Mining Probabilistic Frequent Spatio-Temporal Sequential Patterns with Gap Constraints from Uncertain Databases

Yuxuan Li, James Bailey, Lars Kulik
Department of Computing and Information Systems
The University of Melbourne
VIC 3010, Australia
{*yuxuan.li, baileyj, lkulik*}*@unimelb.edu.au*

Jian Pei
School of Computing Science
Simon Fraser University
BC V5A 1S6, Canada
Email: jpei@cs.sfu.ca

***Abstract*—Uncertainty is common in real-world applications, for example, in sensor networks and moving object tracking, resulting in much interest in itemset mining for uncertain transaction databases. In this paper, we focus on pattern mining for uncertain sequences and introduce probabilistic frequent spatial-temporal sequential patterns with gap constraints. Such patterns are important for the discovery of knowledge given uncertain trajectory data. We propose a dynamic programming approach for computing the frequentness probability of these patterns, which has linear time complexity, and we explore its embedding into pattern enumeration algorithms using both breadth-first search and depth-first search strategies. Our extensive empirical study shows the efficiency and effectiveness of our methods for synthetic and real-world datasets.**

***Keywords*-Uncertain databases, Uncertain pattern mining, Sequential patterns, Spatial-temporal data**

I. Introduction

Sequential pattern mining is an important task in data mining and has been extensively studied [1], [2], [3], [4], [5]. Spatio-temporal sequential pattern mining in trajectory databases is one of its major variations that has also drawn a considerable amount of attention [6], [7], [8], [9], [10], [11]. A trajectory of a moving object consists of time-stamped location data across a sequence of ordered timestamps. A spatio-temporal sequential pattern in a trajectory database is a set of objects (also known as an object cluster) that move "together" for a subsequence of timestamps. This type of pattern has proved attractive in a number of real-world applications. For example, identifying common routes among convoys may lead to more effective traffic control [10], and the discovery of common movements of animals may serve as a basis for knowledge discovery in ecology [11].

The main computational challenge in mining spatio-temporal sequential patterns is the extraction of objects co-occurring at a minimum number (min_t) of consecutive timestamps. This can be seen as a generalization of the frequent itemset mining problem [12], where the order of transactions is additionally being taken into account. To provide more flexibility, the consecutiveness requirement may be relaxed and a *maximum gap constraint* imposed instead. For example, vehicles (objects) in a convoy may mostly be traveling together for consecutive timestamps, but may temporarily become separated at traffic lights (causing a gap in being together), then being together at a later time.

In the frequent itemset mining problem, items are considered as occurring "together" if they occur in the same transaction. In comparison, objects are considered as being together at a timestamp if their locations are in close proximity. A common way to determine the spatial proximity of objects is through a clustering algorithm (e.g., DBSCAN [13]). Alternatively, objects are considered as close to each other at a given timestamp, if they occur at the same point of interest (POI). Spatial proximity of objects is usually application-dependent and commonly considered as a data preprocessing step. In this paper, we consider objects as being together if they are in the same cluster according to some closeness measure. The number of timestamps where the same objects are together is analogous to the pattern's "support". A spatio-temporal sequential pattern is considered as *frequent* if there are at least min_o objects together for at least min_t timestamps and the timestamps satisfy a maximum gap constraint g.

Our work tackles the problem of mining spatio-temporal sequential patterns where the input data is uncertain.

A. Uncertainty in Pattern Mining

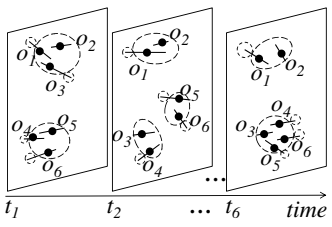
Recent research [14], [15] has shown that the modeling of uncertainty can be important in a wide range of real-world applications. Various factors contribute to data uncertainty, including incompleteness of data sources, the addition of artificial noise in privacy-sensitive applications and, most importantly, uncertainty arising from imprecision in measurements and observations. In the case of ubiquitous computing applications, object trajectory data is often acquired by position-aware devices such as GPS and WiFi systems. However, such devices have limitations in their measurement accuracy and so the location data they record is sometimes represented by a *probability-density function (pdf)* [16] to represent this uncertainty.

Two types of frequentness (or support) measurements for an uncertain pattern have been proposed: expected support [14], [17], [18] and probabilistic frequentness [19], [20],

[21]. In the former, the support of an itemset is estimated by the expected value of its support. However, as pointed out in [20], [21], one major drawback of such a measure is that it does not indicate the confidence level of the estimation, which may lead to the loss of important patterns. In [19], [20], [21], a probabilistic threshold τ based on *possible world semantics* is introduced. A pattern is considered as frequent if its probabilistic measurement exceeds τ. We will use possible world semantics to assess the frequentness of an uncertain spatio-temporal sequential pattern.

As an example scenario, Figure 1 shows moving objects in an uncertain trajectory database. In this example, the location of an object is defined by a one-dimensional *pdf* due to its positional uncertainty, which is represented as a line in the figure. The circles at each timestamp can be seen as a POI or a cluster boundary depending on the application. To compute the spatial proximity of objects with uncertain locations, one could apply first an uncertain clustering algorithm (e.g. UK-means [16]) to obtain precise (hard) clusters and then a classical pattern mining algorithm. However, similar to the use of expected support, such a method would not provide any explicit confidence measure in the result. Alternatively, fuzzy clustering [22] would assign each object a "degree of belongingness" (belongingness probability) for each cluster, i.e., one object can belong to more than one cluster. The belongingness probability is not necessarily based on positional uncertainty. Various fuzzy clustering algorithms [22], [23] handling both certain and uncertain data have been proposed to produce soft clusters.

Our proposed methods do not rely on a particular *pdf* or technique for deciding the spatial closeness of objects, which are instead modeled as preprocessing steps. We only require a set of (fuzzy) clusters as input for each timestamp, where each object is associated with a belongingness probability that specifies the confidence the object is in a cluster at a given timestamp. An example of such an input is shown in the right hand side of Figure 1. To measure the frequentness of an uncertain pattern, we need to calculate the confidence that the set of objects O is in the same cluster for at least min_t timestamps that fulfil the gap constraint g, where $|O| \geq min_o$. If the confidence exceeds a user-given probabilistic threshold τ, then this pattern is considered as probabilistic frequent. *Given an uncertain database, our problem is to find all probabilistic frequent spatio-temporal sequential patterns.*



t	Clusters
1	{o_1:0.6,o_2:1.0,o_3:0.8}; {o_4:0.7,o_5:1.0,o_6:0.8}; {o_1:0.4}; {o_3:0.2}; {o_4:0.3}; {o_6:0.2}
2	{o_1:0.8,o_2:1.0}; {o_3:1.0,o_4:0.5}; {o_5:0.7,o_6:0.5} {o_1:0.2}; {o_4:0.5}; {o_5:0.3};{o_6:0.5};
3	{o_1:0.4,o_2:1.0}; {o_3:0.8,o_4:0.5,o_5:0.6} {o_1:0.6}; {o_3:0.2}; {o_4:0.5}; {o_5:0.4}
4	{o_1:0.6,o_2:1.0}; {o_1:0.4}
5	{o_1:0.8,o_2:1.0}; {o_3:0.9,o_4:0.6,o_5:1.0} {o_1:0.2}; {o_3:0.1}; {o_4:0.4}
6	{o_1:0.4,o_2:1.0}; {o_3:1.0,o_4:0.9,o_5:0.9,o_6:0.8} {o_1:0.6}; {o_4:0.1}; {o_5:0.1}; {o_6:0.2}

Figure 1. An example of an uncertain database. The location of an object is represented as an one-dimensional *pdf* due to its positional uncertainty. Each object is associated with a belongingness probability specifying the confidence that it is in the cluster.

B. Related Work

The problem of classical sequential pattern mining has been an area of extensive research in the context of deterministic databases [1], [2], [3], [4], [5]. Several variations have been proposed. Sequential patterns in trajectory databases [6], [7], [8], [9], [10], [11] and sequential patterns in event databases [24], [25] are two popular approaches.

