Knowl Inf Syst (2017) 51:821–850
DOI 10.1007/s10115-016-0977-1

REGULAR PAPER

Sequential pattern mining in databases with temporal uncertainty

Jiaqi Ge[1,4] · Yuni Xia[1] · Jian Wang[2] · Chandima Hewa Nadungodage[1] · Sunil Prabhakar[3]

Received: 30 April 2015 / Revised: 20 June 2016 / Accepted: 26 July 2016 /
Published online: 30 July 2016

Abstract Temporally uncertain data widely exist in many real-world applications. Temporal uncertainty can be caused by various reasons such as conflicting or missing event timestamps, network latency, granularity mismatch, synchronization problems, device precision limitations, data aggregation. In this paper, we propose an efficient algorithm to mine sequential patterns from data with temporal uncertainty. We propose an uncertain model in which timestamps are modeled by random variables and then design a new approach to manage temporal uncertainty. We integrate it into the pattern-growth sequential pattern mining algorithm to discover probabilistic frequent sequential patterns. Extensive experiments on both synthetic and real datasets prove that the proposed algorithm is both efficient and scalable.

Keywords Uncertain databases · Sequential pattern mining · Temporal uncertainty

1 Introduction

Sequential pattern mining (SPM) is an important data mining application which provides inter-transactional analysis for timestamped data. SPM is often applied to discover patterns in sequence databases which are widely used to model shopping sequences, medical syndromes and treatments, natural disasters, stock markets, and so on. For example, supermarkets collect customer purchase histories and use SPM methods to reveal customer purchasing patterns,

✉ Jiaqi Ge
jge@expedia.com

1 Department of Computer and Information Science, Indiana University Purdue University Indianapolis, Indianapolis, IN, USA

2 School of Electronic Science and Engineering, Nanjing University, Nanjing, China

3 Department of Computer Science, Purdue University, West Lafayette, IN, USA

4 Expedia Inc., Chicago, IL 60661, USA

which can be informally represented as: If a customer buys an item A, he/she will buy another item B within a certain time period.

Most of the existing work fundamentally relies on an assumption that event occurrences are either in a total order [17,21,23,24,31,32] or in a strict partial order [4,14,18,27,33] based on precise event occurring time. However, this assumption fails in many real-world applications because event time can be inaccurate or even unknown for a variety of reasons:

- *The exact time of an event is often unknown.* For instance, *Yahoo! finance* collects the highest and lowest prices of stocks every day, from which we can detect an event such as "price grows more than 8 % in one day." However, the exact time of this event is unknown because the system does not record when the stocks are traded at the highest or lowest prices. And this type of event is usually assumed to occur equally likely at any time during the day.
- *Temporal uncertainty arises because of granularity mismatch.*

Example 1.1 In a GPS monitoring system, a handheld GPS device $\mathcal{G}_a$ may update its position every 10 min while a GPS $\mathcal{G}_b$ mounted on a fast-moving vehicle might report the position every 5 s. Then, an event e_a reported by $\mathcal{G}_a$ can occur anywhere in a 10-min period (e.g., 09:00:00–09:10:00), and an event e_b reported by $\mathcal{G}_b$ occurs randomly within a 5-s period (e.g., 09:01:00–09:01:05).

It is difficult to determine whether e_a occurs before or after e_b, because the temporal relationship between two events defined in different granularities becomes uncertain and cannot be modeled by either a total or partial order.

- *Temporal uncertainty is added to protect privacy and confidentiality.* Precise time information in monitoring data is often not released if there is a potential to identify individuals. Obfuscation techniques deliberately degrade temporal information, using uncertainty to protect privacy.

In traditional temporal databases, valid-time indeterminacy is modeled by a sequence of consecutive *chronons* as $t_0, t_1, \ldots, t_N$, where a chronon is the smallest time unit in the system [11]. However, this model becomes inefficient when data are collected under different timescales. In Example 1.1, if the chronon is set to be a *second*, the uncertain time of an event reported by the on-vehicle device $\mathcal{G}_B$ may be represented by five consecutive timestamps (e.g., $t_1, t_2, \ldots, t_5$) with equal probabilities, while the timestamp of an event reported by the handheld device $\mathcal{G}_A$ is represented by a sequence of hundreds of timestamps (e.g., $t_1, \ldots, t_{600}$), which is neither efficient nor convenient.

Instead of relying on chronons, we propose our *temporally uncertain model* to efficiently represent uncertain event times by *random variables*. If the timestamp of an event equally likely occurs at any point in a time period, it is reasonable to model it by a uniform probability density function (pdf). Furthermore, for any arbitrary shaped pdf, we approximate it by discrete probability mass functions (pmf) using sampling and histogramming techniques.

In temporally uncertain sequence databases, it is much more difficult to identify frequent sequential patterns because orders of events are uncertain. We adopt *possible world semantics* from probabilistic databases [6,7,15] to interpret our uncertain model by a set of certain databases. However, the number of certain databases derived from a temporally uncertain database is infinite because we use continuous pdf to represent data uncertainty in our model. Besides, aggregating probabilities associated with possible worlds requires the computation of multivariable integration, which brings efficiency and scalability challenges to the uncertain SPM problem.

Another challenge comes from integrating gap constraints into SPM process. A gap constraint requires the pattern appears frequently in the database such that the time difference between every two adjacent events must be longer or shorter than a given gap [20]. Incorporating gap constraints (e.g., minimum gap and maximum gap) in SPM can help to mine user-interested patterns; however, it makes the management of temporal uncertainty more complicated. For example, suppose event A occurs equally likely in the range of [1, 5] and event B occurs within [6, 10], then it is certain that B occurs after A; however, after applying minimal gap ($g_l = 2$) and maximal gap ($g_h = 5$), it becomes complicated to check whether B occurs after A with satisfying gap constraints or not. And we will propose a solution of this problem in Sect. 5.

In this paper, we address the SPM problem in temporally uncertain databases. And our major contributions include:

(1) Besides representing temporal uncertainty by uniformly distributed random variables in our preliminary work [12], we use a discrete probability mass function (pmf) to approximate the distribution of any arbitrary shaped uncertainty and propose a general solution based on this model. This approximation is proved to be effective and efficient, according to the experimental results.
(2) We develop a novel approach to compute temporal uncertainty during the SPM process which efficiently calculate multivariable pdfs/pmfs for uncertain timestamps.
(3) We incorporate gap constraints to mine sequential patterns with user-specified criteria.
(4) We develop three new pruning techniques to speed up the computation.
(5) We conduct extensive experiments on both synthetic and real datasets to test and prove the efficiency and scalability of the proposed algorithms. We also apply our method on a real-world stock market dataset and analyze the results via visualization techniques.

2 Related works

2.1 Traditional sequential pattern mining

Sequential pattern mining problem in traditional deterministic databases has attracted a lot of attention. There have been many algorithms on efficient sequential pattern mining and its applications [3,5,9,13,19,22,28,29]. In general, sequential pattern mining algorithms can be categorized into three classes: (1) Apriori-like algorithm with horizontal data format, e.g., GSP [22]; (2) Apriori-like algorithm with vertical data format, like SPADE [29]; and (3) projection-based pattern-growth algorithm, like PrefixSpan [19].

PrefixSpan is proved to be more efficient than other Apriori-like algorithms such as GSP due to its prefix-projection technique. Here, we briefly review the framework of PrefixSpan, which is related to our proposed algorithms. We first present the following definitions.

Definition 2.1 A *sequential pattern* $s = \langle s_1, s_2, \ldots, s_n \rangle$ is a temporally ordered sequence of itemsets, where $s_i \in s$ is called an *element* of s.

Definition 2.2 A sequential pattern $\alpha = \langle a_1, a_2, \ldots, a_n \rangle$ is a *sub-sequence* of a sequence $\beta = \langle b_1, \ldots, b_m \rangle$, denoted by $\alpha \sqsubseteq \beta$, if there exist n integers $1 \leq k_1 < \cdots < k_n \leq m$ such that $a_i \subseteq b_{k_i}$.

For example, $\langle (a)(b) \rangle$ is a subsequence of $\langle (ab)(bc) \rangle$. Without loss of generality, items in an element are assumed to be ordered alphabetically.

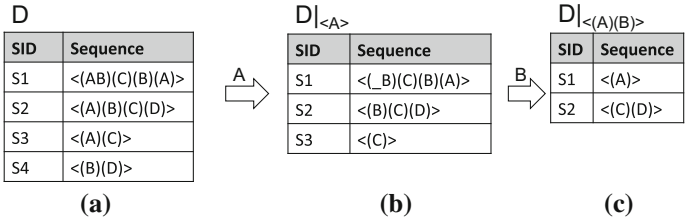

Fig. 1 Example of database projection in PrefixSpan

Definition 2.3 A sequential pattern $\alpha = \langle a_1, \ldots, a_m \rangle$ is a *prefix* of a sequence $\beta = \langle b_1, \ldots, b_n \rangle$ $(m \leq n)$ if (1) $a_i = b_i, \forall i \in [1, m-1]$; (2) $a_m \subseteq b_m$ and all items in $b_m - a_m$ are alphabetically larger than those in a_m.

For example, both $\langle (a) \rangle$ and $\langle (ab)(b) \rangle$ are prefixes of $\langle (ab)(bc) \rangle$. And for ease of presentation, we denote $\alpha\beta$ to be a sequence resulted from appending sequence β to α.

Definition 2.4 Given a 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 $\alpha \sqsubseteq \beta$. And the *α-projected database* $D|_\alpha$ is defined to be a collection of projected sequences $\{s_\alpha | s \in D \wedge s_\alpha \neq \phi\}$.

Consider the sequence database D shown in Fig. 1a. The $\langle A \rangle$-projected database $D|_{\langle A \rangle}$, shown in Fig. 1b is built by projecting item A from each sequence. For example, $\langle (\_B)(C)(B)(A) \rangle$ is the suffix after removing prefix $\langle A \rangle$ from sequence s_1. Notice that $(\_B)$ means that the last element in the prefix, which is A here, together with item $\_B$, form one element (AB). And we add the notation _ ahead to distinguish it from a regular item B. Suppose B is a frequent item in $D|_{\langle A \rangle}$, then PrefixSpan recursively constructs $\langle (A)(B) \rangle$-projected database, as shown in Fig. 1c. For example, the projected sequence $s_1|_{\langle (A)(B) \rangle} = \langle A \rangle$ is generated by projecting prefix $(A)(B)$ from s_1. PrefixSpan adopts a depth-first strategy to find frequent sequential patterns with growing lengths and it stops when all frequent patterns are found.

2.2 Sequential pattern mining in uncertain data

Interval-based SPM An event that does not occur at a time point but lasts for a period of time can be modeled by a time interval which represents the duration of the event. In [4], Allen et al. [14,18,27] introduce thirteen temporal relationships between two time intervals, and many algorithms have been designed to mine Allen's relations from data with interval-based timestamps. However, events in these algorithms still have precise timestamps and are usually imposed to have a strict partial order. In contrast, our work deals with events that occur at a time point but with uncertain timestamps. When a partial order is pre-defined, some possible orders of events are not allowed, which will cause the loss of information.

SPM with existential uncertainty Muzammal and Raman [17] proposed an SPM algorithm in probabilistic database using the expected support as the measurement of pattern frequentness. Zhao et al. [31,32] measure pattern frequentness in possible world semantics and propose a pattern-growth uncertain SPM algorithm. Sun et al. [24] use approximation with probabilistic guarantee to improve the efficiency of mining uncertain frequent itemsets. Dynamic

programming is used to mine frequent serial episodes in an uncertain sequence [26] and probabilistic spatial-temporal frequent sequential patterns [16]. All these methods are designed for dealing with existential uncertainty in databases with accurate timestamps.

Indeterminate temporal database Dyreson and Snodgrass [11] introduced indeterminate semantics which models valid-time indeterminacy by a set of consecutive timestamps with equal probabilities. Zhang et al. [30] proposed a pattern recognition algorithm in temporal uncertain streams, and pattern queries in temporal uncertain sequences are studied in [33]. Our work distinguishes from the above in that we use random variables to represent uncertain timestamps. A random variable is more flexible and efficient in modeling data collected from different scales. Meanwhile, the above work focused on matching patterns in one sequence, while our work addresses mining patterns from a large number of sequences. In [23], Sun et al. introduce one type of uncertain event into Apriori-like sequential pattern discovery with only considering the total order of events; however, every event in our model is uncertain, and we do not assume any total or partial orders of events. In addition to our preliminary work [12], we extend the model of temporal uncertainty from uniform distributions to any arbitrary shaped distributions and propose a general solution based on this model; we develop new pruning techniques to help improve efficiency; we also apply our algorithm to a real stock dataset and analyze the results via visualization techniques.

