Continuous Inverse Ranking Queries in Uncertain Streams

Thomas Bernecker[1], Hans-Peter Kriegel[1], Nikos Mamoulis[2], Matthias Renz[1] and Andreas Zuefle[1]

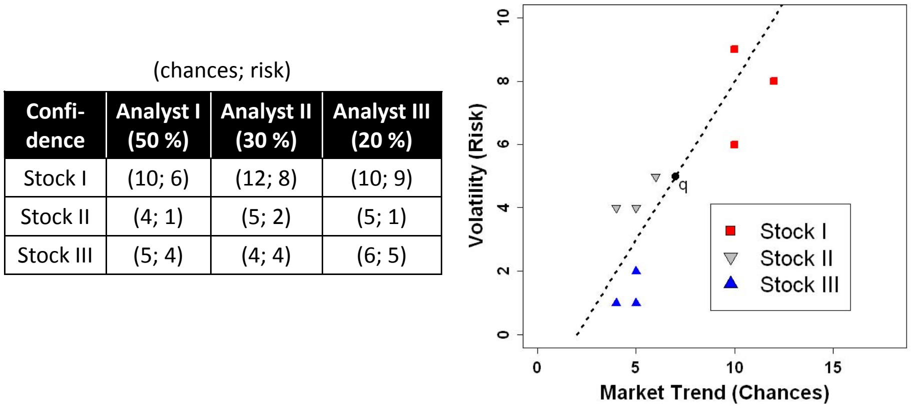
[1] Department of Computer Science, Ludwig-Maximilians-Universität München
{bernecker,kriegel,renz,zuefle}@dbs.ifi.lmu.de
[2] Department of Computer Science, University of Hong Kong
nikos@cs.hku.hk

Abstract. This paper introduces a scalable approach for continuous inverse ranking on uncertain streams. An uncertain stream is a stream of object instances with confidences, e.g. observed positions of moving objects derived from a sensor. The confidence value assigned to each instance reflects the likelihood that the instance conforms with the current true object state. The inverse ranking query retrieves the rank of a given query object according to a given score function. In this paper we present a framework that is able to update the query result very efficiently, as the stream provides new observations of the objects. We will theoretically and experimentally show that the query update can be performed in linear time complexity. We conduct an experimental evaluation on synthetic and real-world data, which demonstrates the efficiency of our approach.

1 Introduction

Recently, it has been recognized that many applications dealing with spatial, temporal, multimedia, and sensor data have to cope with uncertain or imprecise data. For instance, in the spatial domain, the locations of objects usually change continuously, thus the positions tracked by GPS devices are often imprecise. Similarly, vectors of values collected in sensor networks (e.g., temperature, humidity, etc.) are usually inaccurate, due to errors in the sensing devices or time delays in the transmission. Finally, images collected by cameras may have errors, due to low resolution or noise. As a consequence, there is a need to adapt storage models and indexing/search techniques to deal with uncertainty.

Special formulations of queries are required in order to take the uncertainty of the data into account. In this paper, we focus on the *probabilistic inverse ranking* (PIR) query on uncertain streaming data, i.e. the data change with elapsing time. While PIR queries have been studied for static data [1], to the best of our knowledge, there is no previous work in the context of dynamic data or data streams. Given a stream of uncertain objects, a user-defined score function S that ranks the objects and a user-defined (uncertain) query object q, a PIR query computes all the possible ranks of q associated with a probability. The PIR query is important for many real applications including financial data analysis, sensor data monitoring and multi-criteria decision making where one might be interested in the identification of the rank (significance) of a particular



(chances; risk)

Confi-dence	Analyst I (50 %)	Analyst II (30 %)	Analyst III (20 %)
Stock I	(10; 6)	(12; 8)	(10; 9)
Stock II	(4; 1)	(5; 2)	(5; 1)
Stock III	(5; 4)	(4; 4)	(6; 5)

(a) Stock example values.

(b) Stock example chart.

