Open access • Proceedings Article • DOI:10.1145/1557019.1557039

Probabilistic frequent itemset mining in uncertain databases — Source link

Thomas Bernecker, Hans-Peter Kriegel, Matthias Renz, Florian Verhein ...+1 more authors

Institutions: Ludwig Maximilian University of Munich

Published on: 28 Jun 2009 - Knowledge Discovery and Data Mining

Topics: Probabilistic database and Probabilistic logic

Proc. 15th ACM SIGKDD Conf. on Knowledge Discovery and Data Mining (KDD'09), Paris, France, 2009.

Probabilistic Frequent Itemset Mining in Uncertain Databases

Thomas Bernecker, Hans-Peter Kriegel, Matthias Renz, Florian Verhein, Andreas Zuefle
Institut für Informatik, Ludwig-Maximilians-Universität, München, Germany
{bernecker,kriegel,renz,verhein,zuefle}@dbs.ifi.lmu.de

ABSTRACT

Probabilistic frequent itemset mining in uncertain transaction databases semantically and computationally differs from traditional techniques applied to standard "certain" transaction databases. The consideration of existential uncertainty of item(sets), indicating the probability that an item(set) occurs in a transaction, makes traditional techniques inapplicable. In this paper, we introduce new probabilistic formulations of frequent itemsets based on possible world semantics. In this probabilistic context, an itemset X is called frequent if the *probability* that X occurs in at least *minSup* transactions is above a given threshold τ. To the best of our knowledge, this is the first approach addressing this problem under possible worlds semantics. In consideration of the probabilistic formulations, we present a framework which is able to solve the Probabilistic Frequent Itemset Mining (PFIM) problem efficiently. An extensive experimental evaluation investigates the impact of our proposed techniques and shows that our approach is orders of magnitude faster than straight-forward approaches.

Categories and Subject Descriptors

H.2.8 [**Database Applications**]: Data Mining

General Terms

Algorithms, Theory

Keywords

Uncertain Databases, Frequent Itemset Mining, Probabilistic Data, Probabilistic Frequent Itemsets

1. INTRODUCTION

Association rule analysis is one of the most important fields in data mining. It is commonly applied to market-basket databases for analysis of consumer purchasing behaviour. Such databases consist of a set of transactions, each containing the items a customer purchased. The most important and computationally intensive step in the mining process is the extraction of *frequent itemsets* – sets of items that occur in at least *minSup* transactions.

It is generally assumed that the items occurring in a transaction are known for certain. However, this is not always the case. For instance;

- In many applications the data is inherently noisy, such as data collected by sensors or in satellite images.
- In privacy protection applications, artificial noise can be added deliberately [16]. Finding patterns despite this noise is a challenging problem.
- By aggregating transactions by customer, we can mine patterns across customers instead of transactions. This produces estimated purchase probabilities per item per customer rather than certain items per transaction.

In such applications, the information captured in transactions is *uncertain* since the existence of an item is associated with a likelihood measure or existential probability. Given an uncertain transaction database, it is not obvious how to identify whether an item or itemset is frequent because we generally cannot say for certain whether an itemset appears in a transaction. In a traditional (certain) transaction database, we simply perform a database scan and count the transactions that include the itemset. This does not work in an uncertain transaction database.

Dealing with such databases is a difficult but interesting problem. While a naive approach might transform uncertain items into certain ones by thresholding the probabilities, this loses useful information and leads to inaccuracies. Existing approaches in the literature are based on expected support, first introduced in [6]. Chui et. al. [5, 6] take the uncertainty of items into account by computing the expected support of itemsets. Itemsets are considered frequent if the expected support exceeds *minSup*. Effectively, this approach returns an estimate of whether an object is frequent or not with no indication of how good this estimate is. Since uncertain transaction databases yield uncertainty w.r.t. the support of an itemset, the probability distribution of the support and, thus, information about the confidence of the support of an itemset is very important. This information, while present in the database, is lost using the expected support approach.

EXAMPLE 1. *Consider a department store. To maximize sales, customers can be analysed to find sets of items that are*

KDD'09, June 28–July 1, 2009, Paris, France.

Customer	Item	Prob.
A	Game	1.0
A	Music	0.2
B	Video	0.4
B	Music	0.7

ID	Transaction
t_A	(Game, 1.0) ; (Music, 0.2)
t_B	(Video, 0.4) ; (Music, 0.7)

(a) Uncertain Transaction Database

World	TransactionDB	Prob.
1	{Game} ; {}	0.144
2	{Game, Music} ; {}	0.036
3	{Game} ; {Video}	0.096
4	{Game, Music} ; {Video}	0.024
5	{Game} ; {Music}	0.336
6	{Game, Music} ; {Music}	0.084
7	{Game} ; {Video, Music}	0.224
8	{Game, Music} ; {Video, Music}	0.056

(b) Corresponding possible worlds

Figure 1: Example application of an uncertain transaction database.

all purchased by a large group of customers. This information can be used for advertising directed at this group. For example, by providing special offers that include all of these items along with new products, the store can encourage new purchases. Figure 1(a) shows such customer information. Here, customer A purchases games every time he visits the store and music (CDs) 20% of the time. Customer B buys music in 70% of her visits and videos (DVDs) in 40% of them. The supermarket uses a database that represents each customer as a single uncertain transaction*, also shown in Figure 1(a).*

1.1 Uncertain Data Model

The uncertain data model applied in this paper is based on the possible worlds semantic with existential *uncertain items*.

DEFINITION 2. *An* uncertain item *is an item $x \in I$ whose presence in a transaction $t \in T$ is defined by an* existential probability $P(x \in t) \in (0, 1)$. *A* certain item *is an item where $P(x \in t) \in \{0, 1\}$. I is the set of all possible items.*

DEFINITION 3. *An* uncertain transaction *t is a transaction that contains uncertain items. A transaction database T containing uncertain transactions is called an* uncertain transaction database.

An uncertain transaction t is represented in an uncertain transaction database by the items $x \in I$ associated with an existential probability value[1] $P(x \in t) \in (0, 1]$. Example uncertain transaction databases are depicted in Figures 1 and 2. To interpret an uncertain transaction database we apply the *possible world* model. An uncertain transaction database generates *possible worlds*, where each world is defined by a fixed set of (certain) transactions. A possible world is instantiated by generating each transaction $t_i \in T$ according to the occurrence probabilities $P(x \in t_i)$. Consequently, each probability $0 < P(x \in t_i) < 1$ derives two possible worlds *per transaction*: One possible world in which x exists in t_i, and one possible world where x does not exist in t_i. Thus, the number of possible worlds of a database increases exponentially in both the number of transactions and the number of uncertain items contained in it.

