On Probabilistic Models for Uncertain Sequential Pattern Mining

Muhammad Muzammal and Rajeev Raman

Department of Computer Science, University of Leicester, UK.
{mm386,r.raman}@mcs.le.ac.uk

Abstract. We study uncertainty models in *sequential pattern mining*. We consider situations where there is uncertainty either about a *source* or an *event*. We show that both these types of uncertainties could be modelled using probabilistic databases, and give possible-worlds semantics for both. We then describe "interestingness" criteria based on two notions of frequentness (previously studied for frequent itemset mining) namely *expected support* and *probabilistic frequentness*. We study the interestingness criteria from a complexity-theoretic perspective, and show that in case of source-level uncertainty, evaluating probabilistic frequentness is #P-complete, and thus no polynomial time algorithms are likely to exist, but evaluate the interestingness predicate in polynomial time in the remaining cases.

Keywords: Mining Uncertain Data, Sequential Pattern Mining, Probabilistic Databases, Novel Algorithms for Mining, Theoretical Foundations of Data Mining.

1 Introduction

The problem of *sequential pattern mining (SPM)*, or finding frequent sequences of events in data with a temporal component, was originally proposed by Agrawal and Srikant for analyzing retail data [16], [3]. Since then, it has gained a great deal of attention [23], [15], [4] and has found applications in other domains, including discovering web access patterns and motif extraction [9], [20].The classical SPM problem assumes that the data is *deterministic*, or entirely determined. However, it is recognized that data obtained from a wide range of data sources is inherently uncertain [1], [17]. An approach to modelling uncertainty that has seen increased interest recently is that of *probabilistic* databases [17], [1]; even more recently data mining and ranking problems have been studied in the context of probabilistic databases, including top-k [24], [7] and frequent itemset mining (FIM) [2], [5], [6]. This paper considers the SPM problem in probabilistic databases.

Although SPM is superficially similar to FIM, we observe that generalizing SPM to probabilistic databases gives rise to some subtle issues. As we show, different ways of formalizing the problem can lead to very different outcomes in terms of computational complexity. We now outline the main issues, for which we

give a high-level overview of SPM (a formal definition is in Section 2). In SPM, *events* (retail transactions, observations of objects/persons) are associated with *sources* (customers, sensors/cameras). Events have a time-stamp, so the event database is easily re-organized into a collection of *source sequences*, each of which is a sequence of events (ordered by time stamp) associated with a given source. The SPM problem is to find sequential patterns – patterns of events that have a temporal order – that occur in a significant number of source sequences. Unlike FIM, uncertainty in SPM can occur in three different aspects: the source, the event and the time may all be uncertain. Uncertainty in time seems not well-suited to the probabilistic database approach, and we focus on uncertainty in (any one of) the source and the event.

Source-Level Uncertainty (SLU). In the following situations, an event is recorded deterministically, but the source is not readily identifiable:

(a) a customer (source) purchases some items (event) from a store, and provides identity information (loyalty programme, filling a form etc.). However, multiple matches may emerge in the customer database: the customer's details may be incomplete or incorrect, or the customer database itself may be probabilistic as a result of "deduplication" or cleaning [11].
(b) a vehicle/person is identified by a sensor/camera using methods such as biometrics or automatic number plate recognition (ANPR), which are inherently noisy. To mine patterns such as "10% of cars pass camera X, then camera Y and later camera Z", we consider each car as a source and each sighting as an event.

In such scenarios, it is certain that an event occurred (e.g. a customer bought some items, a vehicle/person entered an area) but the source associated with that event is uncertain. The software would typically assign confidence values to the various alternative matches. Whilst large databases of type (a) are readily available in commercial organizations, large databases of type (b) are now becoming more common, a notable example being the UK's police ANPR database [21]. We model the above scenarios by assuming that each event is associated with a probability distribution over possible sources that could have resulted in the event. This formulation thus shows *attribute-level* uncertainty [17], as the "source" attribute of each event is uncertain.

Event-Level Uncertainty (ELU). In some scenarios, the source of the data is known, but the events are uncertain. Such scenarios could be modelled as a sequence of events output by a source, but with each event only having a certain probability of having truly occurred (i.e. *tuple-level* uncertainty [17]), or as above, using attribute-level uncertainty. For example:

(a) Employees movements are tracked in a building using RFID sensors [13]. The stream of tags read by various sensors is stored in a relation `SIGHTING(t, tID, aID)`, which denotes that the RFID tag `tID` was detected by antenna `aID` at time `t`. Since an RFID antenna has only a certain probability of reading a tag within its range, the PEEX system [13] processes the

`SIGHTING` relation to output an *uncertain* higher-level event relation such as `MEETING(time, person1, person2, room, prob)`. An example tuple in `MEETING` could be (103, 'Alice', 'Bob', 435, 0.4), which means that at time 103, PEEX believes that Alice and Bob are having a meeting (event) with probability 0.4 in room 435 (source) [13]; since antennae are at fixed locations, the source is certain but the event is uncertain.[1]

(b) A logged-in user (source) enters terms into a search engine (events), which are disambiguated in many different ways e.g. a search term "Tiger" could be disambiguated as $\langle(\text{Animal}, 0.5), (\text{Sports Personality}, 0.3), (\text{Airplane}, 0.1), \ldots\rangle$.