Fig. 1. Chances and Risk predictions by three analysts for three stocks.

object among peers. Consider the exemplary application illustrated in Figure 1(a): A financial decision support system monitors diverse prognostic attributes of a set of three stocks, e.g. predicted market trend (chances) and volatility (risk), which are used to rate the profitability of the stocks according to a given score function. As it can be observed, the chance and risk estimations are not unique among different analysts and each analyst is given a different confidence level. Figure 1(b) shows graphically the three stocks with their respective analyst predictions and the query object q. Here we assume that we are given a score function defined as $S = (Chances - Risk)$. The dotted line in Figure 1(b) denotes all points x where $S(x) = S(q)$, i.e. all points with the same score as q. Any instance located to the right of this line has a higher score than q, while any instance to the left has a lower score. Therefore, we can safely assume that Stock II has a lower score than q while Stock III certainly has a higher score than q. However, the relative ranking of Stock I with respect to q is uncertain. While two of three analysts (with a total confidence of 80%) would rank Stock I higher than q, the third analyst would rank it lower. Thus, the PIR query for q returns that q is on rank two with a probability of 80%, on rank three with a probability of 20% and definitely not on rank one or four. This result can be used to answer questions like "Given a score function, what is the likelihood that a query stock q is one of the global top-3 best stocks?". The problem we study in this paper is how to efficiently update these likelihoods when the analysts release new estimations on a ticker stream.

As another example (taken from [2]), for a newborn, we may be interested in his/her health compared with other babies, in terms of height, weight, and so on. In this case, we can infer the newborn's health from his/her rank among others. Note that data of newborn babies in a hospital are confidential. Thus, for the sake of privacy preservation, such information is usually perturbed by adding synthetic noise or generalized by replacing exact values with uncertain intervals, before releasing them for research purposes. Thus, in these situations, we can conduct a PIR query over uncertain data (perturbed or generalized) in order to obtain all possible ranks that a newborn may have with high confidence. In addition, we may want the distribution of possible ranks of the baby to be dynamically updated, as new data arrive, in order to be confident that the baby's status remains good compared to new cases. Therefore, rank updates for the query (baby) have to be applied, as new measurements arrive from a stream.

The rest of the paper is organized as follows: In the next section, we survey existing work in the field of managing and querying uncertain data streams. In Section 3, we formally define the problem of probabilistic inverse ranking on data streams. Our approach for solving the problem efficiently is described in Section 5. In Section 6, we generalize the problem by additionally considering uncertain queries. We experimentally evaluate the efficiency of our approach in Section 7 and conclude the paper in Section 8.

2 Related Work

In this paper, we focus on inverse similarity ranking of uncertain vector data. A lot of work was performed in the direction of ranking among uncertain data [3–7], but there is limited research on the inverse variant of ranking uncertain data [1]. In a nutshell, there are two models for capturing uncertainty of objects in a multi-dimensional space. In the *continuous* uncertainty model, the uncertain values of an object are represented by a continuous probability density function (pdf) within the vector space. This type of representation is often used in applications where the uncertain values are assumed to follow a specific probability density function (pdf), e.g. a Gaussian distribution [5]. Similarity search methods based on this model involve expensive integrations of the pdf's, thus special approximation techniques for efficient query processing are typically employed [8]. In the *discrete* uncertainty model, each object is represented by a discrete set of alternative values, and each value is associated with a probability. The main motivation of this representation is that, in most real applications, data are collected in a discrete form (e.g., information derived from sensor devices). The uncertain stream data, as assumed in this paper, correspond to the discrete uncertainty model which also complies with the x-relations model used in the *Trio* system [9].

In order to deal with massive datasets that arrive online and have to be monitored, managed and mined in real time, the data stream model has become popular. Surveys of systems and algorithms for data stream management are given in [10] and [11]. A generalized stream model, the probabilistic stream model, was introduced in [12]. In this model, each item of a stream represents a discrete probability distribution together with a probability that the element is not actually present in the stream. There has been interesting work on clustering uncertain streams [13], as well as on processing more complex event queries over streams of uncertain data [14]. [15] presents algorithms that capture essential features of the stream, such as quantiles, heavy-hitters, and frequency moments. To the best of our knowledge, this paper is the first addressing the processing of inverse ranking queries on uncertain streams.

The *inverse ranking query* on static data was first introduced by Li [2]. Chen et al. [1] apply inverse ranking to probabilistic databases by introducing the *probabilistic inverse ranking query (PIR)*. Apart from considering only static data, their PIR query definition varies from ours, since its output consists of all possible ranks for a query object q, for which q has a probability higher than a given threshold. Another approach for answering PIR queries has been proposed by [16] which computes the expected inverse rank of an object. The expected inverse rank can be computed very efficiently, however, it lacks from a semantic point of view. In particular, an object that has a very high chance to be on rank one, may indeed have a expected rank far from one, and may not be in the result using expected ranks. Thus, no conclusion can be made about the actual rank probabilities if the expected rank is used, since the expected rank is an aggregation that drops important information.