Each possible world w is associated with a probability that that world exists, $P(w)$. Figure 1(b) shows all possible worlds derived from Figure 1(a). For example, in world 6 both customers bought music, customer B decided against a new video and customer A bought a new game.

[1]If an item x has an existential probability of zero, it does not appear in the transaction.

ID	Transaction
t_1	(A, 0.8) ; (B, 0.2) ; (D, 0.5) ; (F, 1.0)
t_2	(B, 0.1) ; (C, 0.7) ; (D, 1.0) ; (E, 1.0) ; (G, 0.1)
t_3	(A, 0.5) ; (D, 0.2) ; (F, 0.5) ; (G, 1.0)
t_4	(D, 0.8) ; (E, 0.2) ; (G, 0.9)
t_5	(C, 1.0) ; (D, 0.5) ; (F, 0.8) ; (G, 1.0)
t_6	(A, 1.0) ; (B, 0.2) ; (C, 0.1)

Figure 2: Example of a larger uncertain transaction database.

We assume that uncertain transactions are mutually independent. Thus, the decision by customer A has no influence on customer B. This assumption is reasonable in real world applications. Additionally, independence between items is often assumed in the literature [5, 6]. This can be justified by the assumption that the items are observed independently. In this case, the probability of a world w is given by:

$$P(w) = \prod_{t \in I} (\prod_{x \in t} P(x \in t) * \prod_{x \notin t} (1 - P(x \in t)))$$

For example, the probability of world 5 in Figure 1(b) is $P(Game \in t_A) * (1 - P(Music \in t_A)) * P(Music \in t_B) * (1 - P(Video \in t_B)) = 1.0 * 0.8 * 0.7 * 0.6 = 0.336$.

In the general case, the occurrence of items may be dependent. For example, the decision to purchase a new music video DVD may mean they are unlikely to purchase a music CD by the same artist. Alternatively, some items must be bought together. If these conditional probabilities are known, they can be used in our methods. For example, the probability that both a video and music are purchased by customer B is $P(\{Video, Music\} \in t_B) = P(Video \in t_B) * P(Music \in t_B | Video \in t_B)$.

1.2 Problem Definition

An itemset is a *frequent itemset* if it occurs in at least *minSup* transactions, where *minSup* is a user specified parameter. In uncertain transaction databases however, the support of an itemset is uncertain; it is defined by a discrete probability distribution function (p.d.f). Therefore, each itemset has a *frequentness probability*[2] – the probability that it is frequent. In this paper, we focus on the problem of efficiently calculating this p.d.f. and extracting all *probabilistic frequent itemsets*;

DEFINITION 4. *A* Probabilistic Frequent Itemset *(PFI) is an itemset with a* frequentness probability *of at least τ.*

The parameter τ is the user specified minimum confidence in the frequentness of an itemset.

We are now able to specify the *Probabilistic Frequent Itemset Mining (PFIM) problem* as follows; Given an uncertain transaction database T, a minimum support scalar *minSup* and a frequentness probability threshold τ, find all probabilistic frequent itemsets.

[2]Frequentness is the rarely used word describing the property of being frequent.

1.3 Contributions

We make the following contributions:

- We propose a probabilistic framework for frequent itemset mining in databases containing uncertain transactions, based on the possible worlds model.
- We present a dynamic computation method for computing the probability that an itemset is frequent, as well as the entire probability distribution function of the support of an itemset, in $O(|T|)$ time[3]. Without this technique, it would run in exponential time in the number of transactions. Using our approach, our algorithm has the same time complexity as methods based on the expected support [5, 6, 11]. However, our approach yields much better effectiveness since it provides confidences for frequent itemsets.
- We propose an algorithm to mine all itemsets that are frequent with a probability of at least τ. Furthermore, we propose an additional algorithm that incrementally outputs the uncertain itemsets in the order of their frequentness probability. This ensures that itemsets with the highest probability of being frequent are output first. This has two additional advantages; First, it makes the approach free of the parameter τ. Secondly, it solves the top k itemsets problem in uncertain databases.

The remainder of this paper is organised as follows; Section 2 surveys related work. Section 3 presents our probabilistic support framework. Section 4 shows how to compute the frequentness probability in $O(|T|)$ time. Section 5 presents a probabilistic frequent itemset mining algorithm. Section 6 presents our incremental algorithm. We present our experiments in Section 7 and conclude in Section 8.

2. RELATED WORK

There is a large body of research on Frequent Itemset Mining (FIM) but very little work addresses FIM in uncertain databases [5, 6, 11]. The approach proposed by Chui et. al [6] computes the expected support of itemsets by summing all itemset probabilities in their U-Apriori algorithm. Later, in [5], they additionally proposed a probabilistic filter in order to prune candidates early. In [11], the UF-growth algorithm is proposed. Like U-Apriori, UF-growth computes frequent itemsets by means of the expected support, but it uses the FP-tree [9] approach in order to avoid expensive candidate generation. In contrast to our probabilistic approach, itemsets are considered frequent if the expected support exceeds *minSup*. The main drawback of this estimator is that information about the uncertainty of the expected support is lost; [5, 6, 11] ignore the number of possible worlds in which an itemsets is frequent. [18] proposes exact and sampling-based algorithms to find likely frequent items in streaming probabilistic data. However, they do not consider itemsets with more than one item. Finally, except for [15], existing FIM algorithms assume binary valued items which precludes simple adaptation to uncertain databases. To the best of our knowledge, our approach is the first that is able to find frequent itemsets in an uncertain transaction database in a probabilistic way.

[3]Assuming *minSup* is a constant.

Existing approaches in the field of uncertain data management and mining can be categorized into a number of research directions. Most related to our work are the two categories "*probabilistic databases*" [4, 12, 13, 3] and "*probabilistic query processing*" [7, 10, 17, 14].

The uncertainty model used in our approach is very close to the model used for probabilistic databases. A probabilistic database denotes a database composed of relations with uncertain tuples [7], where each tuple is associated with a probability denoting the likelihood that it exists in the relation. This model, called "*tuple uncertainty*", adopts the possible worlds semantics [3]. A probabilistic database represents a set of possible "certain" database instances (worlds), where a database instance corresponds to a subset of uncertain tuples. Each instance (world) is associated with the probability that the world is "true". The probabilities reflect the probability distribution of all possible database instances. In the general model description [13], the possible worlds are constrained by rules that are defined on the tuples in order to incorporate object (tuple) correlations. The ULDB model proposed in [4], which is used in *Trio*[1], supports uncertain tuples with alternative instances which are called x-tuples. Relations in ULDB are called x-relations containing a set of x-tuples. Each x-tuple corresponds to a set of tuple instances which are assumed to be mutually exclusive, i.e. no more than one instance of an x-tuple can appear in a possible world instance at the same time. Probabilistic top-k query approaches [14, 17, 12] are usually associated with uncertain databases using the tuple uncertainty model. The approach proposed in [17] was the first approach able to solve probabilistic queries efficiently under tuple independency by means of dynamic programming techniques. In our paper, we adopt the dynamic programming technique for the efficient computation of frequent itemsets in a probabilistic way.