3 Problem statement

3.1 Temporally uncertain model

The uncertain model applied in this paper is based on temporally uncertain events.

Definition 3.1 A *temporally uncertain event* is an event whose occurance time is uncertain and can be represented by a random variable.

Definition 3.2 An *uncertain sequence* is a list of temporally uncertain events. An *uncertain sequence database* is a collection of uncertain sequences.

We represent a temporally uncertain event by $e = \langle \text{sid}, \text{eid}, T, I \rangle$, where sid is the sequence-id, eid is the event-id, and I is an itemset that describes the content of event e. T is a random variable representing the uncertain event time of e. And in this paper, we consider the following two models of temporal uncertainty:
(1) T is modeled by a uniform probability density function (pdf) in a range, denoted by $T \sim U[t^-, t^+]$; (2) T is modeled by a discrete probability mass function (pmf), denoted by $\{T|t_1 : p_1, \ldots, t_n : p_n\}$, where p_i is the probability that $T = t_i$.

We denote an event with sid $= i$ and eid $= j$ by e_{ij} and denote its uncertain event time by T_{ij}. Table 1 shows two examples of temporally uncertain databases in a logging system which monitors a large number of distributed computers. In Table 1(a), each sequence is a list of events that are detected by a server. A server pings its clients periodically to test connections. For example, the event $e_{11} = \{\text{Client A} : \text{Connection lost}\}$ is detected because the server does connect the client at time 50 but fails to reach it at time 60. And the occurring time of e_{11} is equally likely to be at anywhere between 50 and 60, which is naturally modeled by the uniform distribution $T_{11} \sim U(50, 60)$. For simplicity's sake, we write $U[t^-, t^+]$ as $[t^-, t^+]$ in Table 1(a).

Table 1 Examples of uncertain sequence databases in logging systems

Sid	Eid	T	I
(a) Uncertain time modeled by uniform pdf			
1	1	[50, 60]	{Client A: Connection Lost}
1	2	[55, 70]	{Client B: Connection established}
2	1	[160, 170]	{Client C: Sleep}
2	2	[165, 175]	{Client D: Connection Lost}
2	3	[180, 190]	{Client E: Wake up}
(b) Uncertain time modeled by discrete pmf			
1	1	{100: 0.2, 101: 0.5, 102: 0.3}	{ssh, http-web}
1	2	{103: 0.2, 104: 0.6, 105: 0.2}	{ftp}
1	3	{107: 0.2, 108: 0.6, 109: 0.2}	{ssh}
2	1	{33: 0.2, 34: 0.5, 35: 0.3}	{buffer-overflow}
2	2	{35: 0.2, 36: 0.6, 37: 0.2}	{ftp, ssh}
2	3	{38: 0.2, 39: 0.6, 40: 0.2}	{smtp-mail, ftp}

Table 1(b) records sequences of actions in a cluster of distributed computers. The exact time of these actions is unknown because of the network latency. For example, suppose the network latency δ_t is a Gaussian noise $\delta_t \sim N(-4, 1)$, then the occurrence time of event e_{11}, which is recorded at time 105, is $T_{11} = 105 + \delta_t$. In real applications, it is usually infeasible to obtain the exact distribution of an arbitrary shaped noise. Therefore, it is more practical to approximate the underlying continuous distribution by a discrete pmf, which can be obtained by sampling and/or histogramming methods. For example, the distribution of T_{11} is approximated by the pmf {100: 0.2, 101: 0.5, 102: 0.3} in Table 1(b).

A sequential pattern $s = \langle s_1, \ldots, s_n \rangle$ is a sequence of itemsets, where s_i is called an *element* of s. The length of a sequential pattern is the sum of number of items for all the elements. For example, $\langle (\textit{buffer-overflow})(\textit{ssh})(\textit{ftp}) \rangle$ in Table 1(b) is a 3-length sequential pattern which corresponds to a Web-based attack followed by copying data from the host computer to remote destination via *ftp*.

3.2 Temporal possible world semantics

An uncertain database D is interpreted by a set of possible worlds under possible world semantics. A temporal possible world is a certain sequence database with point-value timestamps drawn from the pdfs/pmfs of uncertain timestamps.

Pdf-modeled uncertainty A sequence database with pdf-modeled uncertain timestamps derives an infinite number of possible worlds. Figure 2a shows two example possible worlds that are instantiated from the uncertain database in Table 1(a). In Fig. 2a, each event time is certain and is drawn from the corresponding uniform distribution. For example, the event time $t_{11} = 50$ in w_1 is instantiated from $T_{11} \sim U(50, 60)$ in Table 1(a).

The pdf of a possible word w is $f_D(w) = f(\hat{d}_1, \hat{d}_2, \ldots, \hat{d}_n)$, where $\hat{d}_i \in w$ is a certain sequence instantiated from the uncertain sequence d_i. It is widely assumed that uncertain sequences are mutually independent, which is known as the *tuple-level independence* [1,15] in probabilistic databases. Event times are also assumed to be independent [6,8,10,31], which can be justified by the fact that events are usually observed independently in real applications.

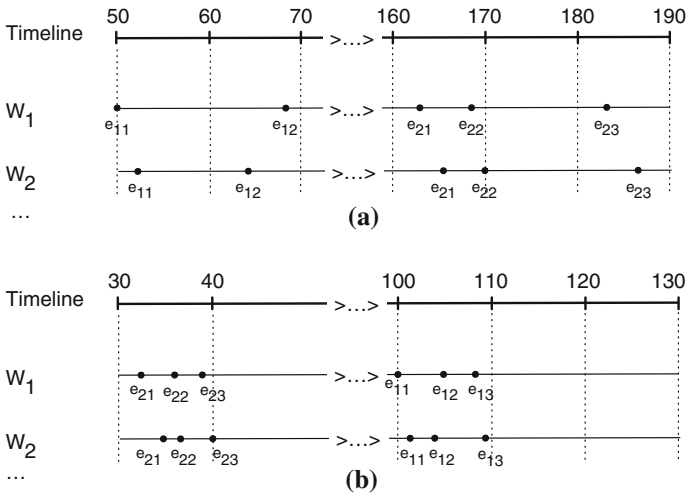

Fig. 2 Examples of possible worlds of the temporal uncertain database. **a** Possible worlds of uncertain database in Table 1(a). **b** Possible worlds of uncertain database in Table 1(b)

With these assumptions, the computation of $f_D(w)$ can be simplified, as shown in Eq. (1).

$$f_D(w) = \prod_{i=1}^{|D|} f(d_i = \hat{d}_i) = \prod_{i=1}^{|D|} \prod_{j=1}^{|d_i|} f_{T_{ij}}(t) \tag{1}$$

Here, $|D|$ is the number of sequences in $D and |d_i|$ is the number of events in sequence d_i. $f_{T_{ij}}(t)$ is the pdf of $T_{ij} \sim U(t^-, t^+)$, as shown in Eq. (2).

$$f_{T_{ij}}(t) = \begin{cases} \frac{1}{t^+ - t^-}, & t \in [t^-, t^+] \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

Pmf-modeled uncertainty A sequence database with discrete temporal uncertainty is interpreted by a finite set of possible words. Figure 2b shows two possible world examples derived from the example in Table 1(b). In the possible world w_1, the event time $t_{11} = 101$ is instantiated from the discrete pdf $\{T_{11}|101: 0.2, 102: 0.5, 103: 0.3\}$ in Table 1(b). With the independence assumptions mentioned above, we can compute the probability mass function (pmf) of w in Eq. (3).

$$f_D(w) = \prod_{i=1}^{|D|} P(d_i = \hat{d}_i) = \prod_{i=1}^{|D|} \prod_{j=1}^{|d_i|} f_{T_{ij}}(t) \tag{3}$$

where $f_{T_{ij}}(t) = P(T_{ij} = t)$ is the probability that T_{ij} equals to t.

3.3 Uncertain SPM problem

In traditional certain database, a sequential pattern s is *supported* by a sequence d, denoted by $s \preceq d$, if and only if it satisfies: (1) there is an occurrence of s in d; (2) this occurrence satisfies gap constraints.

There is an occurrence of a sequential pattern $s = \langle s_1, \ldots, s_n \rangle$ in a sequence $d = \langle e_1, \ldots, e_m \rangle$ if and only there exist n integers $1 \leq k_1 < \cdots < k_n \leq m$ which have

$s_i \subseteq e_{k_i}$. And the ordered set of events $o = \langle e_{k_1}, \ldots, e_{k_n} \rangle$ is so called an occurrence of s in d. For example, let $d = \langle (a)(b)(cd)(ef) \rangle$ and $s = \langle (a)(b)(c) \rangle$, then $o = \{(a)(b)(cd)\}$ is an occurrence of s in d.

A pattern's occurrence satisfies gap constraints if the occurring time of every two adjacent events are longer or shorter than a given gap [20]. Given the minimum gap g_l and the maximum gap g_h, an occurrence $o = \langle e_{k_1}, \ldots, e_{k_n} \rangle$ satisfies the gap constraints if $g_l \leq T_{k_{i+1}} - T_{k_i} \leq g_h$ for $\forall i \in [1, n)$, where T_{k_i} is the timestamp of e_{k_i}.

The support of a pattern s, denoted by $\sup(s)$, is the number of sequences that support it. In a deterministic database, s is *frequent* if $\sup(s) \geq \tau_s$. Here, τ_s is the user-defined minimum threshold. However, the frequentness of s in an uncertain database D is probabilistic. And we define *probabilistic frequent sequential pattern* as follows.

Definition 3.3 A sequential pattern s is a *probabilistic frequent pattern* (p-FSP) if and only if its probability of being frequent is at least τ_p, denoted by $P\big(\sup(s) \geq \tau_s\big) \geq \tau_p$.

Here, τ_p is the user-defined minimum confidence in the frequentness of a sequential pattern. Let A_s be a set of possible worlds in which s is frequent, then $P(\sup(s) \geq \tau_s)$ is the sum of the existential probabilities of all possible worlds in A_s. Depending on the model of temporal uncertainty, it can be computed by Eq. (4a) if temporal uncertainty is pdf-modeled or Eq. (4b) if discrete pmf is adopted.

$$P(\sup(s) \geq \tau_s) = \int_{w \in A_s} f_D(w) \mathrm{d}w \tag{4a}$$

$$P(\sup(s) \geq \tau_s) = \sum_{w \in A_s} f_D(w) \tag{4b}$$

Therefore, the SPM problem in temporal uncertain databases is defined as: *Given thresholds τ_s, τ_p and gap constraints g_l, g_h return all p-FSPs in a temporally uncertain database D.*

Note that directly expanding all possible worlds of D is usually infeasible in practice, because the number of possible worlds can be infinite. And our goal is to efficiently mine p-FSPs without enumerating all possible worlds.

4 USPM algorithm

In this section, we propose our uncertain sequential pattern mining (USPM) algorithms in temporally uncertain databases.

4.1 A depth-first search framework

First of all, we define two types of sequential pattern as follows.

Definition 4.1 An *item-extended pattern* s is a sequential pattern which is generated by appending an item i to the last element of another pattern s', denoted by $s = s' \cup \{i\}$.

Definition 4.2 A *sequence-extended pattern* s is a sequential pattern generated by appending an itemset $\{i\}$ to another pattern s' as its last element, denoted by $s = s' + \{i\}$.

For example, given $s' = \langle (a)(b)(c) \rangle$ and an item d, $s_1 = \langle (a)(b)(cd) \rangle$ is an item-extended pattern of s with item d, while $s_2 = \langle (a)(b)(c)(d) \rangle$ is a sequence-extended pattern.

The anti-monotonicity property of p-FSPs in Lemma 4.1 allows us to prune a sequential pattern if it is extended from a pattern that is not a p-FSP.

Lemma 4.1 *If s is extended from s' and s is a p-FSP, then s' is also a p-FSP.*

Proof In a possible world w, if s is frequent in w, s' is also frequent because $s' \sqsubseteq s$. Thus, $P(\sup(s') \geq \tau_s) \geq P(\sup(s) \geq \tau_s)$. Since s is a p-FSP, we have $P(\sup(s') \geq \tau_s) \geq P(\sup(s) \geq \tau_s) \geq \tau_p$. Therefore, s' is a p-FSP. □

In Algorithm 1, we extend the PrefixSpan framework in uncertain databases. Here, s is a p-FSP in an uncertain database D and $D|_s$ is the uncertain projected database with respect to s; according to Lemma 4.1, if an item i is not probabilistic frequent, a pattern generated by extending s with item i is not a p-FSP; therefore, for each frequent item i in $D|_s$, we generate a candidate pattern s' by extend s with i. Then, we search new patterns by the depth-first strategy. For every candidate pattern s', we build the s'-projected database $D|_{s'}$ in the sub-procedure *Project* and compute the frequentness probability of s' in the sub-procedure *FreqProb*. If s' is a p-FSP, we save it to L and continue to search longer p-FSPs using $D|_{s'}$. Here, L is a set of all p-FSPs, and the details of the procedures *Project* and *FreqProb* are discussed in the following sections.