Our Contributions. The problem of SPM in probabilistic databases was introduced in [14]. Our main contributions are as follows.

(1) We formalize the notion of event-level uncertainty, provide a possible-worlds semantics, and contrast it with source-level uncertainty. From the possible-worlds semantics, we obtain definitions for frequent sequences under two measures: *expected support* and *probabilistic frequentness*. These measures were used earlier for FIM in probabilistic data [5], [6]. For SPM [14] addressed source-level uncertainty with the expected support; the other three combinations are new.

(2) We discuss the computational complexity of the fundamental question:

Given a sequence s and a probabilistic database, is s frequent?

In the framework described in [10], which includes not only SPM but also FIM and a host of other database optimization and machine learning problems, the above question is the *interestingness* or *quality* predicate. As noted in [10], given such a predicate as a "black box", one can embed it into a variety of candidate generate-and-test frameworks for finding not only frequent sequential patterns, but also maximal frequent sequential patterns. Many popular classical SPM algorithms such as GSP [16], SPADE [23] or SPAM [4] all fit into this framework, and algorithms such as PrefixSpan [15], which do not explicitly generate candidates, also implicitly evaluate the interestingness predicate.

Whilst the interestingness predicate can often be readily evaluated in polynomial time in the deterministic setting, this is not so clear in the probabilistic setting, as a naive approach would involve enumerating exponentially many possible worlds. We show a significant difference from a complexity-theoretic viewpoint between the four variants of the problem (two models and two measures); whilst for three variants, it takes polynomial time to check if a sequence is frequent, for the fourth, this task is #P-complete, and thus no polynomial-time algorithms are likely to exist.

Related Work. Classical SPM has been studied extensively [16], [23], [15], [4]. Modelling uncertain data as *probabilistic* databases [17], [1] has led to several ranking/mining problems being studied in this context. The top-k problem (a ranking problem) was intensively studied recently (see [12], [24], [7] and references therein). In particular [7] highlights subtle issues in the semantics of

[1] Some formulations of this and similar problems may even exhibit SLU.

the term "top-k" when applied to probabilistic databases. FIM in probabilistic databases was studied under the *expected* support measure in [2], [6]; and under the *probabilistic frequentness* measure in [5]. A dichotomy result, showing that certain queries on probabilistic databases are either #P-complete or in PTIME was given in [8], but this applies to "tuple-level independent" probabilistic structures, which do not model SPM (upper bounds) or source-level uncertainty for SPM (for hardness). To the best of our knowledge, apart from [14], the SPM problem in probabilistic databases has not been studied. However, uncertainty in the time-stamp attribute was considered in [18] – we do not consider time to be uncertain. Also [22] studies SPM in "noisy" sequences, but the model proposed there is very different to ours and does not fit in the probabilistic database framework.

2 Problem Statement

2.1 Models of Uncertainty

In this section, we give formal definitions of the source-level and event-level uncertain models. We begin by formally defining the classical SPM problem, then the event-level uncertain model. When defining the event-level uncertain model, for simplicity we consider only the tuple-level uncertain case (example in Table 1). We then recap the source-level uncertain case from [14].

Deterministic SPM. We begin by recapitulating the definition of standard SPM. Let $\mathcal{I} = \{i_1, i_2, \ldots, i_q\}$ be a set of *items* and $\mathcal{S} = \{\sigma_1, \ldots, \sigma_m\}$ be a set of *sources*. An *event* $e \subseteq \mathcal{I}$ is a collection of items. A *database* $D = \langle r_1, r_2, \ldots, r_n \rangle$ is an ordered list of *records* such that each $r_i \in D$ is of the form (eid_i, e_i, σ_i), where eid_i is an event-id, e_i is an event and σ_i is a source. A *sequence* $s = \langle s_1, s_2, \ldots, s_a \rangle$ is an ordered list of events. Let $s = \langle s_1, s_2, \ldots, s_q \rangle$ and $t = \langle t_1, t_2, \ldots, t_r \rangle$ be two sequences. We say that s is a *subsequence* of t, denoted $s \preceq t$, if there exist integers $1 \leq i_1 < i_2 < \cdots < i_q \leq r$ such that $s_k \subseteq t_{i_j}$, for $k = 1, \ldots, q$. The *source sequence* corresponding to a source i, denoted by D_i, is just the multiset $\{e | (eid, e, i) \in D\}$, ordered by *eid*. For a sequence s and source i, let $X_i(s, D)$ be an indicator variable, whose value is 1 if $s \preceq D_i$, and 0 otherwise. The objective is to find all sequences s whose *support* ($Supp$) is at least some user-defined threshold $\theta, 1 \leq \theta \leq m$, where:

$$Supp(s, D) = \sum_{i=1}^{m} X_i(s, D). \tag{1}$$

Event-Level Uncertainty. A *probabilistic database* D^p is a collection of *p-sequences* $D_1^p, \ldots, D_m^p$, where D_i^p is the p-sequence of source $i \in \mathcal{S}$. A p-sequence is the natural analogue of the source sequence in classical SPM, and each p-sequence is of the form $\langle (e_1, c_1) \ldots (e_k, c_k) \rangle$, where the e_j's are events (ordered by eid) and c_j is the confidence that e_j actually occurred. In examples, we write a p-sequence $\langle (\{a, d\}, 0.4), (\{a, b\}, 0.2) \rangle$ as $(a, d : 0.4)(a, b : 0.2)$. An example of an event-level uncertain database is in Table 1(L). The possible worlds semantics

of D^p is as follows. For each event e_j in a p-sequence D_i^p there are two kinds of worlds; one in which e_j occurs and the other where it does not. Let *occur* $= \{x_1, \ldots, x_l\}$, where $1 \leq x_1 < \ldots < x_l \leq k$, be the indices of events that occur in D_i^*. Then $D_i^* = \langle e_{x_1}, \ldots, e_{x_l} \rangle$, and $\Pr(D_i^*) = \prod_{j \in occur} c_j * \prod_{j \notin occur} (1 - c_j)$. In other words, we assume that the events in a p-sequence are stochastically independent. For example, all possible worlds of D_Y^p are shown in Table 1(R) along with detailed probabilities. The set of possible worlds of D_i^p, denoted by $PW(D_i^p)$, is obtained by taking all possible 2^l alternatives for *occur*, and we say $PW(D^p) = PW(D_1^p) \times \ldots \times PW(D_m^p)$. The full set of possible worlds for each source is shown in Table 2(L). For any $D^* \in PW(D^p)$ such that $D^* = (D_1^*, \ldots, D_m^*)$, the probability of D^* is given by $\Pr[D^*] = \prod_{i=1}^m \Pr(D_i^*)$, i.e. we assume that the p-sequences of all sources are mutually independent. An example possible world D^* is shown in Table 2(R), which is obtained by taking one possible world each from the worlds of every p-sequences in Table 2(L). The probability of D^* is the product of probabilities of all the worlds in it, $\Pr[D^*] = 0.29 \times 0.32 \times 0.35 = 0.03$.

Table 1. An event-level uncertain database (L), and all possible worlds of D_Y^p along with their probabilities (R).

	p-sequence
D_X^p	$(a, d : 0.6)(a, b : 0.3)(b, c : 0.7)$
D_Y^p	$(a, d : 0.4)(a, b : 0.2)$
D_Z^p	$(a : 1.0)(a, b : 0.5)(b, c : 0.3)$

$\langle\rangle$	$(1 - 0.4) \times (1 - 0.2) = 0.48$
$\{(a, d)\}$	$(0.4) \times (1 - 0.2) = 0.32$
$\{(a, b)\}$	$(1 - 0.4) \times (0.2) = 0.12$
$\{(a, d)(a, b)\}$	$(0.4) \times (0.2) = 0.08$

Table 2. The set of possible worlds for every p-sequence in Table 1 (L). In all, there are $8 \times 4 \times 4 = 128$ possible worlds of D^p, one such possible world is shown (R).

$PW(D_X^p)$	$\{\langle\rangle = 0.084\}; \{(a, d) = 0.126\}; \{(a, b) = 0.036\}; \{(b, c) = 0.196\}; \{(a, d)(a, b) = 0.054\}; \{(a, d)(b, c) = 0.294\}; \{(a, b)(b, c) = 0.084\}; \{(a, d)(a, b)(b, c) = 0.126\}$
$PW(D_Y^p)$	$\{\langle\rangle = 0.48\}; \{(a, d) = 0.32\}; \{(a, b) = 0.12\}; \{(a, d)(a, b) = 0.08\}$
$PW(D_Z^p)$	$\{(a) = 0.35\}; \{(a)(a, b) = 0.35\}; \{(a)(b, c) = 0.15\}; \{(a)(a, b)(b, c) = 0.15\}$

D_X^*	$\{(a, d)$ (b,c)$\}$	0.294
D_Y^*	$\{(a, d)\}$	0.32
D_Z^*	$\{(a)$ $(a, b)\}$	0.35

Source-Level Uncertainty. A *probabilistic database* D^p is an ordered list of records $\langle r_1, \ldots, r_n \rangle$ of the form (eid, e, W) where eid is an event-id, e is an event and W is a probability distribution over $\mathcal{S}$. The distribution W contains pairs of the form (σ, c), where $\sigma \in \mathcal{S}$ and $0 < c \leq 1$ is the confidence that the event e is associated with source σ; we assume $\sum_{(\sigma,c) \in W} c = 1$. An example of a source-level uncertain database is in Table 3(L). A *possible world* D^* of D^p is generated by taking each event e_i in turn, and assigning it to one of the possible sources $\sigma_i \in W_i$, where $\sigma_i \in \mathcal{S}$. Thus every record $r_i = (eid_i, e_i, W_i) \in D^p$ takes the form $r_i' = (eid_i, e_i, \sigma_i)$, for some $\sigma_i \in \mathcal{S}$ in D^*. By enumerating all such possible combinations, we get the complete set of possible worlds. Assuming that the distributions associated with each record r_i in D^p are stochastically independent, the probability of a possible world D^* is $\Pr[D^*] = \prod_{i=1}^n \Pr_{W_i}[\sigma_i]$. For example, a possible world D^* for the database of Table 3 can be generated by assigning events e_1, e_3 and e_4 to X with probabilities $0.6, 0.3$ and 0.7 respectively, and e_2 to Z with probability 1.0, and $\Pr[D^*] = 0.6 \times 1.0 \times 0.3 \times 0.7 = 0.126$. The complete set of possible worlds for database of Table 3 is in Table 4.