3. PROBABILISTIC FREQUENT ITEMSETS

Recall that previous work was based on the expected support [5, 6, 11].

DEFINITION 5. *Given an uncertain transaction database T, the* expected support $E(X)$ *of an itemset X is defined as* $E(X) = \sum_{t \in T} P(X \subseteq t)$.

Considering an itemset frequent if its expected support is above *minSup* has a major drawback. Uncertain transaction databases naturally involve uncertainty concerning the support of an itemset. Considering this is important when evaluating whether an itemset is frequent or not. However, this information is forfeited when using the expected support approach. Let us return to the example shown in Figure 2. The expected support of the itemset $\{D\}$ is $E(\{D\}) = 3.0$. The fact that $\{D\}$ occurs for certain in one transaction, namely in t_2, and that there is at least one possible world where X occurs in five transactions are totally ignored when using the expected support in order to evaluate the frequency of an itemset. Indeed, suppose $minSup = 3$; do we call $\{D\}$ frequent? And if so, how certain can we even be that $\{D\}$ is frequent? By comparison, consider itemset $\{G\}$. This also has an expected support of 3, but its presence or absence in transactions is more certain. It turns out that the probability that $\{D\}$ is frequent is 0.7 and the probability that G is frequent is 0.91. While both have the same expected support, we can be quite confident that $\{G\}$ is frequent, in

Notation	Description
W, w	Set of all possible worlds, Possible world instance $w \in W$
T, t	Uncertain transaction database, transaction $t \in T$
I	Set of all items
X, x	Itemset $X \subseteq I$, item $x \in I$
$S(X, w)$	Support of X in world w
$P_i(X)$	Probability that the support of X is i
$P_{\geq i}(X)$	Probability that the support of X is *at least* i
$P_{i,j}(X)$	Probability that i of the first j transactions contain X
$P_{\geq i,j}(X)$	Probability that *at least* i of the first j transactions contain X

Figure 3: Summary of Notations

contrast to $\{D\}$. An expected support based technique does not differentiate between the two.

The confidence with which an itemset is frequent is very important for interpreting uncertain itemsets. We therefore require concepts that allow us to evaluate the uncertain data in a probabilistic way. In this section, we formally introduce the concept of probabilistic frequent itemsets.

3.1 Probabilistic Support

In uncertain transaction databases, the support of an item or itemset cannot be represented by a unique value, but rather, must be represented by a discrete probability distribution.

DEFINITION 6. *Given an uncertain (transaction) database T and the set W of possible worlds (instantiations) of T, the* support probability $P_i(X)$ *of an itemset X is the probability that X has the support i. Formally,*

$$P_i(X) = \sum_{w_j \in W, (S(X,w_j)=i)} P(w_j)$$

where $S(X, w_j)$ is the support of X in world w_j.

Intuitively, $P_i(X)$ denotes the probability that the support of X is exactly i . The support probabilities associated with an itemset X for different support values form the *support probability distribution* of the support of X.

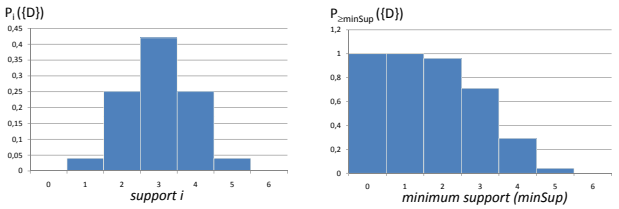
DEFINITION 7. *The probabilistic* support *of an itemset X in an uncertain transaction database T is defined by the support probabilities of X ($P_i(X)$) for all possible support values $i \in \{0, ..., |T|\}$. This probability distribution is called* support probability distribution. *The following statement holds:* $\sum_{0 \leq i \leq |T|} P_i(X) = 1.0$.

Returning to our example of Figure 2, Figure 4(a) shows the support probability distribution of itemset $\{D\}$.

The number of possible worlds $|W|$ that need to be considered for the computation of $P_i(X)$ is extremely large. In fact, we have $O(2^{|T| \cdot |I|})$ possible worlds, where $|I|$ denotes the total number of items. In the following, we show how to compute $P_i(X)$ without materializing all possible worlds.

LEMMA 8. For an uncertain transaction database T with mutually independent transactions and any $0 \leq i \leq |T|$, the support probability $P_i(X)$ can be computed as follows:

$$P_i(X) = \sum_{S \subseteq T, |S|=i} \left(\prod_{t \in S} P(X \subseteq t) \cdot \prod_{t \in T-S} (1 - P(X \subseteq t)) \right) \quad (1)$$



(a) Support probability distribution of $\{D\}$

(b) Frequentness probabilities of $\{D\}$

Figure 4: Probabilistic support of itemset $X = \{D\}$ in the uncertain database of Figure 2.

Note that the transaction subset $S \subseteq T$ contains exactly i transactions.

PROOF. The transaction subset $S \subseteq T$ contains i transactions. The probability of a world w_j where all transactions in S contain X and the remaining $|T-S|$ transactions do not contain X is $P(w_j) = \prod_{t \in S} P(X \subseteq t) \cdot \prod_{t \in T-S} (1 - P(X \subseteq t))$. The sum of the probabilities according to all possible worlds fulfilling the above conditions corresponds to the equation given in Definition 6. □

3.2 Frequentness Probability

Recall that we are interested in the *probability* that an itemset is frequent, i.e. the probability that an itemset occurs in at least *minSup* transactions.

DEFINITION 9. *Let T be an uncertain transaction database and X be an itemset. $P_{\geq i}(X)$ denotes the probability that the support of X is* at least *i, i.e. $P_{\geq i}(X) = \sum_{k=i}^{|T|} P_k(X)$. For a given minimal support* minSup $\in \{0, \ldots, |T|\}$*, the probability $P_{\geq \text{minSup}}(X)$, which we call the* frequentness probability *of X, denotes the probability that the support of X is at least* minSup.