Algorithm 1: USPM

Input: s: a p-FSP, $D|_s$: s-projected uncertain database
τ_s: minimal support, τ_p: minimal frequentness probability
Output: L: a set of p-FSPs
$L \leftarrow \phi$
foreach *probabilistic frequent item* $i \in D|_s$ **do**
  $s' \leftarrow$ extend s by item i
  $D|_{s'} \leftarrow$ Project(i, $D|_s$) `// build s'-projected database`
  $P(\sup(s') \geq \tau_s) \leftarrow$ FreqProb(s', $D|_{s'}$) `// compute frequentness probability of s'`
  if $P(\sup(s') \geq \tau_s) \geq \tau_p$ **then**
    $L \leftarrow L \cup \{s'\}$
    $L \leftarrow L \cup$ USPM(s', $D|_{s'}$, τ_s, τ_p)
  end
end
return L

4.2 Approximate frequentness probability

In an uncertain database D, the probabilistic support $\sup(s)$ can be represented by a random variable. Since sequences are assumed to be mutually independent, $\sup(s)$ is the sum of n independent random variables, as shown in Eq. (5),

$$\sup(s) = \sum_{i=1}^{n} \sup(s|d_i) \tag{5}$$

where $d_i \in D$ is an uncertain sequence and n is the number of sequences in D. $\sup(s|d_i) \sim \mathcal{B}(1, p_i)$ is a Bernoulli random representing the probabilistic support of s in d_i, and $p_i = P(s \preceq d_i)$ is so called the *support probability* of s in d_i. We will discuss the computation of p_i in Sect. 4.3.

As the sum of n independent but nonidentical Bernoulli trials, $\sup(s)$ follows a Poisson–Binomial distribution. The fast Fourier transform (FFT) technique is adopted to compute the pmf of $\sup(s)$ in $O(n \log n)$ time [31]. In order to further improve the efficiency, we use Gaussian distribution to approximate the underlying Poisson–Binomial distribution. This type of approximation has also been used in frequent patterns mining problems [24,32].

Lemma 4.2 *Let $p_i = P(s \preceq d_i)$, then the overall support $\sup(s)$ in a large uncertain database converges in distribution to a Gaussian random variable, shown as follows:*

$$\sup(s) = \sum_{i=1}^{n} \sup(s|d_i) \xrightarrow{n\to\infty} N\left(\sum_{i=1}^{n} p_i, \sum_{i=1}^{n} p_i(1-p_i)\right)$$

Proof The variance of $\sup(s)$ is $\sigma^2 = \sum_{i=1}^{n} \sigma_i^2$, where $\sigma_i^2 = (1-p_i)p_i$ is the variance of Bernoulli random variable $\sup(s|d_i)$. Therefore, we have $\sigma^2 \to \infty$ when $n \to \infty$, which satisfies the Lindeberg's condition of the *central limit theorem*. □

Therefore, we can compute the *approximate frequentness probability* of s by:

$$P(\sup(s) \geq \tau_s) = 1 - P(\sup(s) \leq (\tau_s - 1)) = 1 - \Phi\left(\frac{\tau_s - 1 - \mu}{\sigma}\right) \tag{6}$$

where $\mu = \sum_{i=1}^{n} p_i, \sigma^2 = \sum_{i=1}^{n} p_i(1-p_i)$, and Φ is the cumulative distribution function of standard normal distribution.

4.3 Support probabilities in uncertain sequences

In this section, we discuss the computation of support probability in a temporally uncertain sequence. As previously mentioned, directly expanding all possible sequences is infeasible in practice. Therefore, we design an efficient approach to compute the support probability without enumerating all possible worlds. We first define the *minimum possible occurrence* (*mpo*) as follows.

Definition 4.3 Given an uncertain sequence $d = \{e_1, \ldots, e_m\}$ and a sequential pattern $s = \langle s_1, \ldots, s_n\rangle$ $(n \leq m)$, a minimum possible occurrence (*mpo*) of s in d is an ordered size-n subset $\langle e_{k_1}, \ldots, e_{k_n}\rangle$ of d which have $s_i \subseteq e_{k_i}$.

Here, we use $s_i \subseteq e_{k_i}$ to represent $s_i \subseteq e_{k_i}.I$, for simplicity's sake. In the following example, we use only itemsets to represent uncertain events. Suppose $d = \{(a)(ab)(abc)(cd)\}$ and $s = \langle(a)(b)\rangle$, then $\langle(\mathbf{a})(a\mathbf{b})\rangle$, $\langle(\mathbf{a})(a\mathbf{b}c)\rangle$, $\langle(\mathbf{a}b)(a\mathbf{b}c)\rangle$ and $\langle(\mathbf{a}bc)(a\mathbf{b})\rangle$ are four *mpo*s of s in d, while $\langle(a)(ab)(abc)\rangle$ is not a *mpo*.

Two properties of *mpo* are presented in Lemmas 4.3 and 4.4, which will be used in computing support probability from *mpo*s.

Lemma 4.3 *If there are no mpo of s in d, $P(s \preceq d) = 0$*

Proof If no *mpo*s of s are found in d, there is no possible world $\hat{d}$ of d having $s \sqsubseteq \hat{d}$. Therefore, there does not exist any possible world which support s, and this proves that $P(s \preceq d) = 0$. □

Lemma 4.4 *If s is not probabilistically supported by any mpo of s in d, $P(s \preceq d) = 0$.*

Proof Let $O = \{o_1, \ldots, o_n\}$ be a set of all *mpos* of s in d. Suppose $P(s \preceq d | s \npreceq o_1, \ldots, s \npreceq o_n) > 0$, then there must exist a possible world $\hat{d}$ that contains an occurrence o_i of s. Thus, $P(s \preceq o_i) = P(d = \hat{d}) > 0$, which contrasts with $s \npreceq o_i, \forall o_i \in O$. This proves the correctness of the lemma. □

Let $O = \{o_1, \ldots, o_n\}$ be n *mpos* of s in d. If $O = \phi$, we have $P(s \preceq d) = 0$, according to Lemma 4.3; otherwise, we can compute the support probability $P(s \preceq d)$ in Eq. (7), referring to *the law of total probability*.

$$\begin{aligned} P(s \preceq d) =& P(s \preceq d | s \preceq o_1) P(s \preceq o_1) + P(s \preceq d | s \npreceq o_1) P(s \npreceq o_1) \\ =& P(s \preceq o_1) + P(s \preceq d | s \npreceq o_1) P(s \npreceq o_1) \end{aligned} \tag{7}$$

where $P(s \preceq d | s \preceq o_i) = 1$ by definition. Since o_1 and o_2 are independent, the probability $P(s \preceq d | s \npreceq o_1)$ can be further decomposed as:

$$\begin{aligned} P(s \preceq d | s \npreceq o_1) =& P(s \preceq o_2 | s \npreceq o_1) + P(s \preceq d | s \npreceq o_1, s \npreceq o_2) P(s \npreceq o_2) \\ =& P(s \preceq o_2) + P(s \preceq d | s \npreceq o_1, s \npreceq o_2) P(s \npreceq o_2) \end{aligned} \tag{8}$$

By substituting Eq. (8) into Eq. (7), the support probability $P(s \preceq d)$ can be computed by:

$$\begin{aligned} P(s \preceq d) =& P(s \preceq o_1) + P(s \preceq o_2) P(s \npreceq o_1) \\ &+ P(s \preceq d | s \npreceq o_1, s \npreceq o_2) P(s \npreceq o_2) P(s \npreceq o_1) \end{aligned} \tag{9}$$

We continue to decompose the support probability until it iterates through all *mpos*, as shown in Eq. (10):

$$\begin{aligned} P(s \preceq d) =& P(s \preceq o_1) + \sum_{i=2}^{n} \left(P(s \preceq o_i) \prod_{j=1}^{i-1} P(s \npreceq o_j) \right) \\ &+ P(s \preceq d | s \npreceq o_1, \ldots, s \npreceq o_n) \prod_{i=1}^{n} P(s \npreceq o_i) \end{aligned} \tag{10}$$

According to Lemma 4.4, we have $P(s \preceq d | s \npreceq o_1, \ldots, s \npreceq o_n) = 0$. Then, Eq. (10) can be shortened as:

$$\begin{aligned} P(s \preceq d) &= P(s \preceq o_1) + \sum_{i=2}^{n} \left(P(s \preceq o_i) * \prod_{j=1}^{i-1} P(s \npreceq o_j) \right) \\ &= \sum_{i=1}^{n} P(s \preceq o_i) * \mathcal{A}_i \end{aligned} \tag{11}$$

We use an auxiliary variable $\mathcal{A}_i$, which is defined in Eq. (12), to track the product of $P(s \npreceq o_1), \ldots, P(s \npreceq o_{(i-1)})$. Therefore, we can see that the amortized time complexity of Eq. (11) is $O(n)$.

$$\mathcal{A}_i = \begin{cases} 1 & \text{if } \ i = 1 \\ \mathcal{A}_{i-1} * P(s \npreceq o_{i-1}) & \text{if } \ i \geq 2 \end{cases} \tag{12}$$

Figure 3 is an example to show the process of computing $P(s \preceq d)$ by *mpos*. Let $g_l = 1$ and $g_h = 3$. Given $s = \langle (a)(b)(c) \rangle$, we can compute $P(s \preceq d) = 0.7$ by enumerating all possible worlds of d, as shown in Fig. 3b.

event	T	I
e1	1	{a}
e2	{3:0.4, 8:0.6}	{b}
e3	{4:0.5, 9:0.5}	{b}

(a)

world	T1	T2	T3	Prob	Satisfy?
w1	1	3	9	0.2	✓
w2	1	3	4	0.2	✓
w3	1	8	9	0.3	✗
w4	1	8	4	0.3	✓

(b)

Fig. 3 An example of computing support probability. **a** Uncertain sequence d. **b** Possible worlds $\hat{d}$

Procedure 2: FreqProb(s, D)

Input: s: a sequential pattern
D: an uncertain database
Output: $P(\sup(s) \geq \tau_s)$
$\mu \leftarrow 0, \sigma^2 \leftarrow 0$
foreach $d \in D$ **do**
  $\mathcal{A} \leftarrow 1$
  $p \leftarrow 0$ `// p is the support probability` $P(s \preceq d)$
  foreach $o_s \in d$ **do**
    compute $P(s \preceq o_s)$
    $p \leftarrow p + P(s \preceq o_s) * \mathcal{A}$
    $\mathcal{A} \leftarrow \mathcal{A} * P(s \not\preceq o_s)$
  end
  $\mu \leftarrow \mu + p$
  $\sigma^2 \leftarrow \sigma^2 + p * (1 - p)$
end
$P(\sup(s) \geq \tau_s) \leftarrow 1 - \Phi(\frac{\tau_s - 1 - \mu}{\sigma})$
return $P(\sup(s) \geq \tau_s)$

Next, we compute $P(s \preceq d)$ without enumerating the possible worlds table. First, we find two *mpo*s of s, which are $o_1 = \langle e_1, e_2 \rangle$ and $o_2 = \langle e_1, e_3 \rangle$; then, we can compute $P(s \preceq o_1) = 0.4$ and $P(s \preceq o_2) = 0.5$. Finally, $P(s \preceq d)$ is computed as $P(s \preceq d) = P(s \preceq o_1) + P(s \preceq o_2) * P(s \not\preceq o_1) = 0.4 + 0.5 * 0.6 = 0.7$, which is consistent with the result of expanding all possible worlds.

Suppose s is a sequential pattern and D is an uncertain database. Procedure 2 shows the process of computing the frequentness probability of s in D. In an uncertain sequence d, suppose $\mathcal{A}$ is the auxiliary variable defined in Eq. (12), then we can compute the support probability $p = P(s \preceq d)$ from all the values of $P(s \preceq o_s)$ by Eq. (11). Here, o_s is one of the *mpo*s of s in d and $P(s \preceq o_s)$ is the probability that s is supported by o_s. And we will discuss the computation of $P(s \preceq o_s)$ in Sect. 5. Since μ and σ^2 are the mean and variance of the Gaussian distribution which approximate the distribution of overall $\sup(s)$, we directly use Eq. (6) to compute the approximate frequentness probability $P(\sup(s) \geq \tau_s)$.

5 Management of temporal uncertainty

In this section, we discuss the computation of the probability that a pattern is supported by one of its minimal possible occurrences. Let $o_s = \langle e_{k_1}, \ldots, e_{k_n} \rangle$ be a *mpo* of a pattern s in an uncertain sequence d. We denote the probability of o_s satisfying gap constraints by $P_t(o_s)$. Therefore, we have $P(s \preceq o_s) = P_t(o_s)$ by definition.