Of course, a source-level uncertain database can be transformed into a collection of p-sequences, and it will be useful to do so. Specifically, for $i = 1, \ldots, m$, let D_i^p be a list of those events in D^p that have non-zero confidence of being associated with source i, ordered by *eid*, together with the associated confidence. However, the resulting p-sequences are *not* independent; as shown in Table 3. Thus, one may view a source-level uncertain database as a collection of p-sequences with dependencies in the form of x-tuples [7].

Table 3. A source-level uncertain database (L) transformed to p-sequences (R). Note that the p-sequence representation is the same as the event-level uncertain database of Table 1. However, events like e_1 (marked with † on (R)) can only be associated with one of the sources X and Y in any possible world.

eid	event	W
e_1	(a, d)	$(X : 0.6)(Y : 0.4)$
e_2	(a)	$(Z : 1.0)$
e_3	(a, b)	$(X : 0.3)(Y : 0.2)(Z : 0.5)$
e_4	(b, c)	$(X : 0.7)(Z : 0.3)$

	p-sequence
D_X^p	$(a, d : 0.6)^\dagger(a, b : 0.3)(b, c : 0.7)$
D_Y^p	$(a, d : 0.4)^\dagger(a, b : 0.2)$
D_Z^p	$(a : 1.0)(a, b : 0.5)(b, c : 0.3)$

Table 4. All possible worlds for the database of Table 3 along with their probabilities. The right-most column shows the support of the sequence (a)(b) in each possible world.

D^*	X	Y	Z	$\Pr(D^*)$	
D_1^*	$(a, d : 0.6)(a, b : 0.3)(b, c : 0.7)$	$\langle\rangle$	$(a : 1.0)$	0.126	1
D_2^*	$(a, d : 0.6)(a, b : 0.3)$	$\langle\rangle$	$(a : 1.0)(b, c : 0.3)$	0.054	2
D_3^*	$(a, d : 0.6)(b, c : 0.7)$	$(a, b : 0.2)$	$(a : 1.0)$	0.084	1
D_4^*	$(a, d : 0.6)$	$(a, b : 0.2)$	$(a : 1.0)(b, c : 0.3)$	0.036	1
D_5^*	$(a, d : 0.6)(b, c : 0.7)$	$\langle\rangle$	$(a : 1.0)(a, b : 0.5)$	0.210	2
D_6^*	$(a, d : 0.6)$	$\langle\rangle$	$(a : 1.0)(a, b : 0.5)(b, c : 0.3)$	0.090	1
D_7^*	$(a, b : 0.3)(b, c : 0.7)$	$(a, d : 0.4)$	$(a : 1.0)$	0.084	1
D_8^*	$(a, b : 0.3)$	$(a, d : 0.4)$	$(a : 1.0)(b, c : 0.3)$	0.036	1
D_9^*	$(b, c : 0.7)$	$(a, d : 0.4)(a, b : 0.2)$	$(a : 1.0)$	0.056	1
D_{10}^*	$\langle\rangle$	$(a, d : 0.4)(a, b : 0.2)$	$(a : 1.0)(b, c : 0.3)$	0.024	2
D_{11}^*	$(b, c : 0.7)$	$(a, d : 0.4)$	$(a : 1.0)(a, b : 0.5)$	0.140	1
D_{12}^*	$\langle\rangle$	$(a, d : 0.4)$	$(a : 1.0)(a, b : 0.5)(b, c : 0.3)$	0.060	1

2.2 Notions of Frequentness

We now use the possible-worlds semantics to give two definitions of frequentness.

Expected Support. As every possible world D^* is a (deterministic) database, $Supp(s, D^*)$ is defined as in Eq. 1. We then define the *expected support* of a sequence s in D^p as follows:

$$ESupp(s, D^p) = \sum_{D^* \in PW(D^p)} \Pr[D^*] * Supp(s, D^*). \tag{2}$$

We illustrate these concepts by computing the expected support of a sequence $s = (a)(b)$ and the event-level uncertain database of Table 1. Since there are too many possible worlds, we do not use Eq. 2 and compute $ESupp(s)$ as follows. We first compute, for every source, the probability that it supports s. E.g. from Table 2, all worlds in $PW(D_Z^p)$ support s except the first one, so the probability that Z supports s is $(0.35 + 0.15 + 0.15) = 0.65$, and the probability that it does not is 0.35. Similarly, the probabilities that X and Y support s are 0.558 and 0.08. Now for $i = 0, 1, 2, 3$, we use the independence of p-sequences to compute

the probability that exactly i sources support s as shown in Table 5(L), e.g. the probability that s is supported by all three sources is $(0.558 \times 0.08 \times 0.65) = 0.029$. Then we get $ESupp(s) = (0 \times 0.142 + \ldots + 3 \times 0.029) = 1.288$.