In the context of uncertain databases, the problem of uncertain frequent itemset mining in probabilistic databases was earlier studied under the expected support measure in [17], [14], [18], [26]. However, later [19], [20], [21] found that the use of expected support may lead to the loss of important patterns. Thus, the use of a probabilistic frequentness measure has been more popular recently. A recent survey for comparing these two measures and analyzing their relationships is given in [27]. For the problem of uncertain sequential patterns, to the best of our knowledge, the approaches in [28], [29] are the only studies in the literature. In [28], the authors measure the frequentness of a pattern in uncertain event databases based on its expected support, but this may sometimes lead to the loss of interesting patterns [20], [21]. In comparison, our methods proposed in this paper are based on probabilistic frequentness support. In a more recent work, Zhao et al. [29] addressed uncertain sequence mining under two different models of uncertainty, sequence level uncertainty and element level uncertainty. Their study is complementary, yet distinct from our work in this paper, since they focus on the case where unlimited gaps are permitted when matching a query subsequence against a longer sequence. In contrast, in our study gaps are allowed in a pattern according to a maximum gap constraint g.

C. Challenges and Contributions

Mining spatio-temporal patterns from uncertain trajectory data is a challenging problem. In particular:

- As the length of a trajectory (sequence) increases, the number of possible worlds grows exponentially. This makes naive methods for checking pattern characteristics (such as support) infeasible.
- Enabling maximum gap constraints increases flexibility for spatio-temporal patterns. However, checking pattern characteristics for uncertain data in the presence of gaps is considerably more complex than without gaps.
- Mining collections of spatio-temporal patterns for uncertain trajectory data requires a pattern enumeration framework (e.g., breadth first or depth first). The merits of different frameworks are currently an open question.

In this paper, we address these challenges. In particular, we make the following contributions:

- We define a probabilistic model for uncertain spatio-temporal sequential patterns under the possible world semantics and incorporate a maximum gap constraint.
- We introduce a linear time approach to calculate the frequentness probability of an uncertain spatio-temporal sequential pattern. This is the first such linear time result we are aware of for uncertain sequences. Previous linear time results [20] have been developed in the context of itemsets rather than sequences.
- We explore integration of our probabilistic frequentness checking algorithm within apriori-based approaches using both breadth-first and depth-first search. We analyze the relative merits of each of these approaches.

II. Problem Definition

Let $T_S = \{t_1, t_2, ..., t_n\}$ be a linearly ordered list of n timestamps (called the *time space*). Let $O_S = \{o_1, o_2, ..., o_m\}$ be a collection of m objects that appear in T_S (called the *object space*). Objects $O \subseteq O_S$ are observed at timestamps $T \subseteq T_S$ in various locations. We refer to T as a timestamp sequence and its length is $|T|$. A trajectory database D contains location data of moving objects O_S across T_S. We first provide definitions of our patterns for certain databases.

Definition 1: A set of moving objects O (called the *objectset*) that are in the same cluster for a timestamp sequence T is denoted as $S = (O : T)$ and called a **spatio-temporal sequential pattern**, where $O \subseteq O_S$ and $T \subseteq T_S$.

Definition 2: Given a sequence of timestamps T represented by $\{t_{k_1}, t_{k_2}, \ldots, t_{k_p}\}$ where $t_{k_i} \in T$, the **maximum gap** in T is calculated by $max(k_{q+1} - k_q - 1)$, $\forall q \in [1, p)$, and is denoted as $\sqcup_T$.

Example 1: For $T = \{t_1, t_2, t_4, t_7, t_8\}$ the maximum gap is $\sqcup_T = 2$.

Definition 3: **A (maximum) gap constraint** is a user-specified integer g, $g \geq 0$. We say $S = (O : T)$ fulfils the gap constraint g iff the occurrence of O in T satisfies $\sqcup_T \leq g$.

Definition 4: Given user specified values for min_o, min_t and g, $S = (O : T)$ is called a **frequent spatio-temporal sequential pattern** iff $|O| \geq min_o$, $|T| \geq min_t$ and $\sqcup_T \leq g$.

Example 2: Given $min_o = 2$, $min_t = 3$, $g = 1$, we say that $S = (o_1, o_2 : t_1, t_2, t_3, t_4, t_6)$ is a frequent spatio-temporal sequential pattern.

A. Uncertain Trajectory Database Model

We now describe our model for representing uncertainty for object trajectory data. To generalize our methods, we treat the computation of the spatial proximity of objects with uncertain locations as a preprocessing step. This enables the use of different *pdfs* for the representation of uncertainty and different clustering algorithms depending on the application. Specifically, we assume that at each timestamp t, the objects have been partitioned into a set C_t of (fuzzy) clusters.

t	Clusters C_t
1	c_{11}={o_1:0.6, o_2:1.0} c_{12}={o_1:0.4}
2	c_{21}={o_1:0.3, o_2:0.8} c_{22}={o_1:0.7} c_{23}={o_2:0.2}
3	c_{31}={o_1:1.0, o_2:1.0}

i	World	$P(W_i)$	i	World	$P(W_i)$
1	t_1:c_{11}={o_1,o_2} t_2:c_{21}={o_1,o_2} t_3:c_{31}={o_1,o_2}	0.144	5	t_1:c_{11}={o_2}, c_{12}={o_1} t_2:c_{21}={o_1,o_2} t_3:c_{31}={o_1,o_2}	0.096
2	t_1:c_{11}={o_1,o_2} t_2:c_{21}={o_1}, c_{23}={o_2} t_3:c_{31}={o_1,o_2}	0.036	6	t_1:c_{11}={o_2}, c_{12}={o_1} t_2:c_{21}={o_1}, c_{23}={o_2} t_3:c_{31}={o_1,o_2}	0.024
3	t_1:c_{11}={o_1,o_2} t_2:c_{21}={o_2}, c_{22}={o_1} t_3:c_{31}={o_1,o_2}	0.336	7	t_1:c_{11}={o_2}, c_{12}={o_1} t_2:c_{21}={o_2}, c_{22}={o_1} t_3:c_{31}={o_1,o_2}	0.224
4	t_1:c_{11}={o_1,o_2} t_2:c_{22}={o_1}, c_{23}={o_2} t_3:c_{31}={o_1,o_2}	0.084	8	t_1:c_{11}={o_2}, c_{12}={o_1} t_2:c_{22}={o_1}, c_{23}={o_2} t_3:c_{31}={o_1,o_2}	0.056

Figure 2. An example of a spatio-temporal database of uncertain clusters and its possible worlds.

An object $o \in O_S$ with an uncertain location observed at timestamp $t \in T_S$ has a belongingness probability of $P(o \in c)$ being in cluster $c \in C_t$, where $P(o \in c) \in [0,1]$ and $\sum_{c \in C_t} P(o \in c) = P(o@t) = 1$, where $P(o@t)$ is the probability that o is observed at t. Conversely, if o is not observed at t, then o is not in any cluster and $P(o@t) = 0$. We assume that different objects and different timestamps are mutually independent, i.e., the belongingness probability of an object has no effect on those of other objects. Similar probabilistic independence assumptions have been made in previous related work [14], [20]. Therefore, for objectset O, we have $P(O \in c) = \prod_{o \in O} P(o \in c)$. The probability that objects O are in the same cluster at t is $P(O@t) = \sum_{c \in C_t} P(O \subseteq c)$. We also say that $O@t$ means O occurs at timestamp t in the same cluster.

Example 3: Given t_1:c_{11} = {o_1:0.3, o_2:0.5, o_3:1.0}, c_{12} = {o_1:0.6, o_2:0.5, o_5:1.0}), c_{13} = {o_1:0.1} and O = {o_1, o_2}, then $P(o_1 \in c_{11}) = 0.3$, $P(O \in c_{11}) = 0.15$ and $P(O@t_1) = P(O \in c_{11}) + P(O \in c_{12}) = 0.45$.

We define a *spatio-temporal database of uncertain clusters* $D = \{C_{t_1}, C_{t_2}, ..., C_{t_n}\}$ as a collection of objects with uncertain locations within the clusters at timestamps $\{t_1, t_2, ..., t_n\}$. Figure 2 shows an example of D with three timestamps and two objects.

An uncertain database D can be instantiated into a possible world w that contains a collection of certain clusters at each timestamp. Suppose for an object o at each timestamp, that M is the number of clusters for which $P(o \in c) > 0$. Then there are M possible memberships for o at each timestamp. The number of possible worlds of D increases exponentially with both $|O_S|$ and $|T_S|$ (i.e. $M^{|O_S| \cdot |T_S|}$). The right hand side of Figure 2 shows all possible worlds derived from D. The probability of whether a possible world w exists is denoted as $P(w)$ and $\sum P(w) = 1$. Assuming independence, the probability that w exists is computed as:

$$P(w) = \prod_{t \in T_S} \prod_{o \in t} P(o \in c_t(o, w)) \quad (1)$$

where $c_t(o, w)$ is the cluster that o is in at timestamp t in possible world w. For example, $P(o_1 \in c_{t_1}(o_1, W_6)) = P(o_1 \in c_{12}) = 0.4$.