Figure 4(b) shows the frequentness probabilities of $\{D\}$ for all possible *minSup* values in the database of Figure 2. For example, the probability that $\{D\}$ is frequent when $minSup = 3$ is approximately 0.7, while its frequentness probability when $minSup = 4$ is approximately 0.3.

The intuition behind $P_{\geq minSup}(X)$ is to show how confident we are that an itemset is frequent. With this policy, the frequentness of an itemset becomes subjective and the decision about which candidates should be reported to the user depends on the application. Hence, we use the minimum frequentness probability τ as a user defined parameter. Some applications may need a low τ, while in other applications only highly confident results should be reported (high τ).

In the possible worlds model we know that $P_{\geq i}(X) = \sum_{w_j \in W:(S(X,w_j) \geq i)} P(w_j)$. This can be computed according to Equation 1 by

$$P_{\geq i}(X) = \sum_{S \subseteq T, |S| \geq i} \left(\prod_{t \in S} P(X \subseteq t) \cdot \prod_{t \in T-S} (1 - P(X \subseteq t)) \right). \quad (2)$$

Hence, the frequentness probability can be calculated by enumerating all possible worlds satisfying the *minSup* condition through the direct application of Equation 2. This naive

approach is very inefficient however. We can speed this up significantly. First, note that typically $minSup << |T|$ and the number of worlds with support i is at most $\binom{|T|}{i}$. Hence, enumeration of all worlds w in which the support of X is greater than *minSup* is much more expensive than enumerating those where the support is less than *minSup*. Using the following easily verified Lemma, we can compute the frequentness probability exponentially in $minSup << |T|$.

LEMMA 10. $P_{\geq i}(X) = 1 - \sum_{S \subseteq T:|S|<i}(\prod_{t \in S} P(X \subseteq t) \cdot \prod_{t \in T-S}(1 - P(X \subseteq t)))$.

Despite this improvement, the complexity of the above approach, called **Basic** in our experiments, is still exponential w.r.t. the number of transactions. In Section 4 we describe how we can reduce this to linear time.

4. EFFICIENT COMPUTATION OF PROBABILISTIC FREQUENT ITEMSETS

This section presents our dynamic programming approach, which avoids the enumeration of possible worlds in calculating the frequentness probability and the support distribution. We also present probabilistic filter and pruning strategies which further improve the run time of our method.

4.1 Efficient Computation of Probabilistic Support

The key to our approach is to consider it in terms of subproblems. First, we need appropriate definitions;

DEFINITION 11. *The probability that i of j transactions contain itemset X is*

$$P_{i,j}(X) = \sum_{S \subseteq T_j:|S|=i} (\prod_{t \in S} P(X \subseteq t) \cdot \prod_{t \in T_j - S} (1 - P(X \subseteq t)))$$

where $T_j = \{t_1, ..., t_j\} \subseteq T$ is the set of the first j transactions. Similarly, the probability that at least *i of j transactions contain itemset X is*

$$P_{\geq i,j}(X) = \sum_{S \subseteq T_j:|S| \geq i} (\prod_{t \in S} P(X \subseteq t) \cdot \prod_{t \in T_j - S} (1 - P(X \subseteq t)))$$

Note that $P_{\geq i,|T|}(X) = P_{\geq i}(X)$, the probability that at least i transactions in the entire database contain X. The key idea in our approach is to split the problem of computing $P_{\geq i,|T|}(X)$ into smaller problems $P_{\geq i,j}(X)$, $j < |T|$. This can be achieved as follows. Given a set of j transactions $T_j = \{t_1, ..., t_j\} \subseteq T$: If we assume that transaction t_j contains itemset X, then $P_{\geq i,j}(X)$ is equal to the probability that at least $i-1$ transactions of $T_j \backslash \{t_j\}$ contain X. Otherwise, $P_{\geq i,j}(X)$ is equal to the probability that at least i transactions of $T_j \backslash \{t_j\}$ contain X. By splitting the problem in this way we can use the recursion in Lemma 12, which tells us what these probabilities are, to compute $P_{\geq i,j}(X)$ by means of the paradigm of dynamic programming.

LEMMA 12. $P_{\geq i,j}(X) =$

$$P_{\geq i-1,j-1}(X) \cdot P(X \subseteq t_j) + P_{\geq i,j-1}(X) \cdot (1 - P_j(X \subseteq t_j))$$

where

$$P_{\geq 0,j} = 1 \ \forall .0 \leq j \leq |T|, \ P_{\geq i,j} = 0 \ \forall .i > j$$

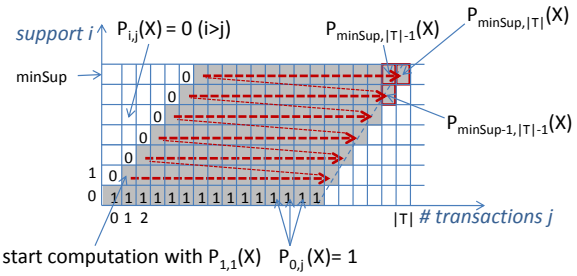


Figure 5: Dynamic Computation Scheme

The above dynamic programming scheme is an adaption of a technique previously used in the context of probabilistic top-k queries by Kollios et. al [17].

PROOF. $P_{\geq i,j}(X) = \sum_{k=i}^{j} P_{k,j}(X) \overset{[Kollios\ et\ al]}{=} \sum_{k=i}^{j} P_{k-1,j-1}(X) \cdot P(X \subseteq t_j) + \sum_{k=i}^{j} P_{k,j-1}(X) \cdot (1 - P(X \subseteq t_j)) \overset{[P_{\geq i,j}=0 \ \forall .i>j]}{=} P(X \subseteq t_j) \cdot \sum_{k=i}^{j-1} P_{k-1,j-1}(X) + \cdot (1 - P(X \subseteq t_j)) \cdot \sum_{k=i}^{j-1} P_{k,j-1}(X) = P(X \subseteq t_j) \cdot P_{\geq i-1,j-1}(X) + (1 - P(X \subseteq t_j)) \cdot P_{\geq i,j-1}(X)$. □

Using this dynamic programming scheme, we can compute the probability that at least *minSup* transactions contain itemset X by calculating the cells depicted in Figure 5. In the matrix, each cell relates to a probability $P_{\geq i,j}$, with i marked on the x-axis, and j marked on the y-axis. Note that according to Lemma 12, in order to compute a $P_{\geq i,j}$, we require the probabilities $P_{\geq i-1,j-1}$ and $P_{\geq i,j-1}$, that is, the cell to the left and the cell to the lower left of $P_{\geq i,j}$. Knowing that $P_{\geq 0,0} = 1$ and $P_{\geq 1,0} = 0$ by definition, we can start by computing $P_{\geq 1,1}$. The probability $P_{\geq 1,j}$ can then be computed by using the previously computed $P_{\geq 1,j-1}$ for all j. $P_{\geq 1,j}$can, in turn, be used to compute $P_{\geq 2,j}$. This iteration continues until i reaches *minSup*, so that finally we obtain $P_{\geq minSup,|T|}$ – the frequentness probability (Definition 9).