To compute $ESupp(s)$ for the source-level uncertain case (Table 3), we directly use Eq. 2 together with Table 4 (and get that $ESupp(s) = (1 \times 0.126 + 2 \times 0.054 + \ldots + 1 \times 0.060) = 1.288$. Note that the event- and source-level uncertain databases of Table 1 and 3, which have the same p-sequence form, have the same expected support, even though the possible worlds are very different.

We now formalize the computational task of finding all frequent sequences in a probabilistic database D^p [14]:

Definition 1. *Given a probabilistic database D^p, determine all sequences s such that $ESupp(s, D^p) \geq \theta$, for some user-specified threshold θ.*

Table 5. The SPD for the probabilistic DBs of Table 1 (left) and Table 3 (right).

No. of sources	0	1	2	3
support probability	0.142	0.456	0.372	0.029

No. of sources	0	1	2	3
support probability	0.0	0.712	0.288	0.0

Probabilistic Frequent Sequences. A criticism of expected support [5] is that the expectation of a random variable is only one of its measures, and it does not provide confidence bounds that the support of a sequence is high. Following [5], we define the notion of *probabilistic frequent sequences*.

Given a probabilistic database D^p and its set of possible worlds $PW(D^p)$, the *support probability* for a sequence s and support value k is denoted by:

$$\Pr_k(s) = \sum_{D^* \in PW(D^p), (Supp(s, D^*) = k)} \Pr(D^*), \tag{3}$$

where $Supp(s, D^*)$ is the support of s in D^*. In other words, $\Pr_k(s)$ is the probability that the support of s is exactly k. Next define the *support probability distribution* (SPD) as the vector $\langle \Pr_0(s), \ldots, \Pr_m(s) \rangle$. The SPDs for the databases of Table 1 and Table 3 are shown in Table 5 and are very different, even though the p-sequences are the same, e.g. for the event-level DB in Table 1, $\Pr_3(s) = 0.029$, but in the source-level DB of Table 3, $\Pr_3(s) = 0$, as no such world exists where all three sources support s (see Table 4). Finally, denote by $\Pr_{\geq \theta}(s) = \sum_{k=\theta}^{m} \Pr_k(s)$ the probability that the support of s is at least θ. We now define:

Definition 2. *Given a probabilistic database D^p and two user-specified thresholds, a* support *threshold θ, $1 \leq \theta \leq m$ and a* confidence *threshold $\tau \in (0, 1]$, find all* probabilistic frequent sequences (PFSes)*, which are sequences s s.t. $\Pr_{\geq \theta}(s) \geq \tau$ (i.e. s is a PFS if it has probability $\geq \tau$ of having support $\geq \theta$).*

Observe that the SPD gives far more detailed information than expected support; from the SPD of a sequence one can easily compute not only the expected support, but also the variance and higher moments.

3 Support Computation

We now discuss the computational complexity of computing frequent sequences based on the definitions in the previous section. As indicated earlier, we focus on the interestingness predicate, which when specialized to our problem and definitions of frequentness, yield the following questions. Given a probabilistic database D^p (either source-level or event-level uncertain):

- For a given sequence s and a threshold θ, is $ESupp(s, D^p) \geq \theta$?
- For a given sequence s and thresholds θ and τ, is $\Pr(Supp(s, D^p) \geq \theta) \geq \tau$?

We assume that the database, whether source-level or event-level uncertain, is given as a list of p-sequences. Denote by m the number of sources (as before), by N_i the number of events in the p-sequence of the i-th source, by $N = \sum_{i=1}^{m} N_i$ the total size of all p-sequences and by k the number of events in s.[2]

3.1 Source Support Probability

The first task is to determine the *source support probability*, namely the probability that a given source σ_i supports a sequence s, or $\Pr(s \preceq D_i^p)$. An important issue is that s may be a subsequence of D_i^p in many different ways. For example, if $s = \langle \underbrace{(a)(a)\ldots(a)}_{k \text{ times}} \rangle$ and $D_i^p = \langle (a:c_1), (a,c_2), \ldots, (a,c_{N_i}) \rangle$, then any subset of k positions from D_i^p could be the (sole) basis for the i-th source to support s. In [14] a dynamic programming recurrence is given that computes this probability in polynomial time:

Theorem 1 ([14]). *Given s and D_i^p, we can calculate $\Pr(s \preceq D_i^p)$ in $O(k \cdot N_i)$ time.*

3.2 Expected Support

For both event-level and source-level uncertainty, it was shown in [14] that:

$$ESupp(s, D^p) = \sum_{i=1}^{m} \Pr(s \preceq D_i^p). \tag{4}$$

This holds even though p-sequences are not independent in the source-level uncertain case, due to the principle of linearity of expectation. To calculate $ESupp(s, D^p)$ using Eq. 4, we can apply Theorem 1 to each source in turn, which takes $O(k \cdot \sum_{i=1}^{m} N_i) = O(kN)$ time, and obtain:

Theorem 2. *Given s and D^p, we can calculate $ESupp(s, D^p)$, and hence evaluate the interestingness predicate, in $O(kN)$ time, for both event-level and source-level uncertainty.*

[2] We assume that an event consists of at most a constant number of items.