Example 4: In Figure 2, $P(w = W_6) = P(o_1 \in c_{t_1}(o_1,w)) \times P(o_2 \in c_{t_1}(o_2,w)) \times P(o_1 \in c_{t_2}(o_1,w)) \times P(o_2 \in c_{t_2}(o_2,w)) \times P(o_1 \in c_{t_3}(o_1,w)) \times P(o_2 \in c_{t_3}(o_2,w)) = 0.024$.

Note that the assumption of independence between objects is not always true, since in some applications, the locations of objects can be dependent. For example, two classmates might go to the lecture theatre at the same time and we can infer one location from the other's. Furthermore, the locations of an object at two consecutive timestamps are normally constrained by a speed limit, i.e., the current location of an object depends on its location at a previous timestamp. More sophisticated models using conditional probabilities (e.g. a Markov model) might be used in these scenarios. If such knowledge is available, it can be integrated into our techniques, by modifying Equation 1.

B. Probabilistic Frequent Spatio-Temporal Sequential Patterns

Recall that given thresholds min_o, min_t and g, we say $S = (O : T)$ is frequent if $|O| \geq min_o$, $|T| \geq min_t$ and $\sqcup_T \leq g$. In the uncertain scenario, co-occurrence of the objects at the timestamps T is no longer certain. Instead co-occurrence is described by a *discrete probability-distribution function (d-pdf)*. For example, in Figure 2, the *d-pdf* of T for objectset $O = \{o_1, o_2\}$ is: $P(T = \{t_1,t_2,t_3\}) = P(W_1) = 0.144$, $P(T = \{t_1,t_3\}) = P(W_2) + P(W_3) + P(W_4) = 0.456$, $P(T = \{t_2,t_3\}) = P(W_5) = 0.096$, $P(T = \{t_3\}) = P(W_6) + P(W_7) + P(W_8) = 0.304$.

Let T be the timestamps that objects of O are in the same cluster, the probability that $|T| \geq min_t$ and $\sqcup_T \leq g$ is called the *frequentness probability* and it is denoted as $P^g_{\geq min_t}(O)$. The frequentness probability is interpreted as the probability that objects of O are in the same cluster for at least min_t timestamps that satisfy gap constraint g. An uncertain spatio-temporal sequential pattern is denoted as $\widehat{S} = O$ (co-occurrence of the objects at T is uncertain). Using a confidence threshold τ, we formally define a *probabilistic frequent spatio-temporal sequential pattern* as follows:

Definition 5: $\widehat{S} = O$ is called a **probabilistic frequent spatio-temporal sequential pattern** iff $|O| \geq min_o$ and $P^g_{\geq min_t}(O) \geq \tau$.

Definition 6 (Problem definition): Find for a spatio-temporal database D of uncertain clusters the complete set of probabilistic frequent spatio-temporal sequential patterns.

The main computational challenge of our mining task is the calculation of frequentness probabilities for candidate patterns. This is detailed in the next section.

III. Calculating Frequentness Probabilities

A simple way to compute the frequentness probability is to enumerate all possible worlds of D and sum up the probabilities of possible worlds with $|T| \geq min_t$ and $\sqcup_T \leq g$:

$$P^g_{\geq min_t}(O) = \sum_{w \in W:(|T| \geq min_t, \sqcup_T \leq g)} P(w) \quad (2)$$

Table I
Summary of the use of notations.

$\sqcup_T$	The maximum gap produced by T.
$\vee_{T,j}$	The tail gap produced by T at the first j timestamps.
T_j	First j timestamps of T_S, and $T_j = \{t_1,t_2,...,t_j\}$.
$P(O@t)$	Probability that objects in O are in the same cluster at t, i.e. "O occurs at t".
$P(O\overline{@}t)$	Probability that objects in O are NOT in the same cluster at t.
$P^g_{\geq min_t}$	Probability that O occurs at least min_t timestamps of T_S with gap constraint g.
$P^x_{\geq i,j}$	Probability that O occurs at least i timestamps of T_j with gap constraint x.
$P^{\vee y,x}_{\geq i,j}$	Probability that O occurs at least i timestamps of T_j with tail gap constraint y and gap constraint x.

Example 5: In Figure 2, if we set $min_t = 2$ and $g = 1$, for pattern $S = (o_1,o_2 : P^1_{\geq 2}(O))$, we have $P^1_{\geq 2}(O) = P(W_1) + P(W_2) + P(W_3) + P(W_4) + P(W_5) = 0.696$.

As mentioned before, as the size of D increases, the number of possible worlds increases exponentially, up to $|W| = M^{|T_S| \times |O_S|}$ where M is the number of clusters for which $P(o \in c) > 0$. With the assumption that timestamps in T_S are mutually independent [14], [20], we can simplify the calculation as follows:

$$P^g_{\geq min_t}(O) = \sum_{T \subseteq T_S:(|T| \geq min_t, \sqcup_T \leq g)} \left(\prod_{t \in T} P(O@t) \times \prod_{t \in T_S - T} P(O\overline{@}t)\right) \quad (3)$$

where $P(O\overline{@}t) = 1 - P(O@t)$. The computation of $P^g_{\geq min_t}(O)$ via Equation 3 needs to enumerate $\sum_{i=min_t}^{|T_S|} \binom{T_S}{i}$ combinations of T where $\sqcup_T \leq g$, which is still inefficient and the computation cost still increases exponentially with respect to $|T_S|$ (the total length of the trajectory).

Example 6: In Figure 2, to calculate $P^1_{\geq 2}(O)$ for pattern $S = (o_1,o_2 : P^1_{\geq 2}(O))$, the combinations of T fulfilling $|T| \geq 2, \sqcup_T \leq 1$ are $\{t_1,t_2\}$, $\{t_2,t_3\}$, $\{t_1,t_3\}$ and $\{t_1,t_2,t_3\}$. $P(O@t_1) = 0.6$, $P(O@t_2) = 0.24$, $P(O@t_3) = 1.0$. Thus, $P^1_{\geq 2}(O) = P(O@t_1) \times P(O@t_2) \times P(O\overline{@}t_3) + P(O\overline{@}t_1) \times P(O@t_2) \times P(O@t_3) + P(O@t_1) \times P(O\overline{@}t_2) \times P(O@t_3) + P(O@t_1) \times P(O@t_2) \times P(O@t_3) = 0.696$.

A. A Dynamic Programming Approach

To avoid the exhaustive enumeration of possible worlds (or different combinations of mutually independent timestamps), we propose a dynamic programming approach to efficiently calculate the frequentness probability. We first introduce some notations, which are summarized in Table I.

Let $T_j = \{t_1,t_2,...,t_j\}$ be the first j timestamps of T_S, where $T_j \subseteq T_S$. The probability that O occurs at least i

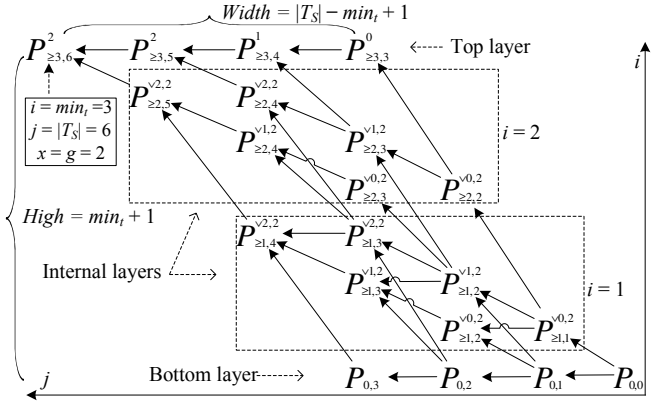


Figure 3. Details of how to calculate $P^{2}_{\geq 3,6}(O)$ using our dynamic programming approach (4 layers in total).

timestamps of T_j with gap constraint x is:

$$P^{x}_{\geq i,j}(O) = \sum_{T \subseteq T_j : (|T| \geq i, \sqcup_T \leq x)} \left(\prod_{t \in T} P(O@t) \times \prod_{t \in T_S - T} P(O\overline{@}t)\right) \tag{4}$$

Thus, the frequentness probability is given by $P^{g}_{\geq min_t, |T_S|}(O)$.

1) Tail gap: The treatment of gap constraints is a central feature of our approach. To develop a dynamic programming method that can handle the gap constraint, we introduce the notion of a *tail gap*.