Let $g_l = l$ and $g_h = h$. And we also denote the uncertain time of event e_{k_i} by T_{k_i}. A brute force approach of computing $P_t(o_s)$ is to directly decompose it by the *chain rule*, as shown in Eqs. (13) and (14).

$$\begin{aligned} P_t(o_s) &= \int_{l \le t_i - t_{i-1} \le h} \cdots \int f(T_{k_1} = t_1, \ldots, T_{k_n} = t_n) \mathrm{d}t_1 \ldots \mathrm{d}t_n \\ &= \int_{l \le t_i - t_{i-1} \le h} \cdots \int f(t_n | t_1 \ldots t_{n-1}) * \cdots * f(t_2|t_1) f(t_1) \mathrm{d}t_1 \ldots \mathrm{d}t_n \end{aligned} \tag{13}$$

$$\begin{aligned} P_t(o_s) &= \sum_{l \le t_{i+1} - t_i \le h} P(T_{k_1} = t_1, \ldots, T_{k_n} = t_n) \\ &= \sum_{l \le t_{i+1} - t_i \le h} \cdots \sum P(t_n | t_1, t_2, \ldots, t_{n-1}) * \cdots * P(t_2|t_1) * P(t_1) \end{aligned} \tag{14}$$

However, the complexity of this approach is exponential to the number of uncertain timestamps, so it is usually too complex to be used in practice. In this section, we propose a recursive approach to compute $P_t(o_s)$ efficiently.

5.1 Base case

Suppose $X \sim U(x^-, x^+)$ and $Y \sim U(y^-, y^+)$ are two uniformly distributed uncertain timestamps. We denote $P(\langle XY \rangle)$ as the probability that X and Y satisfy constraints (e.g., $l \le Y - X \le h$). In this section, we propose a geographic approach to compute $P(\langle XY \rangle)$ in constant time.

Figure 4 shows the geographic representations of X and Y. We can see that the gap constraints $g_l = l$ and $g_h = h$ correspond to the two straight lines $Y = X + l$ and $Y = X + h$ in Fig. 4. Here, we define a function V to restrict values in Y-axis, as shown in Eq. (15).

$$V(y) = \begin{cases} y^+ & \text{if} \quad y \ge y^+ \\ y & \text{if} \quad y^- < y < y^+ \\ y^- & \text{if} \quad y \le y^- \end{cases} \tag{15}$$

By applying the two endpoints in X-axis (x^-, x^+) to the straight lines $(Y = X + l, Y = X + h)$, we generate the following four endpoints in Y-axis:

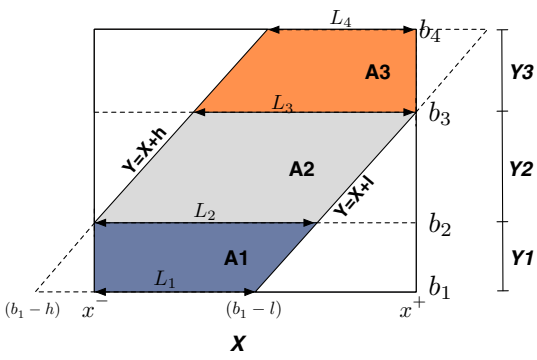

Fig. 4 An example of computings the probability of satisfying gap constraints

$$a_1 = V(x^- + l), \qquad a_2 = V(x^- + h)$$
$$a_3 = V(x^+ + l), \qquad a_4 = V(x^+ + h)$$

We sort the values of these four endpoints. Let $b_1 = a_1, b_2 = \min(a_2, a_3), b_3 = \max(a_2, a_3)$ and $b_4 = a_4$, then we have $b_1 \le b_2 \le b_3 \le b_4$. Therefore, the range of $[b_1, b_4]$ is divided into three sub-partitions as $[b_1, b_4] = [b_1, b_2] \cup [b_2, b_3] \cup [b_3, b_4]$. Referring to *the law of total probability*, we can compute $P(\langle XY \rangle)$ as follows:

$$\begin{aligned} P(\langle XY \rangle) &= \sum_{k=1}^{3} P(\langle XY \rangle \,|\, y \in [b_k, b_{k+1}]) * P(y \in [b_k, b_{k+1}]) \\ &= P(\langle XY_k \rangle) * P(Y_k) \end{aligned} \tag{16}$$

where $Y_k = \{Y | y \in [b_k, b_{k+1}]\}$, and $P(Y_k)$ is computed by:

$$P(Y_k) = \int_{b_k}^{b_{k+1}} \frac{1}{y^+ - y^-} \mathrm{d}y = \frac{b_{k+1} - b_k}{y^+ - y^-} \tag{17}$$

By dividing the range of Y in this way, we can use a geographic approach to compute $P(\langle XY_k \rangle)$ in constant time. In Fig. 4, let S_k be the area of the rectangle with $X \in [x^-, x^+]$ and $Y \in [b_k, b_{k+1}]$; A_k be the area of a trapezoid between two boundary lines within S_k. Then, we can compute $P(\langle XY_k \rangle)$ as:

$$P(\langle XY_k \rangle) = \frac{A_k}{S_k} = \frac{(1/2) * (L_{k+1} + L_k) * (b_{k+1} - b_k)}{(b_{k+1} - b_k) * (x^+ - x^-)} = \frac{L_{k+1} + L_k}{2 * (x^+ - x^-)} \tag{18}$$

where L_k is the length of the X-directed line between two constraints at $y = b_k$, and it can be computed by:

$$L_k = \min(b_k - l, x^+) - \max(b_k - h, x^-) \tag{19}$$

The induction and proof of correctness of Eq. (18) can be found in Appendix 1.

5.2 Recursive function for Pdf-modeled uncertainty

Let $T_1, \ldots, T_n$ be uniformly distributed uncertain timestamps with $T_i \sim U[t_i^-, t_i^+]$. In this section, we develop a recursive function to compute the probability that $T_1, \ldots, T_n$ satisfy gap constraints. Suppose the range of T_{n-1} consists of q disjoint sub-partitions as $[t_{n-1}^-, t_{n-1}^+] = \cup_{j=1}^{q} [t_{n-1}^j, t_{n-1}^{j+1}]$, then each point $x = t_{n-1}^j$ in the T_{n-1} can generate two boundary points:

$$e_j^1 = \left(x, \min\left(t_n^+, \max\left(x + l, t_n^-\right)\right)\right), \quad e_j^2 = \left(x, \min\left(t_n^+, \max\left(x + h, t_n^-\right)\right)\right)$$

For example, in Fig. 5, $\{e_1^1, e_1^2, \ldots, e_4^1, e_4^2\}$ are eight boundary points derived from t_{n-1}^j (for $j \in [1, 4]$). The projections of all the boundary points in T_n-axis are used to divide the range of T_n into q disjoint sub-partitions. For instance, $t_n^1 < \ldots < t_n^6$ are the projections of the boundary points in T_n-axis, which divide the range of T_n into five sub-partitions.

By this way, we can use Eq. (18) to compute the value of $P(\langle T_{n-1}^j T_n^i \rangle)$. For the sake of simplicity, we first define $P_n(T)$ as follows:

Definition 5.1 Suppose $[a, b] \subseteq [t_n^-, t_n^+]$ and $T \sim U[a, b]$, then we define $P_n(T)$ to be the probability that $T_1, \ldots, T_{n-1}, T_n$ satisfies gap constraints when $T_n = T$, denoted as $P_n(T) = P(\langle T_1, \ldots, T_{n-1}, T_n \rangle \wedge T_n = T)$.

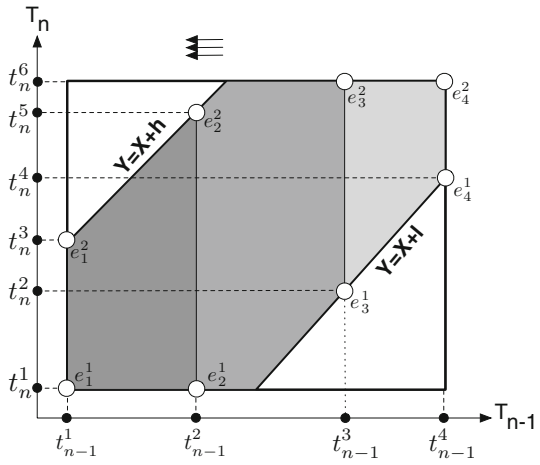

Fig. 5 An example of dividing the range of T_n

Then, we can decompose $P(\langle T_1, \ldots, T_n \rangle)$ by all possible ranges of T_n in Eq. (20).

$$\begin{aligned} P(\langle T_1, \ldots, T_n \rangle) &= \sum_i P\left(\langle T_1, \ldots, T_{n-1}, T_n \rangle \,|T_n^i\right) P\left(T_n^i\right) \\ &= \sum_i P_n\left(T_n^i\right) \end{aligned} \tag{20}$$

The computation of $P_n(T_n^i)$ can be divided into subproblems of computing $P(\langle T_1, \ldots, T_{n-1}^j, T_n^i \rangle)$, for all possible ranges of T_{n-1}.

These subproblems are combined together by the law of total probability, as shown in Eq. (21).

$$P_n\left(T_n^i\right) = \sum_j P\left(\left\langle T_1, \ldots, T_{n-1}^j, T_n^i \right\rangle\right) P\left(T_{n-1}^j\right) P\left(T_n^i\right) \tag{21}$$

Since the gap constraints only regulate relationships between adjacent timestamps, we have:

$$P\left(\left\langle T_1, \ldots, T_{n-1}^j, T_n^i \right\rangle\right) = P\left(\left\langle T_1, \ldots, T_{n-1}^j \right\rangle\right) P\left(\left\langle T_{n-1}^j, T_n^i \right\rangle\right)$$

Thus, Eq. (21) can be written as the following recursive function:

$$\begin{aligned} P_n\left(T_n^i\right) &= \sum_j P\left(\left\langle T_1, \ldots, T_{n-1}^j \right\rangle\right) P\left(\left\langle T_{n-1}^j, T_n^i \right\rangle\right) P\left(T_{n-1}^j\right) P\left(T_n^i\right) \\ &= \sum_j P_{n-1}\left(T_{n-1}^j\right) P\left(\left\langle T_{n-1}^j T_n^i \right\rangle\right) P\left(T_n^i\right). \end{aligned} \tag{22}$$

5.3 Recursive function for pmf-modeled uncertainty

We denote the *pmf* of a discrete uncertain timestamp T_i by $f_{T_i} = P(T_i = t_i^k)$, where t_i^k is one of the possible values of T_i. Here, we define $P_n(t)$ for pmf-modeled uncertain timestamps.

Definition 5.2 Let t be a possible value of T_n, then $P_n(t)$ is defined to be the probability that $T_1, \ldots, T_{n-1}, T_n$ satisfies gap constraints when $T_n = t$, denoted as $P_n(t) = P(\langle T_1, \ldots, T_{n-1}, T_n \rangle \wedge T_n = t)$.