Note that in each line (i.e. for each i) of the matrix in Figure 5, j only runs up to $|T| - minSup + i$. Larger values of j are not required for the computation of $P_{minSup,|T|}$.

LEMMA 13. *The computation of the frequentness probability $P_{\geq minSup}$ requires at most $O(|T| * \text{minSup}) = O(|T|)$ time and at most $O(|T|)$ space.*

PROOF. Using the dynamic computation scheme as shown in Figure 5, the number of computations is bounded by the size of the depicted matrix. The matrix contains $|T| * minSup$ cells. Each cell requires an iteration of the dynamic computation (c.f. Corollary 12) which is performed in $O(1)$ time. Note that a matrix is used here for illustration purpose only. The computation of each probability $P_{i,j}(X)$ only requires information stored in the current line and the previous line to access the probabilities $P_{i-1,j-1}(X)$ and $P_{i,j-i}(X)$. Therefore, only these two lines (of length $|T|$) need to be preserved requiring $O(|T|)$ space. Additionally, the probabilities $P(X \subseteq t_j)$ have to be stored, resulting in a total of $O(|T|)$ space. □

Note that we can save computation time if an itemset is certain in some transactions. If a transaction $t_j \in T$ contains itemset X with a probability of zero, i.e. $P(X \subseteq t_j) = 0$,

transaction t_j can be ignored for the dynamic computation because $P_{\geq i,j}(X) = P_{\geq i,j-1}(X)$ holds (Lemma 12). If $|T'|$ is less than *minSup*, then X can be pruned since, by definition, $P_{\geq minSup,T'} = 0$ if $minSup > T'$. The dynamic computation scheme can also omit transactions T_j where the item has a probability of 1, because $P_{\geq i,j}(X) = P_{\geq i-1,j-1}(X)$ due to $P(X \subseteq t_j) = 1$. Thus, if a transaction t_j contains X with a probability of 1, then t_j (i.e. the corresponding column) can be omitted if *minSup* is reduced by one, to compensate the missing transaction. The dynamic programming scheme therefore only has to consider uncertain items. We call this trick *0-1-optimization*.

4.2 Probabilistic Filter Strategies

To further reduce the computational cost, we introduce probabilistic filter strategies. These reduce the number of probability computations in the dynamic algorithm. Our probabilistic filter strategies exploit the following monotonicity criteria;

4.2.1 Monotonicity Criteria

First, if we increase the minimal support i, then the frequentness probability of an itemset decreases.

LEMMA 14. $P_{\geq i,j}(X) \geq P_{\geq i+1,j}(X)$.

PROOF. $P_{\geq i+1,j}(X) \overset{Definition\,9}{=} P_{\geq i,j}(X) - P_{i+1,j}(X) \leq P_{\geq i,j}(X)$ □

Intuitively, this result is obvious since the predicate "the support is at least i" implies "the support is at least $i+1$". The next criterion says that an extension of the uncertain transaction database leads to an increase of the frequentness probability of an itemset.

LEMMA 15. $P_{\geq i,j}(X) \leq P_{\geq i,j+1}(X)$.

PROOF. $P_{\geq i,j+1}(X) \overset{Lemma\,12}{=} P_{\geq i-1,j}(X) \cdot P(X \subseteq t_{j+1}) + P_{\geq i,j}(X) \cdot (1 - P(X \subseteq t_{j+1})) \overset{Lemma\,14}{\geq} P_{\geq i,j}(X) \cdot P(X \subseteq t_{j+1}) + P_{\geq i,j}(X) \cdot (1 - P(X \subseteq t_{j+1})) = P_{\geq i,j}(X)$ □

The intuition behind this lemma is that one more transaction can increase the support of an itemset. Putting these results together;

LEMMA 16. $P_{\geq i,j}(X) \geq P_{\geq i+1,j+1}(X)$.

PROOF. $P_{\geq i+1,j+1}(X) \overset{Corollary\,12}{=} P_{\geq i,j}(X) \cdot P(X \subseteq t_{j+1}) + P_{\geq i+1,j}(X)(1 - P(X \subseteq t_{j+1})) \overset{Lemma\,14}{\leq} P_{\geq i,j}(X) \cdot P(X \subseteq t_{j+1}) + P_{\geq i,j}(X)(1 - P(X \subseteq t_{j+1})) = P_{\geq i,j}$. □

Next, we describe how these monotonicity criteria can be exploited to prune the dynamic computation.

4.2.2 Pruning Criterion

Lemma 16 can be used to quickly identify non-frequent itemsets. Figure 6 shows the dynamic programming scheme for an itemset X. Keep in mind that the goal is to compute $P_{minSup,|T|}(X)$. Lemma 16 states that the probabilities $P_{minSup-k,|T|-k}(X), 1 \leq k \leq minSup$ (highlighted in Figure 6), are conservative bounds of $P_{minSup,|T|}(X)$. Thus, if any of the probabilities $P_{minSup-k,|T|-k}(X)$, $1 \leq k \leq minSup$ is lower than the user specified parameter τ, then X can be pruned.

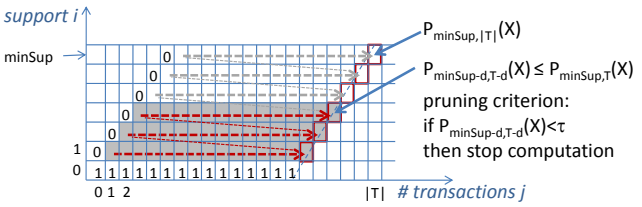


Figure 6: Visualization of the Pruning Criterion

5. PROBABILISTIC FREQUENT ITEMSET MINING (PFIM)

We now have the techniques required to efficiently identify whether a given itemset X is a probabilistic frequent itemset (PFI). In this section, we show how to find all probabilistic frequent itemsets in an uncertain transaction database. Traditional frequent itemset mining is based on support pruning by exploiting the anti-monotonic property of support: $S(X) \leq S(Y)$ where $S(X)$ is the support of X and $Y \subseteq X$. In uncertain transaction databases however, recall that support is defined by a probability distribution and that we mine itemsets according to their frequentness probability. It turns out that the frequentness probability is anti-monotonic:

LEMMA 17. $\forall Y \subseteq X : P_{\geq minSup}(X) \leq P_{\geq minSup}(Y)$. *In other words, all subsets of a probabilistic frequent itemset are also probabilistic frequent itemsets.*

PROOF. $P_{\geq i}(X) = \frac{1}{|W|} \sum_{i=1}^{|W|} P(w_i) \cdot I_{S(X,w_i) \geq minSup}$, since the probability is defined over all possible worlds. Here, I_Z is an indicator variable that is 1 when $z = true$ and 0 otherwise. In other words, $P_{\geq i}(X)$ is the relative number of worlds in which $S(X) \geq minSup$ holds, where each occurrence is weighted by the probability of the world occurring. Since world w_i corresponds to a normal transaction database with no uncertainty, $S(X, w_i) \leq S(Y, w_i) \forall Y \subseteq X$ due to the anti-monotonicity of support. Therefore, $I_{S(X,w_i) \geq minSup} \leq I_{S(Y,W_i) \geq minSup} \forall i \in |W|, \forall Y \subseteq X$ and, thus, $P_{\geq i}(X) \leq P_{\geq i}(Y), \forall Y \subseteq X$. □

We can use the contra-positive of Lemma 17 to prune the search space for probabilistic frequent itemsets. That is, if an itemset Y is not a probabilistic frequent itemset, i.e. $P_{\geq minSup}(Y) < \tau$, then all itemsets $X \supseteq Y$ cannot be probabilistic frequent itemsets either.

Our first algorithm is based on a "marriage" of traditional frequent itemset mining methods and our uncertain itemset identification algorithms. In particular, we propose a probabilistic frequent itemset mining approach based on the Apriori algorithm ([2]). Like Apriori, our method iteratively generates the probabilistic frequent itemsets using a bottom-up strategy. Each iteration is performed in two steps, a join step for generating new candidates and a pruning step for calculating the frequentness probabilities and extracting the probabilistic frequent itemsets from the candidates. The pruned candidates are, in turn, used to generate candidates in the next iteration. Lemma 17 is exploited in the join step to limit the candidates generated and in the pruning step to remove itemsets that need not be expanded.

6. INCREMENTAL PROBABILISTIC FREQUENT ITEMSET MINING (I-PFIM)

Our probabilistic frequent itemset mining approach allows the user to control the confidence of the results using τ. However, since the number of results depends on τ, it may prove difficult for a user to correctly specify this parameter without additional domain knowledge. Therefore, this Section shows how to efficiently solve the following problems, which do not require the specification of τ;

- *Top-k probabilistic frequent itemsets query:* return the k itemsets that have the highest frequentness probability, where k is specified by the user.
- *Incremental ranking queries:* successively return the itemsets with the highest frequentness probability one at a time.

6.1 Incremental Probabilistic Frequent Itemset Mining Algorithm

In our incremental algorithm (Algorithm 1), we keep an *Active Itemsets Queue (AIQ)* that is initialized with all one-item sets. The *AIQ* is sorted by frequentness probability in descending order. Without loss of generality, itemsets are represented in lexiographical order to avoid generating them more than once. In each iteration of the algorithm, i.e. each call of the *getNext()*-function, the first itemset X in the queue is removed. X is the next most probable frequent itemset because all other itemsets in the *AIQ* have a lower frequentness probability due to the order on *AIQ*, and all of X's supersets (which have not yet been generated) cannot have a higher frequentness probability due to Lemma 17. Before X is returned to the user, it is refined in a candidate generation step. In this step, we create all supersets of X obtained by adding single items x to the end of X, in such a way that the lexiographical order of $X \cup x$ is maintained. These are then added to the *AIQ* after their respective frequentness probabilities are computed (Section 4). The user can continue calling the *getNext()*-function until he has all required results. Note that during each call of the *getNext()*-function, the size of the *AIQ* increases by at most $|I|$. The maximum size of the *AIQ* is $2^{|I|}$, which is no worse than the space required to sort the output of a non-incremental algorithm.

6.2 Top-k Probabilistic Frequent Itemsets Query

In many applications however, relatively few top probabilistic frequent itemsets are required. For instance, the store in Example 1 may want to know the top $k = 100$. Top-k highest frequentness probability queries can be efficiently computed by using Algorithm 1 and constraining the length of the *AIQ* to $k - m$, where m is the number of highest frequentness probability items already returned. Any itemsets that "fall off" the end can safely be ignored. The rational behind this approach is that for an itemset X at position p in the *AIQ*, $p - 1$ itemsets with a higher frequentness than X exist in the *AIQ* by construction. Additionally, any of the m itemsets that have already been returned must have a higher frequentness probability. Consequently, our top-k algorithm contrains the size of the initial *AIQ* to k and reduces its size by one each time a result is reported. The algorithm terminates once the size of the *AIQ* reaches zero.

Algorithm 1 Incremental Algorithm

```
//initialise
AIQ = new PriorityQueue
FOR EACH x ∈ I
  AIQ.add([x,P_{≥minSup}(x)])
//return the next probabilistic frequent itemset
getNext() RETURNS X
  X = AIQ.removeFirst()
  FOR EACH (x ∈ I \ X : x = lastInLexOrder(X ∪ x))
    AIQ.add([X ∪ x,P_{≥minSup}(X ∪ x)])
```

7. EXPERIMENTAL EVALUATION

In this Section we present efficiency and efficacy experiments. First, we give efficiency results obtained utilizing the different methods of computing the probabilistic support (cf. Sections 3 and 4). Then, we discuss the performance and utility of the proposed probabilistic frequent itemset mining algorithms (cf. Sections 5 and 6). In all experiments, the runtime was measured in milliseconds (ms).

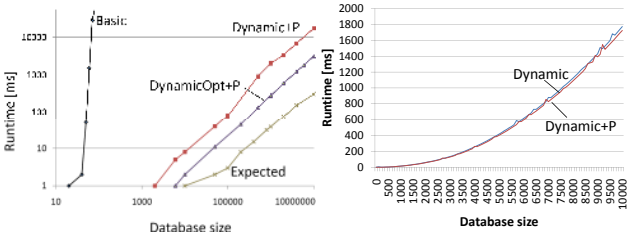
7.1 Evaluation of the Frequentness Probability Calculations

We evaluated our frequentness probability calculation methods on several artificial datasets with varying database sizes $|T|$ and densities. The *density* of an item denotes the expected number of transactions in which an item may be present (i.e. where its existence probability is in $(0, 1]$). The probabilities themselves were drawn from a uniform distribution. Note that the density is directly related to the degree of uncertainty. If not stated otherwise, we used a database consisting of $10,000$ to $10,000,000$ uncertain transactions and a density of 0.5. The frequentness probability threshold τ of was set to 0.9.