Definition 7: Given a sequence of timestamps T represented by $\{k_1, k_2, ..., k_p\}$, where $t_{k_i} \in T$, and given $T_j = \{t_1, t_2, ..., t_j\}$ where $T \subseteq T_j \subseteq T_S$. The **tail gap** produced by T at the first j timestamps, denoted as $\vee_{T,j}$, is defined as $\vee_{T,j} = j - k_p$.

Example 7: For $T = \{t_1, t_2\}$ and $T_6 = \{t_1, ..., t_6\}$, $\vee_{T,6} = 6 - 2 = 4$.

Definition 8: Given a positive integer y and $T_j = \{t_1, t_2, ..., t_j\}$ where $T_j \subseteq T_S$, we say $T \subseteq T_j$ fulfills the **tail gap constraint** iff T satisfies $\vee_{T,j} \leq y$.

Definition 9: $P^{\vee y,x}_{\geq i,j}(O)$ is defined as the probability that O occurs at least i timestamps of T_j with tail gap constraint $\vee y$ and gap constraint x.

Lemma 1 shows one important property of $P^{\vee y,x}_{\geq i,j}(O)$ which will be used in our dynamic programming approach.

Lemma 1: [1] $P^{\vee y,x}_{\geq i,j}(O) = P^{\vee (j-i),x}_{\geq i,j}(O)$, $\forall y > j - i$.

Example 8: $P^{\vee 2,2}_{\geq 3,4}(O) = P^{\vee 1,2}_{\geq 3,4}(O)$.

2) A dynamic programming scheme: Let T be the timestamps that O occurs at the first j timestamps $T_j = \{t_1, t_2, ..., t_j\}$, then $P^{g}_{\geq i,j}(O)$ is equal to the probability that T fulfills $|T| \geq i$ and $\sqcup_T \leq g$. Our dynamic programming approach is to split the problem of computing $P^{g}_{\geq i,j}(O)$ at the first j timestamps into subproblems of computing the frequentness probabilities at the first $j-1$ timestamps. Let T' be the timestamps that O occurs at the first $j-1$ timestamps T_{j-1}. The information we require is what conditions must be met by T' to ensure $|T| \geq i$ and $\sqcup_T \leq g$.

The conditions T' needs to meet to make $|T| \geq i$ as follows. If $O@t_j$, then O must occur at least $i-1$ timestamps of T_{j-1} ($|T'| \geq i-1$). Otherwise, O must occur at least i timestamps of T_{j-1} ($|T'| \geq i$). This technique has also been used in previous work on probabilistic top-k queries [30]. However, we take the order into account and the maximum gap is allowed in a sequence. Thus, techniques for handing the gap constraint are required, which leads to Lemma 2.

Lemma 2 shows the conditions T' needs to meet to make $\sqcup_T \leq g$ as follows. If $O@t_j$, T' needs to satisfy both gap constraint $\sqcup_{T'} \leq g$ and tail gap constraint $\vee_{T',j-1} \leq g$. Otherwise, T' needs to satisfy gap constraint $\sqcup_{T'} \leq g$.

Lemma 2: Let T and T' be the timestamps that O occurs at T_j and T_{j-1} respectively. If $O@t_j$, $\sqcup_T \leq g \Leftrightarrow (\sqcup_{T'} \leq g$ and $\vee_{T',j-1} \leq g)$. Otherwise, $\sqcup_T \leq g \Leftrightarrow \sqcup_{T'} \leq g$.

With the above discussions, we can split the problem of computing $P^{g}_{\geq i,j}(O)$ as follows. If $O@t_j$, then $P^{g}_{\geq i,j}(O)$ is equal to the probability that T' fulfills thresholds $|T'| \geq i-1$, $\sqcup_{T'} \leq g$ and $\vee_{T',j-1} \leq g$. Otherwise, $P^{g}_{\geq i,j}(O)$ is equal to the probability that T' fulfills thresholds $|T'| \geq i$ and $\sqcup_{T'} \leq g$. Lemma 3 shows how to use the dynamic programming scheme to compute $P^{g}_{\geq min_t, |T_S|}(O)$.

Lemma 3: Entry: $x = g, i = min_t, j = |T_S|$

$$P^{x}_{\geq i,j}(O) = P^{\vee x,x}_{\geq i-1,j-1}(O) \times P(O@t_j) + P^{x}_{\geq i,j-1}(O) \times P(O\overline{@}t_j) \tag{5}$$

where if $1 \leq i < min_t$:

$$P^{\vee y,x}_{\geq i,j}(O) = P^{\vee x,x}_{\geq i-1,j-1}(O) \times P(O@t_j) + P^{\vee y-1,x}_{\geq i,j-1}(O) \times P(O\overline{@}t_j) \tag{6}$$

$$\text{or if } \; i = 0: \; P^{\vee y,x}_{\geq i,j}(O) = P_{0,j}(O) + P^{\vee y,x}_{\geq 1,j}(O) \tag{7}$$

recursion termination conditions:

$$P^{\vee y,x}_{\geq 0,0}(O) = P_{0,0}(O) = 1; \tag{8}$$

$$P^{\vee y,x}_{\geq i,j}(O) = P^{x}_{\geq i,j}(O) = 0, \forall i > j \text{ or } \forall x < 0, y < 0; \tag{9}$$

Lemma 3 is explained as follows. Equation 5 is the entry of our dynamic programming approach with $x = g, i = min_t$ and $j = |T_S|$. Then $P^{\vee x,x}_{\geq i-1,j-1}(O)$ of Equation 5 is calculated by Equation 6 if $1 \leq i < min_t$ or by Equation 7 if $i = 0$. $P_{0,j}(O)$ of Equation 7 is the probability that objects in O are not in the same cluster at the first j timestamps. It is calculated by $P_{0,j}(O) = \prod_{1 \leq k \leq j} P(O\overline{@}t_k) = P_{0,j-1}(O) \times P(O\overline{@}t_j)$. These equations are calculated recursively. The recursion termination conditions are shown in Equation 8 and Equation 9.

Figure 3 shows an example of how to use our dynamic programming approach to calculate $P^{2}_{\geq 3,6}(O)$. Theoretically, the equations in Lemma 3 can be used as a top-down approach. However, this approach leads to repeated calculations of internal results. For example, in Figure 3, $P^{\vee 1,2}_{\geq 2,3}(O)$ is used by both $P^{1}_{\geq 3,4}(O)$ and $P^{\vee 2,2}_{\geq 2,4}(O)$. Instead, as a dynamic programming method, we use a bottom-up

[1]The proofs of the lemmas in this paper can be found on http://www.csse.unimelb.edu.au/~jbailey/stsp.pdf.

approach. As shown in Figure 3, we start from the bottom layer, and store internal results for further calculations until we reach the leftmost node of the top layer. In each layer, $j \in [i, |T_S| - min_t + i]$, thus the width of each layer is $|T_S| - min_t + 1$. For $min_t > 1$, there are $min_t + 1$ layers in total. For $min_t = 1$, we still need one internal layer, i.e., three layers in total.

Figure 4 (a) provides a zoom in of the internal layer $i = 1$. In each column j, only those $P^{\vee y,x}_{\geq i,j}(O), y \in [\vee y_{min}, \vee y_{max}]$ need to be calculated and stored. The rightmost node of each row gets the $\vee y_{max}$ for its column, where $\vee y_{max} = min(j - i, g)$ (Lemma 2). For example, $P^{\vee 2,2}_{\geq 1,3}(O)$ gets $\vee y_{max} = 2$ for column $j = 3$. Whilst the leftmost node of each row gets $\vee y_{min}$ for its column, which is calculated as follows. j of the upper-left node is $|T_S| - min_t + i$ where y_{min} get the maximum value of $\vee y_{min} = g$. The difference in j between the upper-left node and other leftmost nodes is $(|T_S| - min_t + i) - j$. Then y_{min} of each leftmost node is equal to g minus that difference. Also, $\vee_{y,j} \geq 0$. Thus, $\vee y_{min} = max(g - (|T_S| - min_t + i - j), 0)$. For example, $P^{\vee 1,2}_{\geq 1,3}(O)$ gets $\vee y_{min} = 2 - (4 - 3) = 1$ for column $j = 3$.