Suppose T_n has p possible values such as $t_n^1, \ldots, t_n^p$, then $P(\langle T_1, \ldots, T_n \rangle)$ is the sum of $P_n(t_n^i)$ $(\forall i \in [1, p])$, as shown in Eq. (23).

$$\begin{aligned} P\left(\langle T_1, \ldots, T_n \rangle\right) &= \sum_i P\left(\langle T_1, \ldots, T_n \rangle \,|T_n = t_n^i\right) P\left(T_n = t_n^i\right) \\ &= \sum_i P\left(\langle T_1, \ldots, T_n \rangle \wedge T_n = t_n^i\right) \\ &= \sum_i P_n\left(t_n^i\right). \end{aligned} \tag{23}$$

We first compute the probability $P(\langle T_1 T_2 \rangle)$ that two uncertain timestamps T_1 and T_2 satisfy gap constraints. Suppose $f_{T_1}(t_1^j) = P(T_1 = t_1^j)$ and $f_{T_2}(t_2^i) = P(T_2 = t_2^i)$. Then, $P(\langle T_1 T_2 \rangle)$ can be computed from the joint distribution of (X, Y), as shown in Eq. (24).

$$P\left(\langle T_1 T_2 \rangle\right) = \sum_i P_2\left(t_2^i\right) = \sum_i \sum_j \delta\left(t_1^j, t_2^i\right) P\left(t_1^j\right) P\left(t_2^i\right) \tag{24}$$

where the value of function $\delta(x, y)$ depends on if x and y satisfy gap constraints or not. Let $g_l = l$ and $g_h = h$, then $\delta(x, y)$ is defined as follows:

$$\delta(x, y) = \begin{cases} 1 & \text{if } \ l \le y - x \le h \\ 0 & \text{otherwise} \end{cases} \tag{25}$$

The key idea of our recursive approach is to split the computation of $P_n(t_n^i)$ into subproblems of computing $P_{n-1}(t_n^j)$ and combine them by the law of total probability, as shown in Eq. (26).

$$P_n\left(t_n^i\right) = \sum_j P\left(\left\langle T_1, \ldots, T_{n-1}^j, T_n^i \right\rangle\right) P\left(T_{n-1} = t_{n-1}^j\right) P\left(T_n = t_n^i\right) \tag{26}$$

Since gap constraints only apply to adjacent timestamps, $T_1, \ldots, T_n$ satisfy gap constraints, if we have $\langle T_1, \ldots, T_{n-1} \rangle$ and $\langle T_{n-1} T_n \rangle$. Therefore, Eq. (26) can be written by the following recursive function:

$$\begin{aligned} P_n\left(t_n^i\right) &= \sum_j P\left(\left\langle t_{n-1}^j t_n^i \right\rangle\right) P\left(\langle T_1, \ldots, T_{n-1} \rangle\right) P\left(t_{n-1}^j\right) P\left(t_n^i\right) \\ &= \sum_j \delta\left(t_{n-1}^j, t_n^i\right) P_{n-1}\left(t_{n-1}^j\right) P\left(t_n^i\right). \end{aligned} \tag{27}$$

6 Integration of uncertainty management

In this section, we integrates our uncertainty management into the process of sequential pattern mining. Referring to the recursive functions in Eqs. (22) and (27), we can see that the complexity of uncertainty computation can be reduced by reusing previous computational results. Specifically, if the values of $P_{n-1}(T_{n-1}^i)$ or $P_{n-1}(t_{n-1}^i)$ are saved during searching a k-length pattern s, they can be reused in mining $(k+1)$-length patterns that are extended from s. We design a new data structure, which is so called *uncertain projected database*, to help save previous computational results in uncertain sequential pattern mining.

6.1 Uncertain database projection

Different from the traditional PrefixSpan method, we project temporally uncertain databases by minimal possible occurrences. First, we have the following definitions.

Definition 6.1 Suppose all the items in an element are listed alphabetically. Let $o_s = \langle e_{k_1}, \ldots, e_{k_n} \rangle$ be a *mpo* of a pattern s in a sequence d, then $\alpha = \langle e'_{k_1}, \ldots, e'_{k_n} \rangle$ is defined to be a *prefix* of the uncertain sequence d with respect to o_s, if and only if: (1) $e'_{k_i} = e_{k_i}$ for $i \leq (n-1)$; (2) $e'_{k_n} \subseteq e_{k_n}$; (3) all the items in $(e_{k_n} - e'_{k_n})$ are alphabetically after those in e'_{k_n}.

Definition 6.2 The *o_s-projected sequence* $d|_{o_s}$ is defined to be the suffix β of d such that $d = \alpha\beta$ with α being the prefix of d derived from o_s.

Consider the following example where we use only items to represent uncertain events. Let $d = \{(ab), (cd), (e)\}$ and $s = \langle (a)(c) \rangle$, then $o = \langle (ab)(cd) \rangle$ is a *mpo* of s in d, and $\langle (ab)(c) \rangle$ is an uncertain prefix derived from o. $d|_o = \{(\_d)(e)\}$ is the o-projected sequence in d, where $\_d$ indicates that the last element of the prefix, which is item c here, form one element (cd) together with d.

Since the minimal gap constraints is $g_l = l$, if an event in $d|_{o_s}$ is not able to support any extension of s together with o_s, we can prune it to minimize the size of $d|_{o_s}$ by referring to Lemma 6.1.

Lemma 6.1 *Suppose $o_s = \langle e_{k_1}, \ldots, e_{k_n} \rangle$ is a mpo of a pattern s in an uncertain sequence d and $d|_{o_s}$ is the o_s-projected sequence of d. An uncertain event e_i in $d|_{o_s}$ can be pruned if $P(T_i \geq T_{k_n} + g_l) = 0$, where T_i is the timestamp of event e_i and T_{k_n} is the timestamp of event e_{k_n}.*

Proof If $P(T_i \geq T_{k_n} + g_l) = 0$, we have $P(T_{k_1}, \ldots, T_{k_n}, T_i) = 0$, which indicates that e_i does not contribute to support any extension of s. Therefore, it can be pruned. □

Here, let $\max(T_i)$ be the maximum possible value of T_i and $\min(T_{k_n})$ be the minimum value of T_{k_n}, then $P(T_i \geq T_{k_n} + g_l) = 0$ is equivalent to $\max(T_i) < \min(T_{k_n}) + g_l$.

Next, we define *uncertain projected sequence* and *uncertain projected database* as follows.

Definition 6.3 Suppose $O = \{o_1, \ldots, o_n\}$ is a set of all possible *mpo*s of s in d, then a *s-projected uncertain sequence* $d|_s$ is defined to be a set of all o_i-projected sequences, denoted by $d|_s = \{d|_{o_1}, \ldots, d|_{o_n}\}$. And the *$s$-projected uncertain database* is a collection of all s-projected sequences, denoted by $D|_s = \{d_1|_s, \ldots, d_n|_s\}$.

6.2 Incremental uncertainty computation

Suppose o_s is one of all the *mpo*s of pattern s in an uncertain sequence d. For the simplicity of notation, we denote $P_t(\cdot|o_s)$ to be a set of either $P_n(T_n^i)$ or $P_n(t_n^i)$ values.

Definition 6.4 If temporal uncertainty is modeled by pdf, we have $P_t(\cdot|o_s) = \{P_n(T_n^1), \ldots, P_n(T_n^p)\}$; we have $P_t(\cdot|o_s) = \{P_n(t_n^1), \ldots, P_n(t_n^p)\}$, if temporal uncertainty is pmf-modeled.

All the values of $P_t(\cdot|o_s)$ are saved and associated with each $d|_{o_s}$ in uncertain projected databases. Procedure 3 shows the process of projecting an uncertain database. Here, s is a pattern, $D|_s$ is the s-projected uncertain database and i is an item in $D|_s$. In each o_s-projected

Procedure 3: Project(i, $D|_s$)

Input: $D|_s$: an uncertain projected database;
i: an item
Output: $D|_{s'}$
$D|_{s'} \leftarrow \phi$
foreach $d|_s \in D|_s$ **do**
  $d|_{s'} \leftarrow \phi$
  foreach $d|_{o_s} \in d|_s$ **do**
    foreach $e \in d|_{o_s} \wedge i \in e$ // search the occurrences of item i
    do
      build $d|_{o_{s'}}$
      compute $P_t(\cdot|o_{s'})$ from $P_t(\cdot|o_s)$ and $e.T$ and save it in $d|_{o_{s'}}$
      $d|_{s'} \leftarrow d|_{s'} \cup \{d|_{o_{s'}}\}$
    end
  end
  $D|_{s'} \leftarrow D|_{s'} \cup \{d|_{s'}\}$
end
return $D|_{s'}$

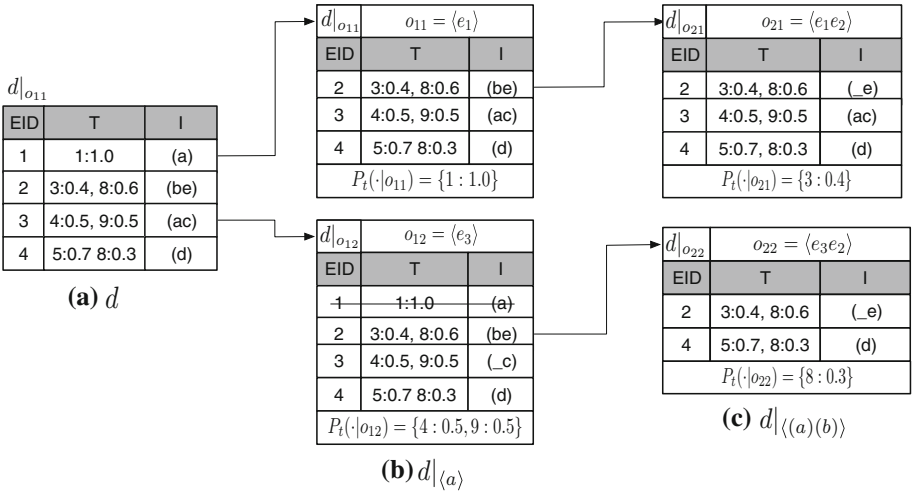

Fig. 6 An example of projecting an uncertain sequence

sequence $d|_{o_s}$, if an event e contains an occurrence of i, it corresponds to a *mpo* $o_{s'}$ of s', where s' is a pattern extended from s with item i. Then, we can build an $o_{s'}$-projected sequence by these two operations: (1) remove any items which is alphabetically smaller than or equal to i in e; (2) remove any event e_k with $\max(T_k) < \min(T) + g_l$, where T_k is the event time of e_k and T is the time of event e.

Since $P_t(\cdot|o_s)$ is saved in $d|_{o_s}$, we use it to compute $P_t(\cdot|o_{s'})$. If s' is an item-extended pattern such as $s' = s \cup \{i\}$, we have $P_t(\cdot|o_{s'}) = P_t(\cdot|o_s)$ because $o_{s'} = o_s$; if $s' = s + \{i\}$, we compute the value of $P_t(\cdot|o_{s'})$ directly from $P_t(\cdot|o_s)$ and the timestamp of event e, by referring to Eqs. (22) and (27). Here, e is the event that contains an occurrence of item i.

Figure 6 shows an example of projecting an uncertain sequence, where we set $g_l = 1$ and $g_h = 5$. The uncertain sequence d is shown in Fig. 6a. Figure 6b illustrates the $\langle a \rangle$-projected uncertain sequence of d. Here, $o_{11} = \langle e_1 \rangle$ and $o_{12} = \langle e_3 \rangle$ are two *mpo*s of the pattern $\langle a \rangle$.

And $d|_{o_{11}}$ and $d|_{o_{12}}$ are the two projected sequences w.r.t. o_{11} and o_{12}. Notice that the event e_1 in $d|_{o_2}$ can be eliminated because $P(T_3 \geq T_1 + 1) = 0$, according to Lemma 6.1.

Figure 6c shows the $\langle(a)(b)\rangle$-projected uncertain sequence. Here, $d|_{o_{21}}$ is projected from $d|_{o_{11}}$, since the event e_2 in $d|_{o_{11}}$ has an occurrence of item b; $d|_{o_{22}}$ is projected from $d|_{o_{12}}$, since the event e_2 in $d|_{o_{12}}$ contains an occurrence of item b. For example, since $g_l = 1$ and $g_h = 5$, we can directly compute $P_t(\cdot|o_{22}) = \{8 : 0.3\}$ from the saved value $P_t(\cdot|o_{12}) = \{4 : 0.5, 9 : 0.5\}$ and $T_2 = \{3 : 0.4, 8 : 0.6\}$ in $d|_{o_{12}}$ by Eq. (27).

6.3 Pruning

Compared to our previous conference version [12], we propose pruning techniques in this section, which are used to speed up uncertainty computation. Suppose $X_i \sim \mathcal{B}(1, p_i)$ are n Bernoulli random variables, where $p_i = P(s \preceq d_i) > 0$ is the nonzero support probability of pattern s in an uncertain sequence d_i. Let $S = \sup(s)$ here, then the overall support S is the sum of X_i such as $S = X_1 + \ldots + X_n$. And the expected value of S is denoted by $\mathrm{E}(S)$. Lemmas 6.2 and 6.3 describe how to prune s without computing its frequentness probability.

Lemma 6.2 (Count pruning) *Suppose there are n uncertain sequences which have $P(s \preceq d) > 0$ then s can be pruned, if $n < \tau_s$.*

Proof Because $S = X_1 + \ldots + X_n$ and $X \in \{0, 1\}$, we have $S \leq n$. If $n < \tau_s$, then $S < \tau_s$ and s is not likely to be a p-FSP. □

The count pruning technique has also been used in mining uncertain frequent itemsets in [25].

Lemma 6.3 (Hoeffding's inequality pruning) *Let $t = \sqrt{\frac{n*\ln(1/\tau_p)}{2}}$, then s can be pruned if $E(S) + t < \tau_s$.*

Proof Since $X_1, \ldots, X_n$ are independent random variables bounded by the interval $[0, 1]$, $S = X_1 + \ldots + X_n$ satisfies *Hoeffding's inequality*: $P(S - \mathrm{E}[S] \geq t) \leq \exp(-\frac{2t^2}{n})$. Here let $t = \sqrt{\frac{n*\ln(1/\tau_p)}{2}}$, then we have $P(S \geq \mathrm{E}(S)+t) \leq \tau_p$. if $\mathrm{E}(S)+t < \tau_s$, $P(\sup(s) \geq \tau_s) < P(\mathrm{E}(S) + t) \leq \tau_p$. □

Beside pruning in computing frequentness probabilities, Lemma 6.4 describes a pruning technique based on gap constraints (e.g., $g_l = l$ and $g_h = h$) to speed up uncertainty management.