3.3 Probabilistic Frequentness

Event-Level Uncertainty. As in [5], we compute the entire support probability distribution (SPD), namely the vector $\langle \Pr_0(s), \ldots, \Pr_m(s) \rangle$, where $\Pr_k(s)$ is the probability that the support of s is exactly k. Given the SPD, we then compute the cumulative probabilities $\Pr_{\geq\theta}(s) = \sum_{k=\theta}^{m} \Pr_k(s)$ in $O(m)$ time and thereby answer the interestingness predicate.

In the case of event-level uncertainty, the p-sequences are independent. This allows the SPD to be calculated as a dynamic programming recurrence as follows. We first compute $\Pr(s \preceq D_i^p)$ for all sources i in $O(kN)$ time as in Thm. 2. Next, we define $\Pr_{i,j}(s)$, for $0 \leq i, j \leq m$, as the probability that exactly i of the first j sources support s. We then use the formula:

$$\Pr_{i,j}(s) = \Pr_{i-1,j-1}(s) \cdot \Pr(s \preceq D_i^p) + \Pr_{i,j-1}(s) \cdot (1 - \Pr(s \preceq D_i^p)), \quad (5)$$

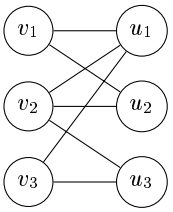
where $\Pr_{0,j}(s) = 1,\ 0 \leq j \leq m$ and $\Pr_{i,j}(s) = 0, \forall\ i > j$, to compute all the values $\Pr_{i,j}$ in $O(m^2)$ time. Since $\Pr_{i,m}(s) = \Pr_i(s)$, we get the full SPD and can use this to determine if s is a PFS; the overall time is $O(kN + m^2)$. In summary:

Theorem 3. *Given s and D^p, if D^p is an event-level uncertain database, we can calculate the support probability distribution, and hence the interestingness predicate in $O(kN + m^2)$ time.*

Remark 1. When computing $\Pr_{i,j}(s)$ using Eq. 5, we consider two cases: either σ_i supports s, and exactly $j-1$ of $\sigma_1, \ldots, \sigma_{i-1}$ support s (the first term) or σ_i does not support s and exactly j of $\sigma_1, \ldots, \sigma_{i-1}$ support s. The correctness of Eq. 5 depends crucially on the fact that we assume independence among p-sequences, so $\Pr(s \preceq D_i^p)$ (resp. $\Pr(s \npreceq D_i^p)$) is unchanged even when conditioned on knowing that exactly $j-1$ (resp. j) of the first $i-1$ sources support s.

Remark 2. Since N/m is the average length of a p-sequence, $N/m \ll m$ and (since k is not very large as well) the m^2 term will often dominate the kN term. This means the interestingness predicate for probabilistic frequentness will often be computationally more expensive than for expected support.

Source-Level Uncertainty. In source-level uncertainty, we cannot use Eq. 5 to efficiently compute the SPD, since the p-sequences are not independent (see Remark 1). Consider the very simple probabilistic database which consists of just the event $\{a\}$, associated with source σ_1 and σ_2 with probabilities 0.5 each. There are only two possible worlds, the first with the event $\{a\}$ associated with σ_1 and nothing with σ_2, and the second the other way around. Clearly, if s is the sequence $\langle\{a\}\rangle$, then $\Pr(s \preceq D_1^p) = \Pr(s \preceq D_2^p) = 0.5$. However, applying Eq. 5 gives that $\Pr_{1,1} = 0.5$ (correct) and $\Pr_{2,2} = 0.25$ (incorrect). The probability that both sources support s is zero, not 0.25 – there is no possible world in which both sources support s. To see what goes wrong, consider the computation of $\Pr_{2,2}(s)$ as $\Pr_{1,1}(s) \cdot 0.5 + \Pr_{2,1}(s) \cdot 0.5 = 0.5 \cdot 0.5 + 0 \cdot 0.5 = 0.25$. Looking at the first term, if σ_1 supports s, then conditioned on this knowledge, the probability that σ_2 supports s is zero. Thus, the correct computation for $\Pr_{2,2}(s)$ would be



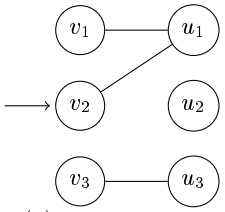
(a)

eid	event	W
e_1	a	$(\sigma_1{:}0.5)(\sigma_2{:}0.5)$
e_2	a	$(\sigma_1{:}0.33)(\sigma_2{:}0.33)(\sigma_3{:}0.33)$
e_3	a	$(\sigma_1{:}0.5)(\sigma_3{:}0.5)$