As shown in Figure 4 (a), each internal layer can be seen as a parallelogram with $width = |T_S| - min_t + 1 - g$ and $height = g + 1$. Thus, the number of nodes in each layer is given by $(|T_S| - min_t + 1 - g) \times (g + 1) = -g^2 + (|T_S| - min_t) \times g + |T_S| - min_t + 1$. It is a quadratic function that reaches its peak for $g = (|T_S| - min_t)/2$. Figure 4 (b) shows the number of nodes per internal layer for calculating $P^g_{\geq 5,20}(O)$, $g \in [0, 15]$. $\#node$ is minimal when $g = 0$ or $g = |T_S| - min$, and maximal when $g = (|T_S| - min_t)/2$.

Intuitively, $g = 0$ means all the timestamps of T are consecutive without a gap. While $g = |T_S| - min$ means there is no gap constraint with T. In this case, timestamps are treated as a set and can be seen as transactions in frequent itemset mining.

We now state a key theorem for the computational complexity of our method to compute probabilistic frequentness.

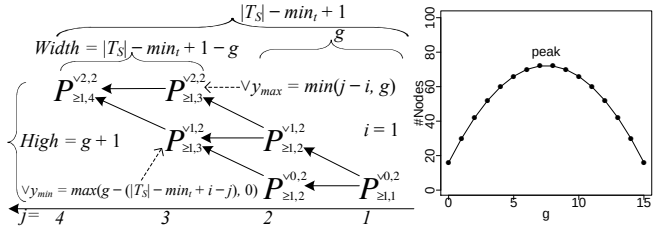
Theorem 1: [2] Both the time complexity and space complexity for calculating $P^g_{\geq min_t, |T_S|}(O)$ are $O(|T_S|)$.

Proof: To calculate $P^g_{\geq min_t, |T_S|}(O)$, we need one bottom layer and one top layer, and $min_t - 1$ internal layers (or one internal layer for $min_t = 1$). Thus the total number of nodes needed is $2 \cdot (|T_S| - min_t + 1) + (min_t - 1) \cdot (|T_S| - min_t + 1 - g) \cdot (g + 1)$ for $min_t > 1$, or $2 \cdot (|T_S| - min_t + 1) + (|T_S| - min_t + 1 - g) \cdot (g + 1)$ for $min_t = 1$. Thus, it requires $O(min_t \cdot g \cdot |T_S| - min_t^2 \cdot g - min_t \cdot g^2) = O(|T_S|)$ time and $O(|T_S|)$ space. ■

Our dynamic programming approach can check probabilistic support in linear time with respect to the length of the trajectory T_S. This is a substantial improvement over the naive approach of enumerating all possible worlds.

Example 10 in Appendix shows how to use our approach to calculate $P^2_{\geq 3,6}(O)$ from Figure 1, where $O = \{o_1, o_2\}$.

[2] Assuming parameters min_t and g are constants, as was done in [20].



(a) Zoom in of $i = 1$ (b) #Nodes with respect to g

Figure 4. (a) Zoom in of the internal layer of $i = 1$ in Figure 3. The shape can be considered as a parallelogram. (b) Number of nodes in each internal layer of $P^g_{\geq 5,20}(O)$ with respect to maximum gap constraint g.

B. Probabilistic Monotonicity Discussion

In this subsection, we explore and discuss probabilistic monotonicity criteria related to our formulation. We also show that how these monotonicity criteria can be used as a probabilistic pruning technique to further reduce the computation cost of our dynamic programming approach.

Lemma 4: $P^{\vee y+1,x}_{\geq i,j}(O) \geq P^{\vee y,x}_{\geq i,j}(O)$.

Lemma 4 indicates that increasing the tail gap allowed in a sequence increases the chance a pattern fulfills the tail gap constraint. In contrast, Lemma 5 shows that increasing the minimum number of timestamps required in a sequence decreases the chance a pattern fulfills min_t threshold.

Lemma 5: $P^{\vee y,x}_{\geq i,j}(O) \geq P^{\vee y,x}_{\geq i+1,j}(O)$.

Compared to the support monotonicity criterion of uncertain frequent itemset mining [20], Lemma 5 extends the criterion by applying gap constraints to two sides of the equation. Based on Lemma 4 and Lemma 5, we have:

Lemma 6: $P^{\vee x,x}_{\geq i,j}(O) \geq P^{\vee x,x}_{\geq i+1,j+1}(O)$.

Lemma 6 suggests that the frequentness probability of the upper-left node of the ith internal layer is no less than that of the upper-left node of the $(i+1)$th internal layer.

Lemma 7: $P^{\vee x,x}_{\geq i,j}(O) \geq P^{x}_{\geq i+1,j+1}(O)$, if $x = j - i$.

Lemma 7 indicates that, under the setting of $x = j - i$, the frequentness probability of the upper-left node of the $(min_t - 1)$th internal layer is no less than that of the leftmost node of the top layer. Recall that only patterns with $P^g_{\geq min_t, |T_S|}(O) \geq \tau$ are of interest in our task. Together with Lemma 6, we now have the following probabilistic pruning rule.

Pruning Rule 1: During the bottom-up computation of $P^g_{\geq min_t, |T_S|}(O)$, where $g = |T_S| - min_t$, if the frequentness probability of the upper-left node of an internal layer $P^{\vee g,g}_{\geq min_t - k, |T_S| - k}(O)$, where $1 \leq k < min_t$, is less than τ, then this pattern can be pruned.

IV. Mining Probabilistic Frequent Spatio-temporal Sequential Patterns

We introduced a linear time solution to calculate the frequentness probability. We now need to efficiently enumerate different combinations of objectsets with $|O| \geq min_o$ in O_S to find all patterns which are probabilistic frequent in an

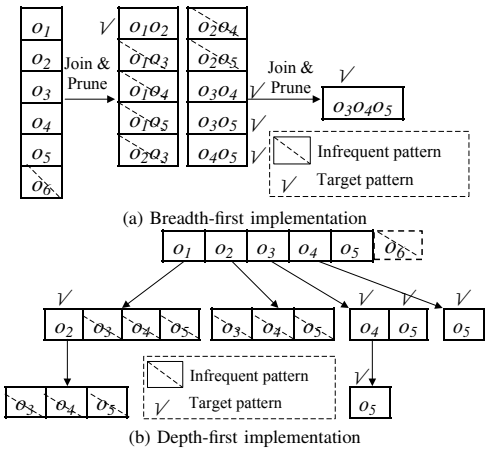


Figure 5. Two implementations of Apriori-based algorithms running on the database in Figure 1.

uncertain database. We discuss both breadth-first and depth-first implementations of Apriori-based algorithms.

Breadth-first implementation: It has been shown that the Apriori algorithm [12], which uses a breadth-first strategy, is promising for identifying frequent itemsets in both certain and uncertain transaction databases [20]. The anti-monotonicity of support, also called the Apriori property, can reduce the search space and thus speed up the mining process. In order to apply an Apriori-based algorithm, we first need to show that the anti-monotonicity property still holds for the frequentness probability with gap constraint g.

***Lemma** 8:* $P^g_{\geq min_t}(O) \geq P^g_{\geq min_t}(O')$, $\forall O \subseteq O'$.

Lemma 8 provides a pruning technique for bottom-up pattern searching. That is, if $P^g_{\geq min_t}(O)$ does not fulfill the probabilistic threshold, then $P^g_{\geq min_t}(O')$ does not fulfill the probabilistic threshold if $O \subseteq O'$. We call this the Apriori pruning rule. Similar to the classical Apriori algorithm, our algorithm using breadth-first implementation also involves two main steps: a join step and a prune step. The join step is responsible for generating new candidates. In the prune step we calculate the frequentness probability for each candidate and extract probabilistic frequent patterns where $|O| \geq min_t$. We start with an objectset containing a single object, then iteratively generate new candidates by a join operation. In each iteration, we scan the database to calculate the frequentness probability for each candidate. Next, we output those patterns satisfying $|O| \geq min_t$ and probabilistic support. Finally, we use the Apriori pruning rule to eliminate those candidates with $P^g_{\geq min_t}(O) < \tau$.

Example 9: Patterns discovered from the database in Figure 1 with the setting of $min_o = 2$, $min_t = 3$, $g = 2$, $\tau = 0.1$: $\{o_1, o_2 : 0.82\}$,$\{o_3, o_4 : 0.43\}$,$\{o_3, o_5 : 0.39\}$,$\{o_4, o_5 : 0.24\}$,$\{o_3, o_4, o_5 : 0.10\}$.The process is shown in Figure 5 (a).