Lemma 6.4 (Gap pruning) *Suppose $a_1 \leq T_1 \leq b_1$ and $a_2 \leq T_2 \leq b_2$, then $P(\langle T_1, T_2\rangle) = 0$ if $b_2 < a_1 + l$ or $a_2 > b_1 + h$.*

Proof The possible range of T, which satisfies $g_l \leq T - T_1 \leq g_h$, is $[a_1 + l, b_1 + h]$. Given $a_2 \leq T_2 \leq b_2$, we have $P(\langle T_1 T_2\rangle) = 0$, if $b_2 < a_1 + l$ or $a_2 > b_1 + h$. □

We apply gap pruning in two places: (1) it helps to compute the base case of two two uncertain timestamps satisfying constraints, as in equation either (16) or (24); (2) it speeds up the computation of $P_t(\cdot|o_{s'})$ in the 3 function. In an uncertain sequence d, suppose the minimal and maximal timestamps of $P_t(\cdot|o_s)$ are a_1 and b_1. Let $a_2 \leq T \leq b_2$ be the time of uncertain event e that contains an occurrence of item i. Therefore, if $b_2 < a_1 + l$ or $a_2 > b_1 + h$, we do not need to compute the values of $P_t(\cdot|o_{s'})$ because we have $P(\cdot|o_{s'}) = 0$. Here, $s' = s + \{i\}$ is a sequence extension of s with item i.

7 Evaluation

7.1 Synthetic data generation

We employ the IBM market-basket data generator [2] to generate synthetic datasets using the following parameters:

C: number of sequences;
T: average number of transactions/itemsets per data-sequence;
L: average number of items per transaction/itemset per data-sequence;
I: number of different items.

To add uniform pdf-modeled uncertainty, we replace each point-value timestamp t by a uniformly distributed random variable $T \sim [(1-r)*t, (1+r)*t]$, where r is randomly drawn from (0, 1). We name the generated synthetic dataset by parameters. For example, the dataset named $T4L10I1C10$ indicates that $T = 4$, $L = 10$, $I = 1 \times 1000$ and $C = 10 \times 1000$.

Recall from Sect. 5 that the brute force method to compute the probability of an occurrence satisfying gap constraints is to compute Eq. (13) or Eq. (14) using chain rule. This method is implemented and abbreviated as *bf*. And our USPM algorithm is abbreviated as *uspm*. In order to evaluate the effect of our pruning techniques, the approximate algorithm with count pruning techniques is named *uspm-c*; we apply both count pruning and Hoeffding's inequality pruning in our algorithm and name it by *usm-ch*; and the algorithm with all three pruning techniques is so called *uspm-chg*. All the experiments were done in a desktop with Intel(R) Core (TM) Duo CPU @ 2.33 GHz and 4 GB memory.

7.2 Scalability

In Fig. 7, we compare the running time of *bf*, *uspm*, *uspm-c*, *uspm-ch* and *uspm-chg* on synthetic datasets with uniformly distributed temporal uncertainty. We set $\tau_s = 0.5\,\%$, $\tau_p = 0.7$, $g_l = 1$, and $g_h = 10$. And $C = 10{,}000$, $T = 4$, $I = 10{,}000$ and $L = 2$ are the default parameters used to generate datasets. Thereafter, we vary one of the parameters to test the performance in different scales. For example, in Fig. 7a, C varies from 10,000 to 100,000; in Fig. 7b, T varies from 10 to 30; in Fig. 7c, L varies from 4 to 10; and in Fig. 7c I varies from 500 to 50,000. Figure 7 shows the following phenomena:

(1) The running time of both *bf* and *uspm* increases with the increment of C, T, L, as the increment of these parameters generates larger-scale datasets.
(2) The running time decreases slightly with the increment of I. When I is set to a larger value, the number of *mpos* of a pattern in a sequence is fewer because there are fewer repeated items in a sequence.
(3) Our algorithm *uspm* is significantly faster than *bf* under every setting of parameters, which proves the effectiveness of our uncertainty computation approach.
(4) The *uspm-chg* algorithm with all three pruning methods is about two times faster than the original algorithm *uspm*, which proves the overall effectiveness of our pruning techniques.
(5) Comparing the performance of *uspm-c*, *uspm-ch* and *uspm-chg*, we can see that the effect of gap pruning is more significant than that of other two pruning techniques. And gap pruning becomes even more effective when T and L are larger.

Figure 9 compares the running time of *bf*, *uspm*, *uspm-c*, *uspm-ch* and *uspm-chg* with different user-defined parameters in the uncertain dataset T10L2I10C10. We set the default values as $\tau_s = 0.2\,\%$, $\tau_p = 0.7$, $g_l = 1$, and $g_h = 20$. In Fig. 8a, τ_s decreases from 0.5 to

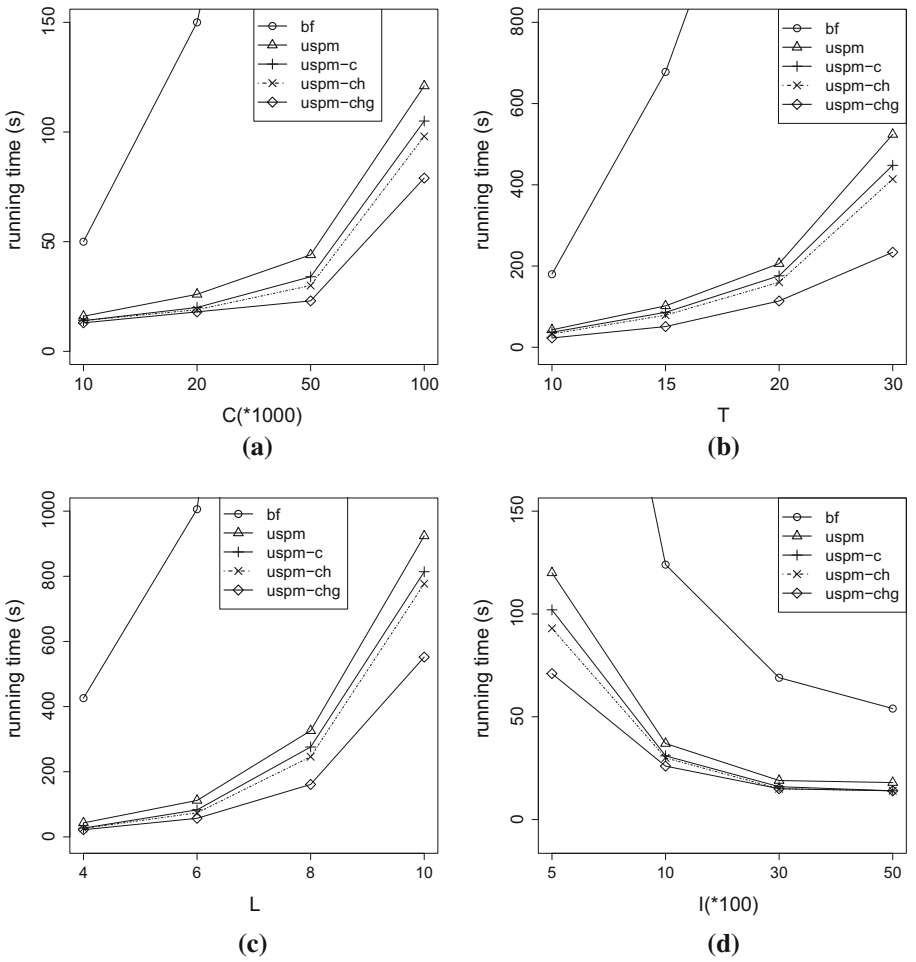

Fig. 7 Scalability in synthetic uncertain datasets. **a** Scale of C. **b** Scale of T. **c** Scale of L. **d** Scale of I

0.1 %; in Fig. 8b, τ_p varies from 0.5 to 0.8; g_h varies from 4 to 10 in Fig. 8c; and g_l varies from 1 to 10 in Fig. 8d. And we observe the following phenomena in Fig. 9:

(1) The running time of all five algorithms increases with the decrement of τ_s. The smaller τ_s is, the more p-FSPare found. This explains why all the algorithms take more time when τ_s is set to a smaller value.
(2) The performance of all the algorithms is relatively stable despite the variation of τ_p. According to the *Hoeffding's inequality*, the support of a sequential pattern is bounded to its expected value as $P(|S - E(S)| > t) < \delta$, where $S = \sup(s)$. Therefore, the frequentness of a pattern is deterministic if its expected support is much smaller or larger than τ_s. As the supports of patterns are evenly distributed in the datasets, only a small number of patterns change from being not frequent to being frequent when we lower the value of t_p.That is the reason why the running time of uncertain SPM algorithms does not fluctuate significantly in Fig. 8b.

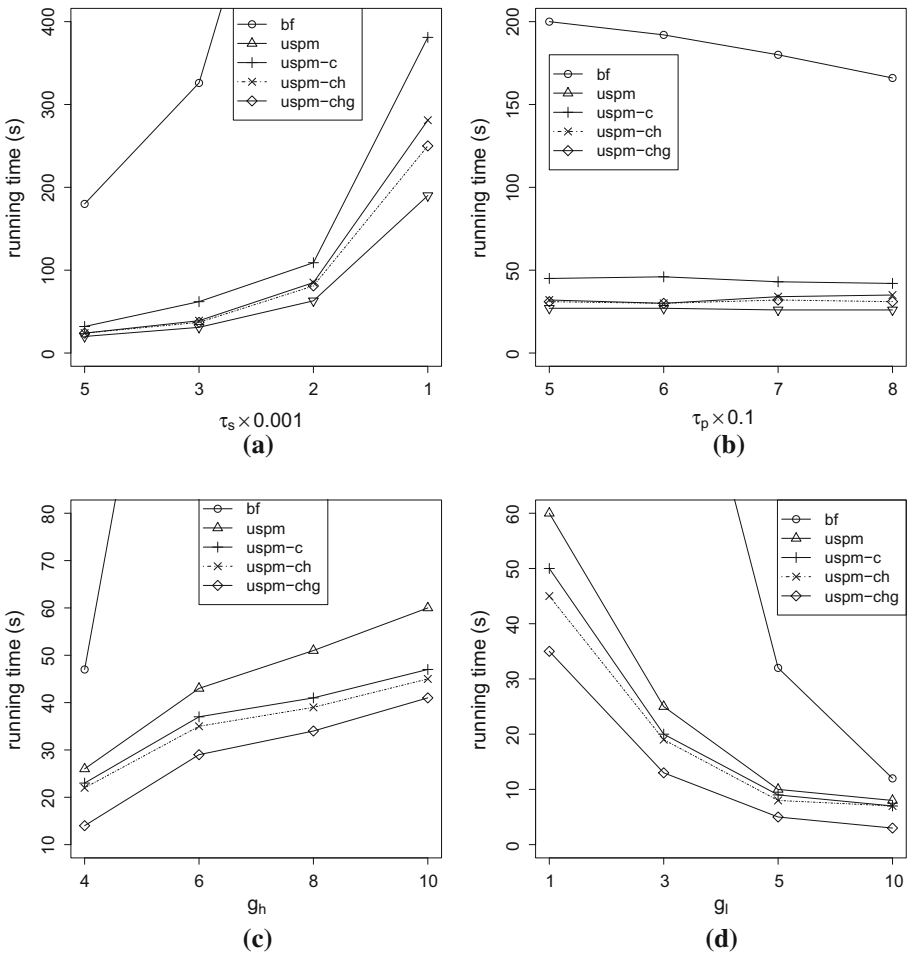

Fig. 8 Effect of parameters in synthetic uncertain datasets. **a** Effect of τ_s. **b** Effect of τ_p. **c** Effect of g_h. **d** Effect of g_l

(3) The running time of all the algorithms decreases when we decrease g_h or increase g_l. Meanwhile, gap pruning becomes more effective when g_h becomes smaller or g_l becomes larger. This is because a smaller gap $(g_h - g_l)$ indicates a more strict constraint to sequential patterns.

7.3 Approximation evaluation

In our model, we use a discrete pmf to approximate any arbitrary shaped pdf of temporal uncertainty. In this section, we evaluate the effectiveness of this approximation. Let $T \sim U(t^-, t^+)$ be an uncertain timestamp modeled by a uniform distribution. To generate a pmf to approximate T, we divide $[t^-, t^+]$ into k equal-width sub-partitions, and then, T can be approximated by $\{T|t_1 : p_1, \ldots, t_k : p_k\}$. Let $[a, b]$ be the ith sub-partition, then we have $t_i = (a + b)/2$ and $p_i = (b - a)/(t_r - t_l)$. Here, k is a parameter to control the approximate level.