(b)

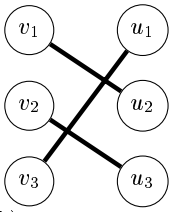
Fig. 1. A sample bipartite graph (a) transformed to a probabilistic database (b).

source	event(s)
σ_1	e_1, e_2
σ_2	$\langle\rangle$
σ_3	e_3



(a)

source	event(s)
σ_1	e_3
σ_2	e_1
σ_3	e_2



(b)

Fig. 2. Two possible worlds in $PW(D^p)$ and their bipartite graph representations. A perfect matching (b), when every vertex in $U \cup V$ is adjacent to a single edge.

as $0.5 \cdot 0 + 0 \cdot 0.5 = 0$. Unfortunately, the difficulty is not with one particular approach but is intrinsic to the problem: we now show that computing even a single entry of the SPD is provably hard. Define the following problem:

Definition 3 (Exact-k Support problem). *The input is a source-level uncertain probabilistic database D^p, a sequence s and a number k, $0 \leq k \leq m$. The output is $\Pr_k(s)$, i.e. the probability that exactly k sources support s.*

Theorem 4. Exact-k Support *is #P-complete.*

Proof. We reduce the problem of computing the number of perfect matchings in a bipartite graph, a known #P-complete problem [19] to Exact-k Support.

Let $G(U, V, E)$ be an undirected bipartite graph, where U and V are disjoint sets of *vertices* and E is the set of *edges* between them, $E \subseteq U \times V$. We assume that $|U| = |V| = n$. A *perfect matching* M is a subset of edges in E such that each vertex in $U \cup V$ is adjacent to exactly a single edge in M. Given a bipartite graph G, the problem of counting how many perfect matchings there are in G is #P-complete [19].

Given a bipartite graph G, to compute the number of matchings in G, we create an instance of Exact-k Support (a probabilistic database D^p, a sequence s and a number k) in polynomial time such that solving the latter instance gives the number of perfect matchings in G. Given $G = (U, V, E)$ where $U = \{u_1, \ldots, u_n\}$ and $V = \{v_1, \ldots, v_n\}$, we create a set of sources $\mathcal{S} = \{\sigma_1, \ldots, \sigma_n\}$ such that $\sigma_i \in \mathcal{S}$ represents $u_i \in U$. The probabilistic database D^p is a set of records $r_i = (e_i, e, W_i)$, where e_i is a event id, e is an event and W_i is a distribution. The record r_i represents v_i in V together with all the edges from v_i to the *neighborhood* of v_i, i.e. the set of vertices in U adjacent

to v_i. In what follows, we denote the neighborhood of v_i as $N(v_i)$. The event contained in *every* record is a set containing just the singleton element $\{a\}$. In the i-th record r_i, the distribution W_i contains only the sources σ_j that represent vertices $u_j \in N(v_i)$. All sources in W_i have the same probability, i.e. $(1/|N(v_i)|)$. An example of such a transformation is shown in Fig. 1. Finally, we choose $k = 0$ and the sequence $s = (a)(a)$, and ask to compute $\Pr_0(s)$, i.e. the probability that no sources support s. This completes creation of the instance of Exact-k Support, and the transformation is clearly polynomial-time. Clearly, every possible world $D^* \in PW(D^p)$ is equally likely, and the probability of a possible world D^* is $\phi = (1/|PW(D^p)|)$, where $|PW(D^p)| = \prod_{i=1}^{n} |N(v_i)|$. For example, there are 12 possible worlds for the database in Fig. 1(b), and the probability of each world is $(1/12)$. In each possible world, each record r_i is associated with some source σ_j, so a possible world can be viewed as a sub-graph of G where every vertex in V has degree exactly one. Two possible worlds and their corresponding graphs are shown in Fig. 2. Those possible worlds where each source is associated with exactly one record corresponds to a perfect matching (Fig. 2(b)); in such possible worlds, the support of the sequence $s = (a)(a)$ will clearly be zero. Thus, we see that $\Pr_0(s) = \phi \cdot (\#$ matchings in $G)$, and once we are given $\Pr_0(s)$, we obtain the number of matchings in G by multiplying by the total number of possible worlds. For example, in database in 1(b), there are only three possible worlds where each source is associated with exactly one event, which are $(\sigma_1 : e_1,\ \sigma_2 : e_2,\ \sigma_3 : e_3)$, $(\sigma_1 : e_2,\ \sigma_2 : e_1,\ \sigma_3 : e_3)$ and $(\sigma_1 : e_3,\ \sigma_2 : e_1,\ \sigma_3 : e_2)$. Hence, $Supp(s, D^*) = 0)$ in three worlds and therefore, $Pr_0(s) = 3 \times (1/12) = 0.25$. We multiply the answer by the number of possible worlds, 12, to get 3, the number of perfect matchings in G.