Depth-first implementation: One bottleneck of the Apriori algorithm is that the join step becomes time-consuming for a large number of candidates. In our depth-first implementation, we simplify the candidate generation by simply adding one object to the k-objectset to form the $(k+1)$-objectset, which does not require a join operation. Such a search order also carries over from a smaller objectset to a larger objectset. It means the depth-first implementation also can take the advantage of the Apriori property in the generation of candidate $(k+1)$-objectset from k-objectset.

Algorithm 1 Depth-first implementation

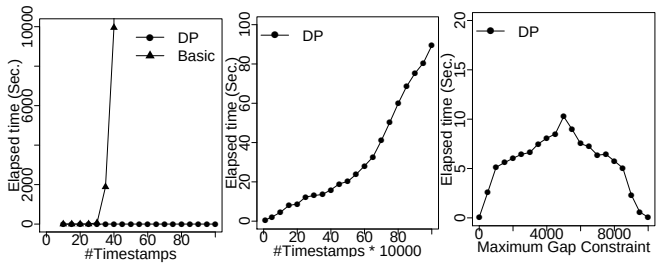
```
1: Calculate the frequentness probability p for each 1-objectset and insert into O_Feq if p ≥ τ
2: m ← maximum object in O_Feq
3: Call Extend({o}, m, O_Feq) for each o ∈ O_Feq
4: Function Extend(O, m, O_Feq)
5: n ← maximum object in O
6: for i ∈ O_Feq and n < i ≤ n do
7:    O ← O ∪ {i}
8:    p ← CalculateFrequentnessProbability(O, D, min_t, g)
9:    if p ≥ τ then {//Apriori property}
10:      OUTPUT (O : p), if |O| ≥ min_o
11:      Extend(O, m, O_Feq)
```

Algorithm 1 shows the depth-first implementation. We first select frequent 1-objectsets (line 1), and then recursively generate candidate $(k+1)$-objectset from k-objectset (line 5-12). At each iteration, only the frequent k-objectsets are extended (Apriori property, line 10). We use a bucket tree structure to store candidate patterns and Figure 5 (b) shows an example of using the depth-first implementation to find patterns from the database in Figure 1. The tree grows from left to right. Each bucket store a pattern represented by the path from root. For example, the bottom-left bucket represents the objectset $\{o_1, o_2, o_3\}$. The tree stops growing once it encounters the infrequent pattern.

Compared with the former implementation, the depth-first strategy does not fully use the downward closure of the probabilistic support. This is due to the fact that the depth-first implementation does not know all frequent k-objectsets before considering the $(k+1)$-objectset. This may lead to a bigger search space. For example, Figure 5 (b) needs to calculate the frequentness probability of 20 candidates compared to 17 candidates of Figure 5 (a). The frequentness probabilities of three candidates $\{o_1, o_2, o_3\}$, $\{o_1, o_2, o_4\}$ and $\{o_1, o_2, o_5\}$ are not calculated by the former implementation since they contain infrequent subset $\{o_2, o_3\}$, $\{o_2, o_4\}$ and $\{o_2, o_5\}$ respectively.

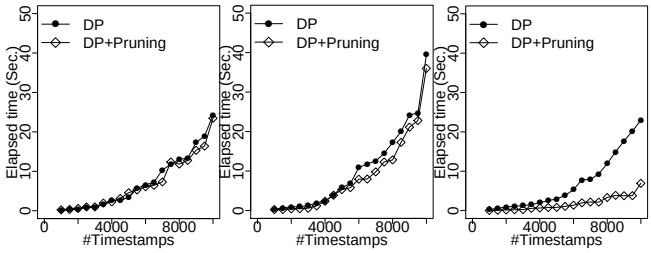
V. Experiments

We first consider large synthetic datasets to test the efficiency of our dynamic programming approach for computing the frequentness probability. Then, we use both synthetic datasets and real-world vehicle tracking datasets to evaluate the effectiveness of our two Apriori-based algorithms. All



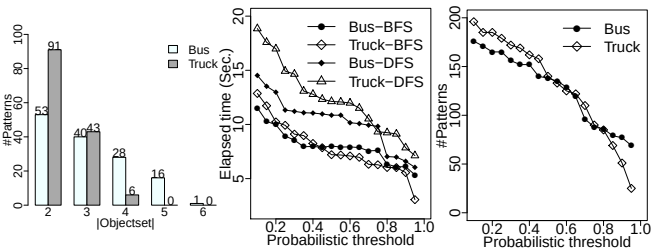
(a) DP method vs. Basic method (b) Scalability of DP method (c) Elapsed time with respect to g

Figure 6. Synthetic Dataset: (a) Dynamic programming approach vs. Basic method. (b) Scalability of DP method. (c) Effect of maximum gap g for DP method.



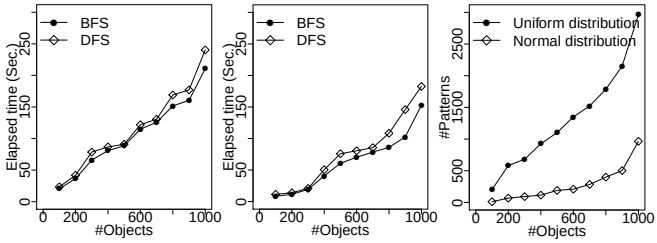
(a) $min_t = |T_S| \times 25\%$ (b) $min_t = |T_S| \times 50\%$ (c) $min_t = |T_S| \times 75\%$

Figure 7. Synthetic Dataset: Effect of probabilistic pruning rule on elapsed time with respect to min_t.



(a) #Patterns with respect to $|O|$ (b) Elapsed time with respect to τ (c) #Patterns with respect to τ

Figure 8. The number of patterns discovered from two real-world datasets and the effect of τ.



(a) Elapsed time with respect to #objects using uniform distribution (b) Elapsed time with respect to #objects using normal distribution (c) #Patterns with respect to #objects

Figure 9. Experiments on the datasets generated by the Brinkhoff simulator.

experiments were performed on an Intel Core i5 2.3GHz machine with 8GB main memory.

A. Evaluation on Calculating the Frequentness Probability

We first use synthetic datasets with varying parameter settings to test the performance of our dynamic programming approach. The size of the datasets varies from 10 timestamps to 10^6 timestamps. The probability that an objectset O occurs at t is randomly chosen as $P(O@t) \in (0,1)$. All results are the average of ten runs. The probability threshold τ is 0.9. Both min_t and gap constraint g were set to 10 unless stated otherwise. Abbreviations used in our figures: (1) Basic: basic approach of probability calculation (c.f. Equation 3); (2) DP: our dynamic approach; (3) DP+Pruning: our dynamic approach with pruning (c.f. Section III-B); (4) BFS: the breadth-first implementation of our algorithm; (5) DFS: the depth-first implementation of our algorithm.

Figure 6 (a) compares the performance of our dynamic programming approach against the basic method. The elapsed time of the basic method increases exponentially when $|T_S|$ increases. The elapsed time of the basic method is out of the scale from the figure if $|T_S|$ is less than 50. In fact, the total number of combinations of timestamps in the basic method has already reached $\binom{50}{10} > 10^{10}$. Thus the basic method is not practical. The scalability of our proposed dynamic approach is illustrated in Figure 6 (b). The performance of proposed approach is very promising: as T_S increases, the running time grows in a linear trend as in line with our time complexity theorem (c.f. Theorem 1).

Effect of the Gap Constraint g: To isolate the effect of the maximum gap constraint g on the running time, we turn off the minimum timestamp threshold by setting $min_t = 1$. This minimizes the effect of min_t on the running time. The running time for calculating $P^g_{\geq 1,10^4}(O)$ is shown in Figure 6 (c), where g varies from 0 to 9999. The running time reaches a peak at $g = 5000$ (the mid point) which is consistent with our analysis from Figure 4 (b).

Effect of Probabilistic Punning Rule: In Section III-B, we showed that the probabilistic pruning rule can be applied if $g = |T_S| - min_t$. Using this setting for g, we test the running time with three different values of min_t as shown in Figure 7. Setting a strict $min_t = |T_S| \times 75\%$ threshold, the probabilistic pruning rule shows a significant speed up. In contrast, if we apply a relatively loose min_t, then the two lines in Figure 7 (a) overlap, since patterns in such a setting more likely fulfill the probabilistic threshold and cannot be pruned.

B. Evaluation on Mining Probabilistic Frequent Spatio-temporal Sequential Patterns

Two real-world vehicle traffic datasets[3] and synthetic datasets generated by the Brinkhoff generator[4] were used to evaluate our two Apriori-based algorithms for mining probabilistic spatio-temporal frequent sequential patterns.