3 Probabilistic Stream Database

Without loss of generality, we consider uncertain vector objects L in a d-dimensional vector space. That is, each object is assigned to a number of alternative locations l each associated with a probability value. For example, the alternative positions of an uncertain object are associated with observations derived from different sources of information (sensors). In our stock example the sources correspond to the assessments of the analysts and in the baby ranking example, the sources correspond to uncertain values uniformly sampled from the corresponding uncertain measurement range.

In this work, we adopt the discrete *x-relation model* proposed in the TRIO system [9], in which an uncertain database $\mathcal{D}$ consists of n uncertain objects which are each modelled by exactly one *x-tuple*. Each x-tuple T includes a number of tuples, which we call (possible) instances, as its alternatives. Each tuple $t \in T$ is associated with a probability $p(t)$, i.e. T follows a discrete probability distribution. Thus, an x-tuple T is a set of a bounded number of instances, subject to the constraint that $\sum_{t \in T} p(t) \leq 1$. Independence is assumed among the x-tuples. For simplicity, we also assume that $\sum_{t \in T} p(t) = 1$.[3] Following the popular possible worlds semantics, $\mathcal{D}$ is instantiated into a possible world with mutual independence of the x-tuples as follows:

Definition 1 (Possible Worlds). *Let $\mathcal{D} = \{T_1, ..., T_n\}$ and let $W = \{t_1, ..., t_n\}$ be any subset of tuples appearing in $\mathcal{D}$ such that $t_i \in T_i$. The probability of this world W is $P(W) = \prod_{j=1}^{n} p(t_j)$. If $P(W) > 0$, we say that W is a possible world, and we denote by $\mathcal{W}$ the set of all possible worlds.*

Definition 2 (Probabilistic Stream). *We define a probabilistic data stream, analogously to [12]. A probabilistic stream is a data stream $A = [x_0, \ldots, x_t, \ldots]$ in which each item x_t encodes a random variable reported at time t from the stream, corresponding to an object update. In particular, each x_t has the form $\langle O, L \rangle$, where O is an object ID and L is a location vector of length $|L|$. Each element $l \in L$ contains a location $l.loc \in \mathbb{R}^d$ and a probability $l.p$. In addition, we assume that $\sum_{l \in L} l.p = 1$, i.e. we assume that the objects have no existential uncertainty. i.e. that each object O is existentially certain.*

[3] For the problem of inverse ranking, this assumption means no loss of generality, since existential uncertainty can be modelled by simply adding to T an additional instance with a probability $1 - \sum_{t \in T} p(t)$ and a score of $-\infty$ (that is a distance of ∞ to the query).

Definition 3 (Probabilistic Stream Database). *A probabilistic stream database is an uncertain database connected to at least one probabilistic stream. Each stream item $x_t = \langle O, L \rangle$ at time t denotes an update of the uncertain object $O \in DB$.*[4] *Therefore, at time t, the x-relation describing object O is replaced by the new location distribution L coming from the stream.*

This probabilistic stream database model is very general and can be easily adapted to simulate popular stream models:

The *sliding window model* of size m can be simulated by imposing the following constraint to the probabilistic stream: For any two stream items $x_t = \langle O, L_t \rangle$, $x_s = \langle O, L_s \rangle, t < s$, of the same object O, it holds that if there are no other stream items between time t and s concerning the same object, it holds that L_{t+1} is derived from L_t by

- adding exactly one new instance to L_t, and
- removing the oldest instance of L_t if $|L_t| > m$

The probabilities $p(l), l \in L_t$ are often set to $p(l) = \frac{1}{|L_t|}$, but other distributions can be used. In particular, more recently observed instances can be given a higher probability thus leading to the *weighted sliding window model*. Additionally, the infinite sliding window model is obtained by setting $k = \infty$. In this work, the stream model is left abstract, as the proposed solutions are applicable for any such model.

4 Problem Definition

Next we define the problem to be solved in this paper. Given a set of n uncertain objects and a probabilistic stream A as defined above, our problem is to compute and update, for a given query object q and a given score function S the result of a probabilistic inverse ranking query PIR(q) at the initial time t_0 and and after each object update. Thereby, a PIR(q) query is formally defined as follows:

Definition 4 (Probabilistic Inverse Ranking Query). *Given an uncertain database $\mathcal{D}$ of size n, a query object q and a score function*

$$S : \mathcal{D} \to \mathbb{R}_o^+.$$

Assuming that only the top-k ranks are of interest, a probabilistic inverse ranking query PIR(q) returns for each $i \in [1, ..., k]$ the probability $P_q^t(i)$ that q is on rank i w.r.t. S, i.e. the probability that there exist exactly $i - 1$ objects $o \in \mathcal{D}$ such that $S(o) > S(q)$ at time t.

