D$ 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: $(pos_{last}, pr_{last})$ ← 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 $\Delta$ 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 $\Delta$ 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 $\Delta$ (Lines 16–17).

When $s|\alpha$ has $k$ events, and each event $e_i$ has suffix of length $\ell_i$, 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 $\ell$ can have no more than $\ell$ events.”

The correctness of this statement is established by the fact that there are at most $\ell$ values for pos. This result 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 is exponential to $\ell$.

Computation of $Pr\{\alpha \supseteq s_i\}$. Consider pattern $\beta = \alpha e$. Suppose that the projected probabilistic sequence $s|\alpha$ 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 $\alpha \subseteq s_i$, it follows that $\beta \not\supseteq s_i$ if and only if $e$ does not occur in any of the elements in the suffix $s_i[\text{pos}+1, \ldots, \text{len}(s_i)]$ (i.e. $s_i|\{e_{pos}(s_i, \alpha)\}$). Thus, it follows that

$$Pr\{\beta \not\supseteq s_i\} = \sum_{pos} Pr\{\beta \not\supseteq s_i|e_{pos}(s_i, \alpha)\} \times Pr(e_{pos}(s_i, \alpha))$$

So we now have

$$Pr\{\beta \not\supseteq s_i\} = 1 - Pr\{\beta \supseteq s_i\}$$

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 $Pr\{\beta \supseteq s_i\}$ for all $s_i|\beta \in D|\beta$, we check the $(\tau_{\text{sup}}, \tau_{\text{prob}})$-frequentness of $\beta$ 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 frequence of patterns, reducing the time
Algorithm 6 ElemU-PrefixSpan(αe, D|α, T|α)

Input: pattern αe, projected probabilistic database D|α, element table T|α

1: vecα ← ∅
2: for each projected sequence s|α ∈ D|α do
3:   Find its corresponding sequence s ∈ D
4:   accum ← 0
5: for each (pos, pr) ∈ s|α do
6:   accum ← accum + pr × ElemProb(s, pos, e)
7: Append accum to vecα
8: (tag, fα) ← PMFCheck(vecα)
9: if tag = TRUE then
10: output α
11: D|α ← Project(D|α, e)
12: T|α ← Prune(T|α, D|α)
13: for each element ℓ ∈ T|α do
14:   ElemU-PrefixSpan(αℓ, D|α, T|α)
15: Free D|α and T|α from memory

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

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

The underlying idea of Equation 14 is to use numerical methods and grows the patterns whose expected support \( \mu \) is greater than \( \mu_m \).

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 \( \alpha \) (i.e. \( \text{expSup}(\alpha) \)) 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 \( \delta \) of \( \alpha \) in its projected database is given by

\[
\delta = \sqrt{\sum_{i=1}^{n} Pr\{\alpha \subseteq s_i\}(1 - Pr\{\alpha \subseteq s_i\})},
\] (15)

and, therefore the NA approximation of the probabilistic frequentness of \( \alpha \) is given by

\[
Pr\{\text{sup}(\alpha) \geq \tau_{sup}\} \approx 1 - G(\tau_{sup} - \frac{1}{2} - \frac{\mu}{\delta})
\] (16)

where \( G(t) = \int_{-\infty}^{t} e^{-x^2/2} dx \) and \( \tau_{sup} - \frac{1}{2} - \frac{\mu}{\delta} \) 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

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

\[
\frac{\partial}{\partial t}(1 - G(t)) = -\frac{\partial}{\partial t} G(t) = -\int_{-\infty}^{t} e^{-x^2/2} dx \\
= -e^{-t^2/2} \\
\leq 0,
\]

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

\[
1 - G(t_m) = \tau_{prob}.
\] (18)

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

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

\[
Pr\{\text{sup}(\alpha) \geq \tau_{sup}\} = 1 - Pr\{\text{sup}(\alpha) \leq \tau_{sup}-1\},
\] (11)

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

Let us denote the Poisson 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 Poisson cumulative distribution \( F(\mu, \tau_{sup} - 1) \). This algorithm checks \( \alpha \) in the projection database by

\[
Pr\{\text{sup}(\alpha) \geq \tau_{sup}\} \approx 1 - F(\mu, \tau_{sup} - 1) \geq \tau_{prob},
\] (12)

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

\[
\mu = \sum_{i=1}^{n_{\alpha}} Pr\{\alpha \subseteq s_i\},
\] (13)

with \( n_{\alpha} \) being the size of \( D|\alpha \).
Finally, in Section 7.4, we successfully apply effective algorithms and their approximation algorithms, using large datasets in Sections 7.1 and 7.2. We define methods as approximate precision in this paper, since the approximate approximation methods. For brevity, we only report the approximation algorithms.

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 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

\[
\text{precision} = \frac{|FSP_{\text{app}} \cap FSP|}{|FSP_{\text{app}}|},
\]

where \(FSP\) is the set of patterns obtained by U-PrefixSpan algorithms, the patterns in \(FSP\) are taken as the ground truth, and \(FSP_{\text{app}}\) 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 \((n, m, \ell, d)\), 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 \(n, m, \ell, d\), we also have two threshold parameters: the support threshold \(\tau_{\text{sup}}\) and the probability threshold \(\tau_{\text{prob}}\).

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
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 $\tau_{sup}$ varies. The NA-SeqU algorithm has good approximate precision on $\tau_{qua}$ 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 $\alpha$.

### 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 $(n, m, \ell, d)$, 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 $n$, $m$, $\ell$, $d$, $\tau_{sup}$, and $\tau_{prob}$. 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 \((n, m, \ell, d) = (10k, 5, 20, 30)\). 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 \(n, \ell\) 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 \(\tau_{sup}\) and \(\tau_{prob}\). 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 \(\alpha\) 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.

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


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

(a) Effect of $n$
(b) Effect of $m$
(c) Effect of $\ell$
(d) Effect of $d$

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


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


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


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.