1) Traffic Datasets: The settings for our two real-world dataset are as follows: (1) a bus dataset recording 2 school

[3]http://www.rtreeportal.org

[4]http://www.fh-oow.de/institute/iapg/personen/brinkhoff/

buses collecting and delivering students around Athens for 108 days and consisting of 145 trajectories; (2) a truck dataset recording 50 trucks delivering concrete to construction sites around Athens over 33 days and consisting of 276 trajectories. To increase the size of moving objects, we considered each distinct trajectory as the ID of an object, yielding 145 buses with 1713 timestamps and 276 trucks with 2449 timestamps. This is a common pre-processing method [10]. The timestamp update frequency was set to every 30 seconds. Any second of a timestamp falling into the range [0", 30") was normalized to 15" and to 45" otherwise. For example, the timestamp 23:22:22 gets normalized to 23:22:15 and 23:22:58 to 23:22:45. The soft clusters at each timestamp are obtained by the fuzzy c-means clustering algorithm [22] with $m = 2$ and $EPS = 0.01$, where m is the weighting exponent and EPS is the termination criterion. Each object is assigned a belongingness probability by the fuzzy clustering algorithm. The number of clusters at each timestamp is drawn from [2, 5].

We first ran our algorithms on the datasets with $min_o = 2$, $min_t = g = 10$ and $\tau = 0.5$. The number of found patterns is illustrated in Figure 8 (a). The largest object cluster we discovered from the bus dataset has 6 objects, compared to 4 for the truck dataset. In addition, 2-objectsets and 3-objectsets are a dominant proportion of the patterns found in the truck dataset, whilst the size of objectsets in the bus dataset is more uniform. We further test the effect of the probabilistic threshold τ on the running time and number of patterns found, see Figure 8 (b) and (c) respectively. As τ increases, the Apriori rule has more effect and thus the running time of both implementations decreases. The effect is more obvious for the truck dataset. With a smaller search space, the breadth-first implementation outperforms the depth-first implementation for both datasets. In Figure 8 (c), the truck dataset returns more patterns than the bus dataset does for lower τ. However, as τ increases, the number of patterns found in both datasets decreases significantly especially for the truck dataset. There are only 25 patterns found in the truck dataset with $\tau = 0.95$.

2) Brinkhoff Generator: Synthetic Datasets: We use the map of Oldenburg as input map data. To control the exact size of the objectset we test, we vary the number of objects from 100 to 1000 and set the number of newly generated objects at each timestamp to zero. The maximum number of timestamps is set to 10000. To make moving objects last longer (thus the data has more timestamps), we set the speed divided by 250 which is the default value for slow. Other parameters were set as default. In order to test the effect of the probabilistic distribution of belongingness probabilities on our algorithms, we assign the belongingness probabilities to objects manually, rather than by applying a fuzzy clustering algorithm. We first apply DBSCAN [13] with $MinPoints = 2$ and $\varepsilon = 0.05$ to obtain (certain) clusters at each timestamp. $MinPoints$ denotes the minimum number of objects in a cluster with a radius of ε. Then, we assign the belongingness probability to each object. The belongingness probabilities of these datasets were assigned according to two different distributions: (1) Each object was assigned a probability according to a uniform distribution in the range of (0.5, 1.0]. (2) Each object was assigned a probability according to a normal distribution with a mean of 0.5 and a standard deviation of 0.2 in the range of [0, 1.0] (if the probability is outside the range, we assign the boundary value of 0 or 1). The parameters of our algorithms are set as $min_o = 2$, $min_t = g = 10$ and $\tau = 0.5$. The results are shown in Figure 9. Again, the algorithm using the breadth-first implementation generally outperforms the depth-first implementation. The elapsed time for each of the two different probabilistic distributions is not significantly different. However, there are many more patterns found in the datasets that use a uniform distribution than the datasets with a normal distribution as shown in Figure 9 (c).

VI. Conclusions

In this paper, we have formulated and studied the problem of mining probabilistic frequent spatio-temporal sequential patterns in uncertain databases. We proposed a dynamic programming approach for computing the frequentness probability with linear time complexity. This is a somewhat surprising result. Linear time support checking has been shown to be possible for uncertain itemsets. However, sequences have a more complex structure than itemsets and gap constraints add even further complexity.

We further introduced and evaluated two Apriori-based algorithms using breadth-first and depth-first implementations for efficient enumeration of all probabilistic frequent spatio-temporal sequential patterns from uncertain databases.

In our further study, we aim to extend our current approach to be able to handle the trajectory data where the identity of objects is uncertain.

Acknowledgment

This research is funded by the Australian Research Council's Discovery Grant (project number DP110100757).

Appendix

Example 10: The calculation of $P^2_{\geq 3,6}(O)$ in Figure 1, where $O = \{o_1, o_2\}$ and $P(O@t_i) = \{0.6, 0.8, 0.4, 0.6, 0.8, 0.4\}$. **Note that** here $P^x_{\geq i,j}$ is short for $P^x_{\geq i,j}(O)$, and $P(t_i)$ ($P(\overline{t_i})$) is short for $P(O@t_i)$ ($P(O\overline{@}t_i)$).

(1) Bottom layer: $i = 0,\ j \in [0,3]$

$P_{0,0} = 1$; $P_{0,1} = P_{0,0} \times P(\overline{t_1}) = 0.4$;

$P_{0,2} = P_{0,1} \times P(\overline{t_2}) = 0.8$; $P_{0,3} = P_{0,2} \times P(\overline{t_3}) = 0.48$

(2) Internal layer: $i = 1,\ j \in [1,4]$

(2.1) $j = 1, \vee y_{min} = \vee y_{max} = 0$

$P^{\vee 0,2}_{\geq 1,1} = P^{\vee 2,2}_{\geq 0,0} \times t_1 + P^{\vee -1,2}_{\geq 1,0} \times \overline{t_1} = 0.6$

(2.2) $j = 2, \vee y_{min} = 0, \vee y_{max} = 1$

$P^{\vee 0,2}_{\geq 1,2} = (P_{0,1} + P^{\vee 2,2}_{\geq 1,1}) \times P(t_2) + P^{\vee -1,2}_{\geq 1,1} \times P(\overline{t_2}) \overset{\left[P^{\vee 2,2}_{\geq 1,1} = P^{\vee 0,2}_{\geq 1,1}\right]}{=} 0.8$

$P^{\vee 1,2}_{\geq 1,2} = (P_{0,1} + P^{\vee 2,2}_{\geq 1,1}) \times P(t_2) + P^{\vee 0,2}_{\geq 1,1} \times P(\overline{t_2}) \overset{\left[P^{\vee 2,2}_{\geq 1,1} = P^{\vee 0,2}_{\geq 1,1}\right]}{=} 0.92$

(2.3) $j = 3, \vee y_{min} = 1, \vee y_{max} = 2$

$P^{\vee 1,2}_{\geq 1,3} = (P_{0,2} + P^{\vee 2,2}_{\geq 1,2}) \times P(t_3) + P^{\vee 0,2}_{\geq 1,2} \times P(\overline{t_3}) \overset{\left[P^{\vee 2,2}_{\geq 1,2} = P^{\vee 1,2}_{\geq 1,2}\right]}{=} 0.88$

$P^{\vee 2,2}_{\geq 1,3} = (P_{0,2} + P^{\vee 2,2}_{\geq 1,2}) \times P(t_3) + P^{\vee 1,2}_{\geq 1,2} \times P(\overline{t_3}) \overset{\left[P^{\vee 2,2}_{\geq 1,2} = P^{\vee 1,2}_{\geq 1,2}\right]}{=} 0.952$

(2.4) $j = 4, \vee y_{min} = \vee y_{max} = 2$

$P^{\vee 2,2}_{\geq 1,4} = (P_{0,3} + P^{\vee 2,2}_{\geq 1,3}) \times P(t_4) + P^{\vee 1,2}_{\geq 1,3} \times P(\overline{t_4}) = 0.952$

(3) Internal layer: $i = 2,\ j \in [2,5]$

(3.1) $j = 2, \vee y_{min} = \vee y_{max} = 0$

$P^{\vee 0,2}_{\geq 2,2} = P^{\vee 2,2}_{\geq 1,1} \times P(t_2) + P^{\vee -1,2}_{\geq 1,1} \times P(\overline{t_2}) = 0.48$