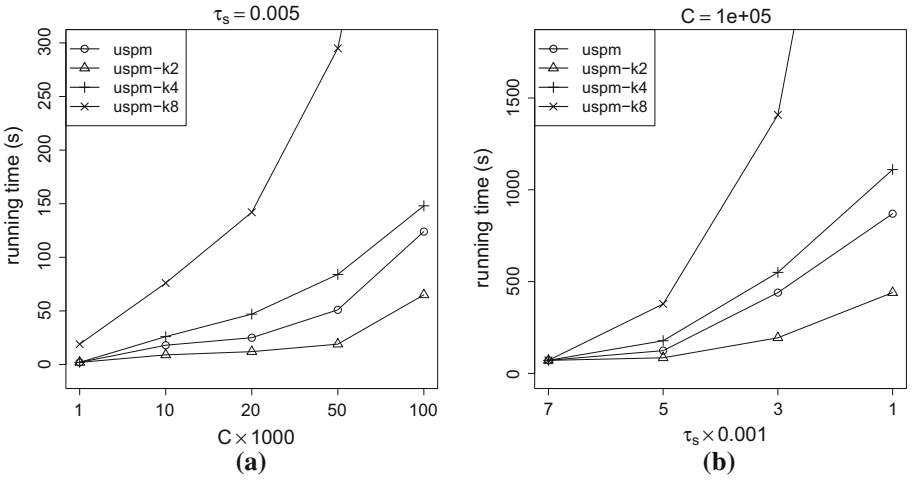

Fig. 9 Compare the running time of approximate USPM algorithms. **a** Vary C. **b** Vary τ_s

Table 2 Precisions and recalls of approximation

τ_s (%)	*uspm-k*2		*uspm-k*4		*uspm-k*8	
	P	*R*	*P*	*R*	*P*	*R*
0.1	0.81	0.74	0.94	0.92	1.00	0.97
0.3	0.88	0.81	0.94	0.95	0.99	0.99
0.5	0.92	0.94	0.98	0.99	1	0.98
0.7	1	0.99	1	0.99	1	0.99

In the uncertain dataset $T10L4I10C100$, we vary the value of τ_s to test the effectiveness of approximation techniques. Table 2 shows the *Precision(P)* and *Recall(R)* of approximate USPM algorithms, where *uspm-kn* represents the approximate algorithm with $k = n$.

The pmf approximation performs well, as we can see that both the *Precision* and *Recall* of *appr* are close to 1. Note that when $\tau_s = 0.7\,\%$, there are only 1-length p-FSPs mined in all algorithms. Therefore, the value of k does not affect the outputs and that is why all *uspm-kn* have the same performance in this case.

Figure 9 compares the running time of our approximate USPM algorithms with different values of k. In Fig. 9a, we set $\tau_s = 0.5\,\%$ and then compare the efficiency of *uspm*, *uspm-k*2, *uspm-k*4 and *uspm-k*8 under different C scales; in Fig. 9b, we set $C = 100{,}000$ and vary τ_s from 0.1 to 0.7 %. The trade-off between accuracy and efficiency is that a larger value of k makes the approximation more accurate, while the time cost increases significantly at the same time.

7.4 Mining sequential patterns in stock dataset

Compared with our previous conference version [12], we apply our USPM algorithm to a real-world stock market dataset. The prices for 1025 stocks from January 1, 2008, to March 20, 2015, in Shanghai Stock Exchange Center are extracted from *Yahoo! Finance*.

We define two types of events here. If the highest price of a stock A grows more than 8 % above its opening price in one day, we denote this event as $1A$; if the lowest price of A

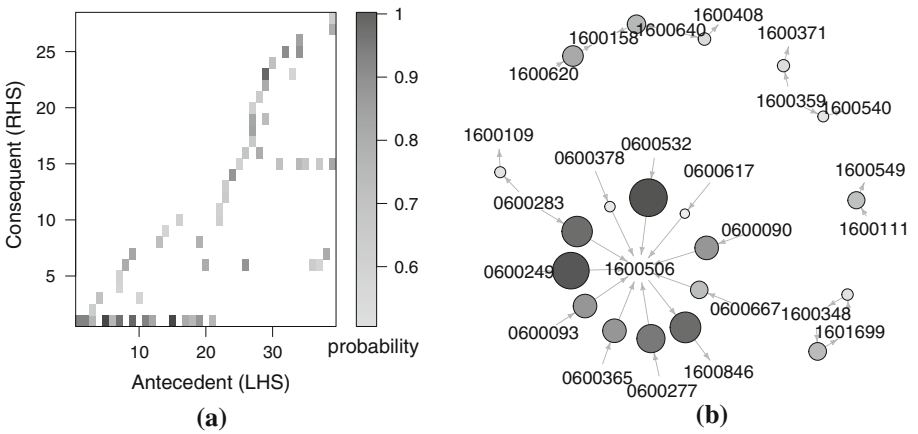

Fig. 10 Mined p-FSPs in stock dataset. **a** Matrix with 58 p-FSPs. **b** A sample of 20 p-FSPs

drops more than 8 %, we denote this event as $0A$, where 1 and 0 are two flags to indicate the increase or decrease of prices.

However, as the original dataset records the daily lowest and highest prices without their exact timestamps, the precise occurrence time of an event is unknown and is modeled by a uniformly distributed random variable in the indicated day. For example, if we observe the event $1A$ in January 1, 2009, then the event time of $1A$ is assumed to be uniformly distributed in the trading time period of this day.

We group the data by weeks. All events within one week forms a sequence, and this generates 373 uncertain sequences in total. We set $\tau_s = 2\,\%, \tau_p = 0.5, g_l = 1$ (h) and $g_h = 48$ (h). The *uspm* algorithm costs about 28 s to find 1490 p-FSPs, which consists of 1432 frequent items and 58 two-length p-FSPs.

Figure 10a shows the frequentness probability distribution of two-length p-FSPs using a matrix. A two-length sequential pattern can be represented by $\{A\} => \{B\}$, where $\{A\}$ is an itemset in antecedent (LHS) and $\{B\}$ is an itemset in consequent (RHS). Each unit in Fig. 10a represents a p-FSP, and the darkness of the unit is determined by the frequentness probability of the pattern. We can see that there is one item in RHS which connects to many item in LHS and most of p-FSPs have moderate frequentness probability between 0.7 and 0.9.

Figure 10b uses a graph to show the connections between stocks by a sample of 20 p-FSPs. Each connection is a p-FSP and is associated with a node. The size of the node is larger and its color becomes darker if the frequentness probability of the pattern is higher. This graph helps to reveal and understand the relations between stocks.

We observe that the USPM algorithm finds some patterns which are consistent with our knowledge. For example, $\langle(1600348)(1601699)\rangle$ is such a pattern, where 600,348 and 601,699 are the stock code of two coal mining companies in the same province. However, some hidden patterns are also found. For example, a p-FSP $\langle(0600365)(1600506)\rangle$ indicates that when the price of 600,365, a wine-making company decreases, it is often followed by the price increase of 600,506, which is a farming company.

8 Conclusion

In this paper, we propose a new SPM algorithm to find sequential patterns from temporally uncertain data. We design an incremental approach to efficiently compute temporal uncer-

tainty and integrate it into a classic pattern-growth SPM algorithm. The experimental results on both synthetic and real datasets prove that our algorithm is efficient and scalable.

Appendix 1: Deriving the geographic approach

Suppose $X \sim U(x^-, x^+)$ and $Y \sim U(y^-, y^+)$ are two uniformly distributed uncertain timestamps, then the joint 2-D distribution of X and Y is:

$$f(x, y) = \begin{cases} \frac{1}{(x^+ - x^-)(y^+ - y^-)} & \text{if } x \in [x^-, x^+], y \in [y^-, y^+] \\ 0 & \text{otherwise} \end{cases} \tag{28}$$

First of all, we define two functions $V_x(x)$ and $V_y(y)$ which restrict the possible values of x and y.

$$V_x(x) = \begin{cases} x^-, & x < x^- \\ x, & x^- \le x \le x^+ \\ x^+, & x > x^+ \end{cases} \tag{29}$$

$$V_y(y) = \begin{cases} y^-, & y < y^- \\ y, & y^- \le y \le y^+ \\ y^+, & y > y^+ \end{cases} \tag{30}$$

Given the minimal gap constraint $g_l = l$ and maximal constraint $g_h = h$ ($l < h$), we have:

$$\begin{aligned} Y \le X + h &\implies y \le x^+ + h \\ Y \ge X + l &\implies y \ge x^- + l \end{aligned}$$

Since $y \in [y^-, y^+]$, we also have $V_y(x^- + l) \le y \le V_y(x^+ + h)$. Let $P(\langle XY \rangle)$ be the probability that X and Y satisfy gap constraints, then $P(\langle XY \rangle)$ can be computed by:

$$\begin{aligned} P(\langle XY \rangle) &= \int\int_{l \le Y - X \le h} f(x, y) \mathrm{d}x \mathrm{d}y \\ &= \int_{V_y(x^- + l)}^{V_y(x^+ + h)} \mathrm{d}y \int_{V_x(y-h)}^{V_x(y-l)} \frac{1}{(x^+ - x^-)(y^+ - y^-)} \mathrm{d}x \\ &= \frac{1}{S} \int_{V_y(x^- + l)}^{V_y(x^+ + h)} [V_x(y - l) - V_x(y - h)] \mathrm{d}y \end{aligned} \tag{31}$$

where $S = (x^+ - x^-)(y^+ - y^-)$ is a constant. Let $f(y) = V_x(y - h) - V_x(y - l)$. Notice that if $V_x(y - h) = x^+$, it implies that $V_x(y - l) = x^+$ because $y - l > y - h$, and then the integration in Eq. (31) equals to 0; similarly, $V_x(y - l) = x^-$ implies $V_x(y - h) = x^-$, which leads to $P(\langle XY \rangle) = 0$. Therefore, we only need to consider the remaining four cases:

$$\begin{aligned} &V_x(y - l) = x^+, V_x(y - h) = y - h && \implies \max(x^+ + l, x^- + h) \le y \le x^+ + h \\ &V_x(y - l) = x^+, V_x(y - h) = x^- && \implies x^+ + l < y < x^- + h \\ &V_x(y - l) = y - l, V_x(y - h) = y - h && \implies x^- + h < y < x^+ + l \\ &V_x(y - l) = y - l, V_x(y - h) = x^- && \implies x^- + l \le y \le \min(x^+ l, x^- + h) \end{aligned}$$

Since we also have $y^- \le y \le y^+$, the value of $f(y)$ depends on different ranges of y:

$$f(y) = \begin{cases} x^+ + h - y & \max\left(V_y(x^+ + l), V_y(x^- + h)\right) \le y \le V_y(x^+ + h) \\ x^+ - x^- & V_y(x^+ + l) < y < V_y(x^- + h) \\ h - l & V_y(x^- + h) < y < V_y(x^+ + l) \\ y - l - x^- & V_y(x^- + l) \le y \le \min\left(V_y(x^+ + l), V_y(x^- + h)\right) \end{cases} \tag{32}$$

Here we first set:

$$a_1 = V_y(x^- + l), \qquad a_2 = V_y(x^- + h)$$
$$a_3 = V_y(x^+ + l), \qquad a_4 = V_y(x^+ + h)$$

Then the range $[a_1, a_4]$ is divided into disjoint sub-partitions by two points a_2, a_3. In order to sort the values of a_1, a_2, a_3 and a_4, we set:

$$b_1 = a_1, \qquad b_2 = \min(a_2, a_3)$$
$$b_3 = \max(a_2, a_3), \qquad a_4 = b_4$$

so that we have $b_1 \le b_2 \le b_3 \le b_4$. According to *the law of total probability*, $P(\langle XY \rangle)$ can be computed by Eq. (33), since $[b_1, b_4] = [b_1, b_2] \cup [b_2, b_3] \cup [b_3, b_4]$.

$$\begin{aligned} P(\langle XY \rangle) &= \sum_{k=1}^{3} P(\langle XY \rangle \,|\, y \in [b_k, b_{k+1}]) * P(y \in [b_k, b_{k+1}]) \\ &= P(\langle XY_k \rangle) * P(Y = Y_k) \end{aligned} \tag{33}$$

where $Y_k \sim U(b_k, b_{k+1})$ is a uniform random variable and $P(Y = Y_k)$ can be computed as:

$$P(Y = Y_k) = \int_{b_k}^{b_{k+1}} \frac{1}{y^+ - y^-} \mathrm{d}y = \frac{b_{k+1} - b_k}{y^+ - y^-} \tag{34}$$

Next, we prove the correctness of Eq. (18) in three cases.