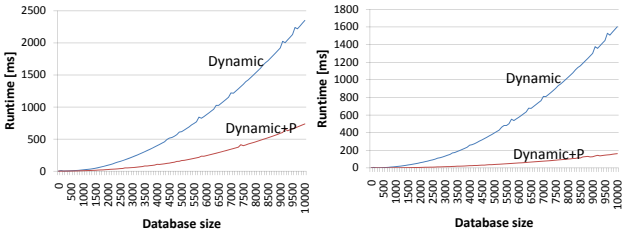
We use the following notations for our frequentness probability algorithms: **Basic**: basic probability computation (Section 3.2), **Dynamic**: dynamic probability computation (Section 4.1), **Dynamic+P**: dynamic probability computation with pruning (Section 4.2), **DynamicOpt**: dynamic probability computation utilizing 0-1-optimization (Section 4.1) and **DynamicOpt+P**: 0-1-optimized dynamic probability computation method with pruning.

7.1.1 Scalability

Figure 7 shows the scalability of the probability calculation approaches when we vary the number of transactions, $|T|$. The runtime of the **Basic** approach increases exponentially in *minSup* as explained in Section 3.2, and is therefore not applicable for a $|T| > 50$ as can be seen in Figure 7(a). Our approaches **Dynamic+P** and **DynamicOpt+P** scale linearly as expected when using a constant *minSup* value. The 0-1-optimization has an impact on the runtime whenever there is some certainty in the database. The performance gain of our pruning strategies depends on the used *minSup* value. In Figures 7(b), 7(c) and 7(d) the scalability of **Dynamic** and **Dynamic+P** is shown for different *minSup* values expressed as percentages of $|T|$. It is notable that the time complexity of $O(|T| * minSup)$ becomes $O(|T|^2)$ if *minSup* is chosen relative to the database size. Also, it can be observed that the higher *minSup*, the higher the difference between **Dynamic** and **Dynamic+P**; a higher *minSup* causes the frequentness probability to fall overall, thus allowing earlier pruning.

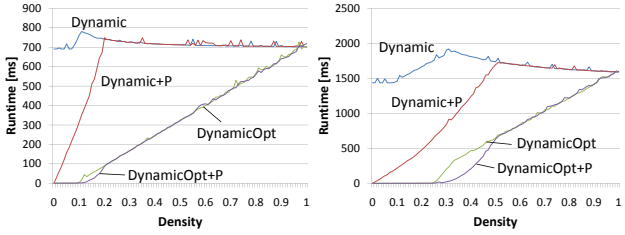


(a) $minSup = 10$ (b) $minSup = 25\%$

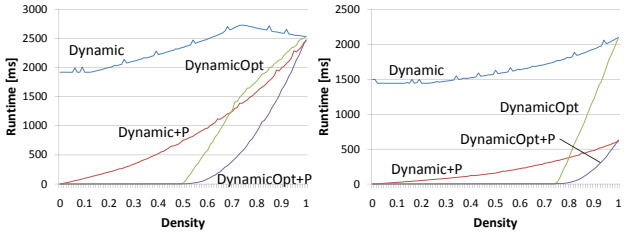


(c) $minSup = 50\%$ (d) $minSup = 75\%$

Figure 7: Runtime evaluation w.r.t. $|T|$.



(a) $minSup = 0.1$ (b) $minSup = 0.25$



(c) $minSup = 0.5$ (d) $minSup = 0.75$

Figure 8: Runtime evaluation w.r.t. the density.

7.1.2 *Effect of the Density*

We now evaluate the effectiveness of our pruning strategy w.r.t. the density. *minSup* is important here too, so we report results for different values in Figure 8. The pruning works well for datasets with low density and has no effect on the runtime for higher densities. The reason is straightforward; the higher the density, the higher the probability that a given itemset is frequent and, thus, cannot be pruned. Regarding the effect of *minSup*; a larger *minSup* value decreases the probability that itemsets are frequent and therefore increases the number of computations that can be pruned. The break-even point between pruning and non-pruning in our experiments is when the density is approximately twice the *minSup* value, since, due to the method of creating our datasets, this corresponds to the expected support. At this value, all itemsets are expected to be frequent.

Overall, with reasonable parameter settings our pruning strategies achieve a significant speed-up for the identification of probabilistic frequent itemsets.

7.1.3 *Effect of minSup*

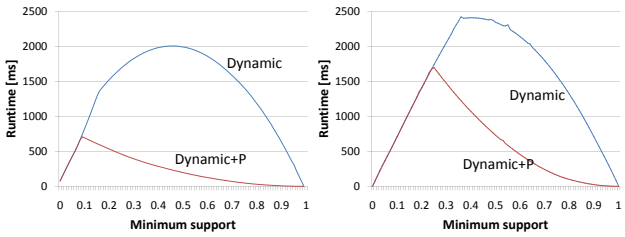
Figure 9 shows the influence of *minSup* on the runtime when using different densities. The runtime of **Dynamic** directly correlates with the size of the dynamic computation matrix (Figure 5). A low *minSup* value leads to few matrix rows which need to be computed, while a high *minSup* value leads to a slim row width (see Figure 5). The total number of matrix cells to be computed is $minSup * (|T| - minSup + 1)$, with a maximum at $minSup = \frac{|T|+1}{2}$. As long as the *minSup* value is below the expected support value, the approach with pruning shows similar characteristics; in this case, almost all item(sets) are expected to be frequent. However, the speed-up due to the pruning rapidly increases for *minSup* above this break-even point.

7.2 Evaluation of the Probabilistic Frequent Itemset Mining Algorithms

Experiments for the probabilistic frequent itemset mining algorithms were run on a subset of the real-world dataset *accidents*[4], denoted by *ACC*. It consists of 340,184 transactions and 572 items whose occurrences in transactions were randomized; with a probability of 0.5, each item appearing for certain in a transaction was assigned a value drawn from a uniform distribution in $(0, 1]$. Here we use **AP** to denote the Apriori-based and **IP** for the incremental probabilistic itemset mining algorithms.