Hence, we have shown that if the value $\Pr_k(s)$ can be computed for $s = (a)(a)$ and $k = 0$ in D^p, we can also find number of perfect matchings in G, thus reducing the problem of counting perfect matchings in a bipartite graph to Exact-k Support, and showing that Exact-k Support is #P-complete. □

4 Conclusions and Future Work

We studied uncertainty models for sequential pattern mining and defined the notions of frequentness under the *expected suppot* and *probabilistic frequentness* measures. We elaborated on these measures from complexity-theoretic viewpoint, and discussed the computational cost of evaluating these measure for our considered models. Thus we were able to show, that whilst dynamic programming could be used to find frequent sequences efficiently, computing probabilistic frequentness for *source-level uncertainty* is #-P complete. An empirical evaluation and comparison of our considered measures in terms of computational cost and quality of the solution should be an interesting direction to explore.

References

1. Aggarwal, C.C. (ed.): Managing and Mining Uncertain Data. Springer (2009)

2. Aggarwal, C.C., Li, Y., Wang, J., Wang, J.: Frequent pattern mining with uncertain data. In: KDD. pp. 29–38 (2009)
3. Agrawal, R., Srikant, R.: Mining sequential patterns. In: ICDE. pp. 3–14 (1995)
4. Ayres, J., Flannick, J., Gehrke, J., Yiu, T.: Sequential pattern mining using a bitmap representation. In: KDD. pp. 429–435 (2002)
5. Bernecker, T., Kriegel, H.P., Renz, M., Verhein, F., Züfle, A.: Probabilistic frequent itemset mining in uncertain databases. In: KDD. pp. 119–128 (2009)
6. Chui, C.K., Kao, B., Hung, E.: Mining frequent itemsets from uncertain data. In: PAKDD. pp. 47–58 (2007)
7. Cormode, G., Li, F., Yi, K.: Semantics of ranking queries for probabilistic data and expected ranks. In: ICDE. pp. 305–316 (2009)
8. Dalvi, N.N., Suciu, D.: The dichotomy of conjunctive queries on probabilistic structures. In: PODS. pp. 293–302 (2007)
9. El-Ramly, M., Stroulia, E., Sorenson, P.G.: From run-time behavior to usage scenarios: an interaction-pattern mining approach. In: KDD. pp. 315–324 (2002)
10. Gunopulos, D., Khardon, R., Mannila, H., Saluja, S., Toivonen, H., Sharm, R.S.: Discovering all most specific sentences. ACM ToDS 28(2), 140–174 (2003)
11. Hassanzadeh, O., Miller, R.J.: Creating probabilistic databases from duplicated data. The VLDB Journal 18(5), 1141–1166 (2009)
12. Hua, M., Pei, J., Zhang, W., Lin, X.: Ranking queries on uncertain data: a probabilistic threshold approach. In: SIGMOD Conference. pp. 673–686 (2008)
13. Khoussainova, N., Balazinska, M., Suciu, D.: Probabilistic event extraction from rfid data. In: ICDE. pp. 1480–1482 (2008)
14. Muzammal, M., Raman, R.: Mining sequential patterns from probabilistic databases. Tech. Rep. CS-10-002, Dept. of Computer Science, Univ. of Leicester (2010), available from http://www.cs.le.ac.uk/people/mm386/pSPM.pdf
15. Pei, J., Han, J., Mortazavi-Asl, B., Wang, J., Pinto, H., Chen, Q., Dayal, U., Hsu, M.: Mining sequential patterns by pattern-growth: The prefixspan approach. IEEE Trans. Knowl. Data Eng. 16(11), 1424–1440 (2004)
16. Srikant, R., Agrawal, R.: Mining sequential patterns: Generalizations and performance improvements. In: EDBT. pp. 3–17 (1996)
17. Suciu, D., Dalvi, N.N.: Foundations of probabilistic answers to queries. In: SIGMOD Conference. p. 963 (2005)
18. Sun, X., Orlowska, M.E., Li, X.: Introducing uncertainty into pattern discovery in temporal event sequences. In: ICDM. pp. 299–306 (2003)
19. Valiant, L.G.: The complexity of computing the permanent. Theor. Comput. Sci. 8, 189–201 (1979)
20. Wang, K., Xu, Y., Yu, J.X.: Scalable sequential pattern mining for biological sequences. In: CIKM. pp. 178–187 (2004)
21. Wikipedia: http://en.wikipedia.org/wiki/anpr — Wikipedia, the free encyclopedia (2010), \url{http://en.wikipedia.org/wiki/ANPR}, [Online; accessed 30-April-2010]
22. Yang, J., 0010, W.W., Yu, P.S., Han, J.: Mining long sequential patterns in a noisy environment. In: SIGMOD Conference. pp. 406–417 (2002)
23. Zaki, M.J.: Spade: An efficient algorithm for mining frequent sequences. Machine Learning 42(1/2), 31–60 (2001)
24. Zhang, Q., Li, F., Yi, K.: Finding frequent items in probabilistic data. In: SIGMOD Conference. pp. 819–832 (2008)