The challenge is to ensure that this can be done correctly in terms of the possible world semantics, and highly efficiently to allow online processing of the probabilistic stream A. Since the number of possible worlds at time t is exponential in the number n of uncertain stream objects at time t, these two challenges are conflicting. In the following we will propose an approach to compute $PIR(q)$ in $O(n^2)$ time from scratch, and to update it in $O(n)$ time when a new update is fetched from the stream. In addition, we will show how the result of PIR(q) can be efficiently updated, if the query object q is itself a stream object that changes frequently.

[4] O may also be a new object.

Table of Notations	
$\mathcal{D}$	An uncertain stream database.
n	The cardinality of $\mathcal{D}$.
q	A query vector in respect to which a probabilistic inverse ranking is computed.
k	The ranking depth that determines the number of ranking positions of the inverse ranking query result.
o_i	An uncertain stream object corresponding to a finite set of alternative vector point instances
p_i^t	The probability that o_i has a higher score than q at time t.
$P^t(i)$	The result of the inverse ranking at time t: The probability that q is at rank i at time t.
$P_{i,j}^t$	The probability that out of j processed objects, exactly i have a higher score than q at time t.
$P_{PBR}^t(i)$	The result of the Poisson Binomial Recurrence at time t: The probability i objects have a higher score than q at time t, if all objects o_i such that $p_i^t = 1$ are ignored.
$\hat{P}_{PBR}^t(i)$	The adjusted result of the Poisson Binomial Recurrence at time t: Identical to$P_{PBR}^t(i)$ except that the effect of the object that changes its position at time $t+1$ is removed from the calculation.
C^t	The number of object at time t such that $p_i^t = 1$.

Table 1. Table of notations used in this work.

5 Probabilistic Inverse Ranking

Consider an uncertain stream database $\mathcal{D}$ of size n, a query object q, a score function S and a positive integer k. Our algorithm basically consists of two modules:

- The *initial computation* of the probabilistic inverse ranking that computes for each rank $i \in [1, ..., k]$ the probability $P^t(i)$ that q is ranked on position i at the initial time t, when the query is issued. We show how this can be performed in $O(k \cdot n)$ time.
- The *incremental stream processing* that updates PIR(q) at time $t+1$, given the probabilistic inverse ranking at time t. Therefore, the probabilities $P^{t+1}(i)$ that Q is ranked on position i at time $t+1$ have to be computed given the $P^t(i)$. We show how this update can be done in O(k) time.

5.1 Initial Computation

For each object $o_j \in \mathcal{D}$ let p_j^t be the probability that o_j has a higher rank than q at time t, i.e. $p_j^t = P(S(o_j) > S(q))$. These probabilities can be computed in a single database scan. We can process the p_j^t successively by means of the *Poisson binomial recurrence* [17], as proposed in [18]. Therefore, let $P_{i,j}^t$ be the probability that, out of the j objects processed so far, exactly i objects have a higher score than q. This probability depends only on the two following events:

- $i-1$ out of the first $j-1$ processed objects have a higher score than q **and** o_j has a higher score than q.
- i out of the first $j-1$ processed objects have a higher score than q **and** o_j does not have a higher score than q.

This observation and the assumption of independence between stream objects can be used to formulate the following Poisson binomial recurrence:

$$P^t_{i,j} = P^t_{i-1,j-1} \cdot p^t_j + P^t_{i,j-1} \cdot (1 - p^t_j) \tag{1}$$

$$\text{with } P^t_{0,0} = 1 \text{ and } P^t_{i,j} = 0 \text{ for } i < 0 \text{ or } i > j.$$

When the last object of the database is processed, i.e. $j = n$, then $P^t_{i,j} = P^t_{i,n} \overset{Definition}{=} P^t(i+1)$.[5] Computing the $P^t_q(i+1)$ for $0 \leq i \leq k-1$ yields the probabilistic inverse ranking. In each iteration, we can omit the computation of any $P^t_{i,j}$ where $i \geq k$, since we are not interested in any ranks greater than k, and thus, are not interested in the cases where at least k objects have a higher score than q. In total, for each $0 \leq i < k$ and each $1 \leq j \leq n$, $P^t_{i,j}$ has to be computed resulting in an $O(k \cdot n)$ time complexity.