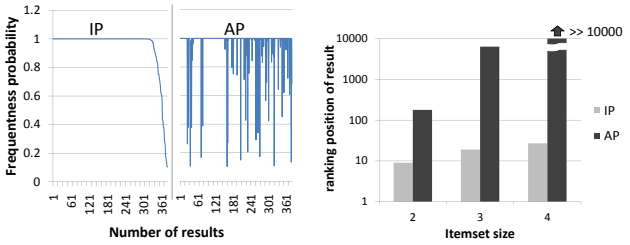
We performed Top-k queries on the first 10,000 transactions of *ACC* using a $minSup = 500$ and $\tau = 0.1$. Figure 10(a) shows the result of **IP**. Note that the frequentness probability of the resulting itemsets is monotonically de-

[4]The *accidents* dataset [8] was derived from the Frequent Itemset Mining Dataset Repository (http://fimi.cs.helsinki.fi/data/)



(a) Density = 0.2 (b) Density = 0.5

Figure 9: Runtime evaluation w.r.t. *minSup*.



(a) Output: AP vs. IP

(b) Effectiveness of ranking queries

Figure 10: Effectiveness of AP vs IP.

creasing. In contrast, **AP** returns probabilistic frequent itemsets in the classic way; in descending order of their size, i.e. all itemsets of size one are returned first, etc. While both approaches return probabilistic frequent itemsets, **AP** returns an arbitrary frequentness probability order, while **IP** returns the most relevant itemsets first.

Next we performed ranking queries on the first $100,000$ itemsets (Figure 10(b)). In this experiment, our aim was to find the m-itemset X with the highest frequency probability of all m-itemsets, where $m \in \{2, 3, 4\}$. We measured the number of itemsets returned before X. It can be seen that the speed up factor for ranking (and thus top-k queries) is several orders of magnitude and increases exponentially in the length of requested itemset length. The reason is that **AP** must return all frequent itemsets of length $m-1$ before processing itemsets of length m, while **IP** is able to quickly rank itemsets in order of their frequentness probability, therefore leading to better quality results delivered to the user much earlier.

8. CONCLUSION

The Probabilistic Frequent Itemset Mining (PFIM) problem is to find itemsets in an uncertain transaction database that are (highly) likely to be frequent. To the best of our knowledge, this is the first paper addressing this problem under possible worlds semantics. We presented a framework for efficient probabilistic frequent itemset mining. We theoretically and experimentally showed that our proposed dynamic computation technique is able to compute the exact support probability distribution of an itemset in linear time w.r.t. the number of transactions instead of the exponential runtime of a non-dynamic computation. Furthermore, we demonstrated that our probabilistic pruning strategy allows us to prune non-frequent itemsets early leading to a large performance gain. In addition, we introduced an iterative itemset mining framework which reports the most likely frequent itemsets first.

9. REFERENCES

[1] P. Agrawal, O. Benjelloun, A. Das Sarma, C. Hayworth, S. Nabar, T. Sugihara, and J. Widom. "Trio: A system for data, uncertainty, and lineage". In *Proc. Int. Conf. on Very Large Databases (VLDB'06)*, 2006.

[2] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proc. ACM SIGMOD Int. Conf. on Management of Data (SIGMOD'94), Minneapolis, MN*, 1994.

[3] L. Antova, T. Jansen, C. Koch, and D Olteanu. "Fast and Simple Relational Processing of Uncertain Data". In *Proc. 24th Int. Conf. on Data Engineering (ICDE'08), Cancún, México*, 2008.

[4] O. Benjelloun, A. D. Sarma, A. Halevy, and J. Widom. "ULDBs: Databases with Uncertainty and Lineage". In *Proc. Int. Conf. on Very Large Databases (VLDB'06)*, 2006.

[5] Chun Kit Chui and Ben Kao. A decremental approach for mining frequent itemsets from uncertain data. In *The 12th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)*, pages 64–75, 2008.

[6] Chun Kit Chui, Ben Kao, and Edward Hung. Mining frequent itemsets from uncertain data. In *11th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining, PAKDD 2007, Nanjing, China*, pages 47–58, 2007.

[7] N. Dalvi and D. Suciu. "Efficient query evaluation on probabilistic databases". *The VLDB Journal*, 16(4):523–544, 2007.

[8] Karolien Geurts, Geert Wets, Tom Brijs, and Koen Vanhoof. Profiling high frequency accident locations using association rules. In *Proceedings of the 82nd Annual Transportation Research Board, Washington DC. (USA), January 12-16*, page 18pp, 2003.

[9] Jiawei Han, Jian Pei, and Yiwen Yin. Mining frequent patterns without candidate generation. *SIGMOD Rec.*, 29(2):1–12, 2000.

[10] H.-P. Kriegel, P. Kunath, M. Pfeifle, and M. Renz. "Probabilistic Similarity Join on Uncertain Data". In *Proc. 11th Int. Conf. on Database Systems for Advanced Applications, Singapore, pp. 295-309*, 2006.

[11] Carson Kai-Sang Leung, Christopher L. Carmichael, and Boyu Hao. Efficient mining of frequent patterns from uncertain data. In *ICDMW '07: Proceedings of the Seventh IEEE International Conference on Data Mining Workshops*, pages 489–494, 2007.

[12] C. Re, N. Dalvi, and D. Suciu. "Efficient top-k query evaluation on probalistic databases". In *Proc. 23rd Int. Conf. on Data Engineering, Istanbul, Turkey*, 2007.

[13] P. Sen and A. Deshpande. "Representing and querying correlated tuples in probabilistic databases". In *Proc. 23rd Int. Conf. on Data Engineering, Istanbul, Turkey*, 2007.

[14] M.A. Soliman, I.F. Ilyas, and K. Chen-Chuan Chang. "Top-k Query Processing in Uncertain Databases". In *Proc. 23rd Int. Conf. on Data Engineering, Istanbul, Turkey*, pages 896–905, 2007.

[15] Florian Verhein and Sanjay Chawla. Geometrically inspired itemset mining. In *IEEE International Conference on Data Mining (ICDM 2006)*, pages 655–666. IEEE Computer Society, 2006.

[16] Yi Xia, Yirong Yang, and Yun Chi. Mining association rules with non-uniform privacy concerns. In *DMKD '04: Proceedings of the 9th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery*, pages 27–34, 2004.

[17] K. Yi, F. Li, G. Kollios, and D. Srivastava. "Efficient Processing of Top-k Queries in Uncertain Databases". In *Proc. 24th Int. Conf. on Data Engineering (ICDE'08), Cancún, México*, 2008.

[18] Qin Zhang, Feifei Li, and Ke Yi. Finding frequent items in probabilistic data. In Jason Tsong-Li Wang, editor, *SIGMOD Conference*, pages 819–832. ACM, 2008.