(3.2) $j = 3, \vee y_{min} = 0, \vee y_{max} = 1$

$P^{\vee 0,2}_{\geq 2,3} = P^{\vee 2,2}_{\geq 1,2} \times P(t_3) + P^{\vee -1,2}_{\geq 2,2} \times P(\overline{t_3}) \overset{\left[P^{\vee 2,2}_{\geq 1,2} = P^{\vee 1,2}_{\geq 1,2}\right]}{=} 0.368$

$P^{\vee 1,2}_{\geq 2,3} = P^{\vee 2,2}_{\geq 1,2} \times P(t_3) + P^{\vee 0,2}_{\geq 2,2} \times P(\overline{t_3}) \overset{\left[P^{\vee 2,2}_{\geq 1,2} = P^{\vee 1,2}_{\geq 1,2}\right]}{=} 0.656$

(3.3) $j = 4, \vee y_{min} = 1, \vee y_{max} = 2$

$P^{\vee 1,2}_{\geq 2,4} = P^{\vee 2,2}_{\geq 1,3} \times P(t_4) + P^{\vee 0,2}_{\geq 2,3} \times P(\overline{t_4}) = 0.7184$

$P^{\vee 2,2}_{\geq 2,4} = P^{\vee 2,2}_{\geq 1,3} \times P(t_4) + P^{\vee 1,2}_{\geq 2,3} \times P(\overline{t_4}) = 0.8336$

(3.4) $j = 5, \vee y_{min} = \vee y_{max} = 2$

$P^{\vee 2,2}_{\geq 2,5} = P^{\vee 2,2}_{\geq 1,4} \times P(t_5) + P^{\vee 1,2}_{\geq 2,4} \times P(\overline{t_5}) = 0.9052$

(4) Top layer: $i = min_t = 3,\ j \in [3,6]$

$P^{2}_{\geq 3,3} = P^{0}_{\geq 3,3} = P^{\vee 2,2}_{\geq 2,2} \times P(t_3) + P^{2}_{\geq 3,2} \times P(\overline{t_3}) \overset{\left[P^{\vee 2,2}_{\geq 2,2} = P^{\vee 0,2}_{\geq 2,2}\right]}{=} 0.192$

$P^{2}_{\geq 3,4} = P^{1}_{\geq 3,4} = P^{\vee 2,2}_{\geq 2,3} \times P(t_4) + P^{2}_{\geq 3,3} \times P(\overline{t_4}) \overset{\left[P^{\vee 2,2}_{\geq 2,3} = P^{\vee 1,2}_{\geq 2,3}, P^{2}_{\geq 3,3} = P^{0}_{\geq 3,3}\right]}{=} 0.4704$

$P^{2}_{\geq 3,5} = P^{\vee 2,2}_{\geq 2,4} \times P(t_5) + P^{2}_{\geq 3,4} \times P(\overline{t_5}) \overset{\left[P^{2}_{\geq 3,4} = P^{1}_{\geq 3,4}\right]}{=} 0.761$

$P^{2}_{\geq 3,6} = P^{\vee 2,2}_{\geq 2,5} \times P(t_6) + P^{2}_{\geq 3,5} \times P(\overline{t_6}) = 0.8187$

REFERENCES

[1] R. Agrawal and R. Srikant, "Mining sequential patterns," in *ICDE*. IEEE, 1995, pp. 3–14.

[2] R. Srikant and R. Agrawal, "Mining sequential patterns: Generalizations and performance improvements," in *EDBT*. Springer, 1996, pp. 1–17.

[3] M. Zaki, "Spade: An efficient algorithm for mining frequent sequences," *Machine Learning*, pp. 31–60, 2001.

[4] J. Ayres, J. Flannick, J. Gehrke, T. Yiu *et al.*, "Sequential pattern mining using a bitmap representation," in *Proc. of KDD*, 2002, pp. 429–435.

[5] J. Pei, J. Han, B. Mortazavi-Asl, J. Wang, H. Pinto, Q. Chen, U. Dayal, and M. Hsu, "Mining sequential patterns by pattern-growth: The prefixspan approach," *ICDE*, 2004.

[6] S. Hwang, Y. Liu, J. Chiu, and E. Lim, "Mining mobile group patterns: A trajectory-based approach," in *PAKDD*. Springer, 2005, pp. 145–146.

[7] Y. Liu, L. Chen, J. Pei, Q. Chen, and Y. Zhao, "Mining frequent trajectory patterns for activity monitoring using radio frequency tag arrays," in *Pervasive Computing*. IEEE, 2007.

[8] F. Giannotti, M. Nanni, F. Pinelli, and D. Pedreschi, "Trajectory pattern mining," in *SIGKDD*. ACM, 2007, pp. 330–339.

[9] C. Jensen, D. Lin, and B. Ooi, "Continuous clustering of moving objects," in *IEEE TKDE*, 2007, pp. 1161–1174.

[10] H. Jeung, M. Yiu, X. Zhou, C. Jensen, and H. Shen, "Discovery of convoys in trajectory databases," in *PVLDB Endowment*, 2008, pp. 1068–1080.

[11] M. Vieira, P. Bakalov, and V. Tsotras, "On-line discovery of flock patterns in spatio-temporal data," in *ACM GIS*, 2009.

[12] R. Agrawal, R. Srikant *et al.*, "Fast algorithms for mining association rules," in *PVLDB*, vol. 1215, 1994, pp. 487–499.

[13] M. Ester, H. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *SIGKDD*, 1996, pp. 226–231.

[14] C. Chui, B. Kao, and E. Hung, "Mining frequent itemsets from uncertain data," in *PAKDD*. Springer, 2007.

[15] M. Hua, J. Pei, W. Zhang, and X. Lin, "Efficiently answering probabilistic threshold top-k queries on uncertain data," in *ICDE*. IEEE, 2008, pp. 1403–1405.

[16] M. Chau, R. Cheng, B. Kao, and J. Ng, "Uncertain data mining: An example in clustering location data," in *PAKDD*. Springer, 2006, pp. 199–204.

[17] C. Aggarwal, Y. Li, J. Wang, and J. Wang, "Frequent pattern mining with uncertain data," in *SIGKDD*. ACM, 2009.

[18] C. Chui and B. Kao, "A decremental approach for mining frequent itemsets from uncertain data," *Advances in Knowledge Discovery and Data Mining*, pp. 64–75, 2008.

[19] Q. Zhang, F. Li, and K. Yi, "Finding frequent items in probabilistic data," in *SIGMOD*. ACM, 2008, pp. 819–832.

[20] T. Bernecker, H. Kriegel, M. Renz, F. Verhein, and A. Zuefle, "Probabilistic frequent itemset mining in uncertain databases," in *SIGKDD*. ACM, 2009, pp. 119–128.

[21] L. Sun, R. Cheng, D. Cheung, and J. Cheng, "Mining uncertain data with probabilistic guarantees," in *SIGKDD*. ACM, 2010, pp. 273–282.

[22] J. Bezdek, R. Ehrlich, and W. Full, "Fcm: The fuzzy c-means clustering algorithm," *Computers & Geosciences*, vol. 10, no. 2, pp. 191–203, 1984.

[23] C. Hwang and F. C.-H. Rhee, "Uncertain fuzzy clustering: interval type-2 fuzzy approach to c-means," *Fuzzy Systems*, vol. 15, no. 1, pp. 107–120, 2007.

[24] H. Mannila, H. Toivonen, and A. Inkeri Verkamo, "Discovery of frequent episodes in event sequences," *DMKD*, vol. 1, no. 3, pp. 259–289, 1997.

[25] S. Laxman, P. Sastry, and K. Unnikrishnan, "A fast algorithm for finding frequent episodes in event streams," in *SIGKDD*. ACM, 2007, pp. 410–419.

[26] T. Calders, C. Garboni, and B. Goethals, "Efficient pattern mining of uncertain data with sampling," in *PAKDD*. Springer, 2010, pp. 480–487.

[27] Y. Tong, L. Chen, Y. Cheng, and P. Yu, "Mining frequent itemsets over uncertain databases," *PVLDB*, vol. 5, no. 11, pp. 1650–1661, 2012.

[28] M. Muzammal and R. Raman, "Mining sequential patterns from probabilistic databases," in *PAKDD*. Springer, 2011.

[29] Z. Zhao, D. Yan, and W. Ng, "Mining probabilistically frequent sequential patterns in uncertain databases," in *EDBT*. ACM, 2012, pp. 74–85.

[30] K. Yi, F. Li, G. Kollios, and D. Srivastava, "Efficient processing of top-k queries in uncertain databases with x-relations," in *TKDE*. IEEE, 2008, pp. 1669–1682.