– If $Y_k \sim U[b_1, b_2]$. Now $f(y) = y - l - x^-$. According to Eq. (31), we can compute $P(\langle XY_k \rangle)$ as:

$$\begin{aligned} P(\langle XY_k \rangle) &= \frac{1}{S_k} \int_{b_1}^{b_2} y - l - x^- \mathrm{d}y \\ &= \frac{1}{2S_k} (y - l - x^-)^2 |_{b_1}^{b_2} = \frac{b_1 + b_2 - 2l - 2x^-}{2(x^+ - x^-)} \end{aligned} \tag{35}$$

Meanwhile, when $y \in [b_1, b_2]$, we know that $y - l \le x^+$ and $y - h \le x^-$. According to Eq. (19), we can compute $L_1 = b_1 - l - x^-$ and $L_2 = b_2 - l - x^-$. By substituting the values of L_1 and L_2 to Eq. (18), we can compute $P(\langle XY \rangle)$ by the geographic approach in Eq. (36), which is consistent with Eq. (35).

$$P(\langle XY_k \rangle) = \frac{L_{k+1} + L_k}{2(x^+ - x^-)} = \frac{b_1 + b_2 - 2l - 2x^-}{2(x^+ - x^-)} \tag{36}$$

– If $Y_k \sim U[b_2, b_3]$. We consider two sub-cases here.

(a) if $V_y(x^+ + l) < V_y(x^- + h)$. Referring to Eq. (31), we can compute $P(\langle XY_k \rangle)$ as follows:

$$P(\langle XY_k \rangle) = \frac{1}{S_k} \int_{V_y(x^+ + l)}^{V_y(x^- + h)} x^+ - x^- \mathrm{d}y$$
$$= \frac{(x^+ - x^-)(V_y(x^- + h) - V_y(x^+ + l))}{S_k} = 1 \tag{37}$$

Meanwhile, when $V_y(x^+ + l) \leq y \leq V_y(x^- + h)$, we have $L_2 = x^+ - x^-$ and $L_3 = x^+ - x^-$ by Eq. (19). Then, $P(\langle XY_k \rangle)$ can be computed by the geographic function in Eq. (38), which is consistent with Eq. (37).

$$P(\langle XY_k \rangle) = \frac{L_{k+1} + L_k}{2(x^+ - x^-)} = 1 \tag{38}$$

(b) if $V_y(x^- + h) < V_y(x^+ + l)$. We first have:

$$P(\langle XY_k \rangle) = \frac{1}{S_k} \int_{V_y(x^- + h)}^{V_y(x^+ + l)} (h - l) \mathrm{d}y$$
$$= \frac{(h - l)(V_y(x^+ + l) - V_y(x^- + h))}{S_k} = \frac{h - l}{x^+ - x^-} \tag{39}$$

Meanwhile, $L_2 = h - l$ and $L_3 = h - l$, since now we have $V_y(x^- + h) \leq y \leq V_y(x^+ + l)$. And the output of the geographic function in Eq. (40) is consistent with that of Eq. (39).

$$P(\langle XY_k \rangle) = \frac{L_{k+1} + L_k}{2(x^+ - x^-)} = \frac{h - l}{(x^+ - x^-)} \tag{40}$$

– If $Y_k \sim U[b_3, b_4]$. First, referring to Eq. (31), we can compute $P(\langle XY_k \rangle)$ as follows:

$$P(\langle XY_k \rangle) = \frac{1}{S_k} \int_{b_3}^{b_4} (x^+ + h - y) \mathrm{d}y = \frac{1}{2S_k} (x^+ + h - y)^2 |_{b_4}^{b_3}$$
$$= \frac{2x^+ + 2h - b_3 - b_4}{2(x^+ - x^-)} \tag{41}$$

Meanwhile, since $\max(V(x^+ + l), V(x^- + h)) \leq y \leq V(x^+ + h)$, we have $L_3 = x^+ - b_3 + h$ and $L_4 = x^+ - b_4 + h$, and then we have Eq. (42), which is consistent with Eq. (41).

$$P(\langle XY_k \rangle) = \frac{L_{k+1} + L_k}{2 * (x^+ - x^-)} = \frac{2x^+ + 2h - b_3 - b_4}{2(x^+ - x^-)} \tag{42}$$

Therefore, the correctness of our geographic approach in Sect. 5.1 is proved.

References

1. Aggarwal C, Yu P (2009) A survey of uncertain data algorithms and applications. IEEE Trans Knowl Data Eng 21(5):609–623
2. Agrawal R, Srikant R (1994) Fast algorithms for mining association rules in large databases. In: Proceedings of the 20th international conference on very large data bases VLDB'94, pp 487–499

3. Agrawal R, Srikant R (1995) Mining sequential patterns. In: Proceedings of the eleventh international conference on data engineering, ICDE '95, pp 3–14
4. Allen J (1983) Maintaining knowledge about temporal intervals. Commun ACM 26(11):832–843
5. Ayres J, Flannick J, Gehrke J et al (2002) Sequential pattern mining using a bitmap representation. In: Proceedings of the eighth ACM SIGKDD international conference on knowledge discovery and data mining, KDD '02, pp 429–435
6. Bernecker T, Kriegel H, Renz M et al (2009) Probabilistic frequent itemset mining in uncertain databases. In: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining, KDD'09, pp 119–128
7. Cheng R, Kalashnikov D, Prabhakar S (2003) Evaluating probabilistic queries over imprecise data. In: Proceedings of the ACM international conference on management of data, SIGMOD '03, pp 551–562
8. Chui C, Kao B (2008) A decremental approach for mining frequent itemsets from uncertain data. In: Proceedings of the 12th Pacific-Asia conference on advances in knowledge discovery and data mining, PAKDD'08, pp 64–75
9. Chiu D, Wu Y, Chen A (2004) An efficient algorithm for mining frequent sequences by a new strategy without support counting. In: Proceedings of the 20th international conference on data engineering, ICDE '04, pp 275–286
10. Chui C, Kao B, Hung E (2007) Mining frequent itemsets from uncertain data. In: Proceedings of the 11th Pacific-Asia conference on advances in knowledge discovery and data mining, PAKDD'07, pp 47–58
11. Dyreson C, Snodgrass R (1998) Supporting valid-time indeterminacy. ACM Trans Datab Syst 23(1):1–57
12. Ge J, Xia Y, Wang J (2015) Towards efficient sequential pattern mining in temporal uncertain databases. In: Proceedings of the 19th Pacific-Asia conference on advances in knowledge discovery and data mining, PAKDD'15, pp 268-279
13. Han J, Pei J, Mortazavi-Asl B et al (2000) Freespan: frequent pattern-projected sequential pattern mining. In: Proceedings of the sixth ACM SIGKDD international conference on knowledge discovery and data mining, KDD '00, pp 355–359
14. Höppner F (2001) Discovery of temporal patterns. learning rules about the qualitative behaviour of time series. In: Proceedings of the 5th European conference on principles of data mining and knowledge discovery, PKDD '01, pp 192–203
15. Jestes J, Cormode G, Li F et al (2011) Semantics of ranking queries for probabilistic data. IEEE Trans Knowl Data Eng 23(12):1903–1917
16. Li Y, Bailey J, Kulik L et al (2013) Mining probabilistic frequent spatio-temporal sequential patterns with gap constraints from uncertain databases. In: IEEE 13th international conference on data mining, ICDM'13, pp 448–457
17. Muzammal M, Raman R (2011) Mining sequential patterns from probabilistic databases. In: Proceedings of the 15th Pacific-Asia conference on advances in knowledge discovery and data mining, PAKDD'11, pp 210–221
18. Papapetrou P, Kollios G, Sclaroff S et al (2005) Discovering frequent arrangements of temporal intervals. In: Proceedings of the fifth IEEE international conference on data mining, ICDM '05, pp 354–361
19. Pei J, Han J, Mortazavi-asl B et al (2001) Prefixspan: mining sequential patterns efficiently by prefix-projected pattern growth. In: Proceedings of the 17th international conference on data engineering, ICDE'01, pp 215–224
20. Pei J, Han J, Wang W (2002) Mining sequential patterns with constraints in large databases. In: Proceedings of the eleventh international conference on information and knowledge management, CIKM '02, pp 18–25
21. Sadri R, Zaniolo C, Zarkesh A et al (2004) Expressing and optimizing sequence queries in database systems. ACM Trans Database Syst 29(2):282–318
22. Srikant R, Agrawal R (1996) Mining sequential patterns: generalizations and performance improvements. In: Proceedings of the 5th international conference on extending database technology: advances in database technology, EDBT '96, pp 3–17
23. Sun X, Orlowska M, Li X (2003) Introducing uncertainty into pattern discovery in temporal event sequences. In: Proceedings of the third IEEE international conference on data mining, pp 299–306
24. Sun L, Cheng R, Cheung D et al (2010a) Mining uncertain data with probabilistic guarantees. In: Proceedings of the 16th ACM SIGKDD international conference on knowledge discovery and data mining, KDD '10, pp 273–282
25. Sun L, Cheng R, Cheung D et al (2010b) Mining uncertain data with probabilistic guarantees. In: Proceedings of the 16th ACM SIGKDD international conference on knowledge discovery and data mining, KDD '10, pp 273–282
26. Wan L, Chen L, Zhang C (2013) Mining frequent serial episodes over uncertain sequence data. In: Proceedings of the 16th international conference on extending database technology, EDBT'13, pp 215–226

27. Winarko E, Roddick J (2007) Armada—an algorithm for discovering richer relative temporal association rules from interval-based data. Data Knowl Eng 63(1):76–90
28. Yang J, Wang W, Yu P et al (2002) Mining long sequential patterns in a noisy environment. In: Proceedings of the 2002 ACM SIGMOD international conference on management of data, SIGMOD '02, pp 406–417
29. Zaki M (2001) Spade: an efficient algorithm for mining frequent sequences. Mach Learn 42(1–2):31–60
30. Zhang H, Diao Y, Immerman N (2010) Recognizing patterns in streams with imprecise timestamps. Proc VLDB Endow 3(1–2):244–255
31. Zhao Z, Yan D, Ng W (2012) Mining probabilistically frequent sequential patterns in uncertain databases. In: Proceedings of the 15th international conference on extending database technology, EDBT'12, pp 74–85
32. Zhao Z, Yan D, Ng W (2013) Mining probabilistically frequent sequential patterns in large uncertain databases. IEEE Trans Knowl Data Eng 26(5):1171–1184
33. Zhou Y, Ma C, Guo Q et al (2014) Sequence pattern matching over time-series data with temporal uncertainty. In: Proceedings of the 17th international conference on extending database technology, EDBT'14, pp 205–216

Jiaqi Ge is currently a data scientist of Expedia inc. He received his Ph.D. in Computer Science from Purdue University West Lafayette, and a M.S. in Computer Engineering and a B.S. in Electrical Engineering from Nanjing University, Nanjing, China. His main research area is on data mining and machine learning, focusing on mining and modeling uncertain data and big data including sensor data, online shopping data and time series data. He also works on distributed data mining applications.

Yuni Xia is an Associate Professor of the Computer and Information Science Department at Indiana University Purdue University Indianapolis (IUPUI). She received Ph.D. and M.S. in Computer Science from Purdue University, and B.S. in Computer Science from Central China (HuaZhong) University of Science and Technology. Xia's research is on data mining and databases, focusing on mining and management of big data and data streams including biomedical data, sensor data and moving object data. She also works on managing uncertainty in the decision support process.

Jian Wang is currently an Associate Professor at the School of Electronics and Sciences, Nanjing University, Nanjing, China. He received a M.S. degree from Nanjing University of Technology, Nanjing, China, in 2003, and a Ph.D. degree from Nanjing University, Nanjing, China, in 2006. His research interests include social network, data mining, video coding and transmission and parallel computing.

Chandima Hewa Nadungodage received her B.Sc. in Computer Science and Engineering from University of Moratuwa, Sri Lanka in 2005 and M.S. in Computer and Information Science from Indiana University Purdue University Indianapolis (IUPUI) in 2013. Currently, she is a Ph.D. candidate at Department of Computer and Information Science, IUPUI. Her research interests include mining and management of evolving data streams, mining uncertain data streams, GPU-based parallel acceleration of stream mining algorithms and big-data applications.

Sunil Prabhakar is a Professor and Head of Computer Science at Purdue. His main area of research is database systems. His current research focuses on developing novel databases for handling uncertain data and solutions that ensure authenticity, integrity and privacy for outsourced databases including cloud databases. In earlier work, he has developed solutions for scalability and efficiency for moving objects and sensor databases, digital rights management for databases through digital watermarking, and ensuring correctness over private databases.