Equation 1 is only required for objects o_j for which $0 < p^t_j < 1$. Objects o_j for which $p^t_j = 0$ can safely be ignored in the initial computation, since they have no effect on the $P^t(i)$. For objects o_j for which $p^t_j = 1$, we use a counter C^t that denotes the number of such objects. Thus, when o_j is encountered in the initial computation, the Poisson Binomial Recurrence is avoided and C^t is incremented. The probabilities obtained from the Poisson Binomial Recurrence by ignoring objects such that $p^t_j = 1$ are denoted as $P^t_{PBR}(i)$, $0 \leq i \leq k$.

The probabilistic inverse ranking can be obtained from the $P^t_{PBR}(i)$, $0 \leq i \leq k$ and C^t as follows:

$$P^t(i) = P^t_{PBR}(i - 1 - C^t), \text{ for } C^t + 1 \leq i \leq C^t + 1 + k \tag{2}$$

$$P^t(i) = 0 \text{ otherwise}$$

Example 1. Assume that a database containing objects $o_1, ..., o_4$ and an inverse ranking query with query object q and $k = 2$. Assume that $p^t_1 = 0.1$, $p^t_2 = 0$, $p^t_3 = 0.6$ and $p^t_4 = 1$. To compute the initial inverse ranking, we first process o_1 using Equation 1:

$$P^t_{0,1} = P^t_{-1,0} \cdot p^t_1 + P^t_{0,0} \cdot (1 - p^t_1) = 0 \cdot 0.1 + 1 \cdot 0.9 = 0.9$$

$$P^t_{1,1} = P^t_{0,0} \cdot p^t_1 + P^t_{1,0} \cdot (1 - p^t_1) = 1 \cdot 0.1 + 0 \cdot 0.9 = 0.1$$

Next we process o_2 but notice that $p^t_2 = 0$ so o_2 can be skipped. Then, object o_3 requires an additional iteration of Equation 1:

$$P^t_{0,2} = P^t_{-1,1} \cdot p^t_2 + P^t_{0,1} \cdot (1 - p^t_2) = 0 \cdot 0.6 + 0.9 \cdot 0.4 = 0.36$$

$$P^t_{1,2} = P^t_{0,1} \cdot p^t_2 + P^t_{1,1} \cdot (1 - p^t_2) = 0.9 \cdot 0.6 + 0.1 \cdot 0.4 = 0.58$$

$P^t_{2,2}$ does not need to be computed since $2 = i \geq k = 2$.

Next we process o_4. Since $p^t_4 = 1$, only C^t has to be incremented to 1. At this point, we are done. We have obtained:

$$P^t_{PBR}(0) = 0.36 \text{ and } P^t_{PBR}(1) = 0.58$$

[5] The event that i objects have a higher score than q corresponds to the event that q is ranked on rank $i + 1$.

To get the final inverse ranking at time t, we use Equation 2 to obtain

$$P^t(1) = P^t_{PBR}(1-1-1) = P^t_{PBR}(-1) = 0$$

$$P^t(2) = P^t_{PBR}(2-1-1) = P^t_{PBR}(0) = 0.36$$

5.2 Incremental Stream Processing

A naive solution would apply the Poisson binomial recurrence (cf. Equation 1) whenever a new object location o_x is fetched from the stream. However, the expensive update which is linear in the size of the database would make online stream processing impractical for large databases. In the following, we show how we can update $P^{t+1}(i)$ for $1 \leq i \leq k$ in constant time using the results of the previous update iteration.

Without loss of generality, let o_x be the object for which a new position information is returned by the stream at time $t+1$. p^t_x (p^{t+1}_x) denotes the old (new) probability that o_x has a higher score than q.

Our update algorithm uses two phases:

- **Phase 1:** Removal of the effect of the old value distribution of uncertain object o_x. That is, removal of the effect of the probability p^t_x from the result $P^t_{PBR}(i), 0 \leq i < k$. This yields an intermediate result $\hat{P}^{t+1}_{PBR}(i), 0 \leq i < k$.
- **Phase 2** Incorporation of the new value distribution of uncertain object o_x. That is including the probability p^{t+1}_x in the intermediate result $\hat{P}^{t+1}(i), 0 \leq i < k$ obtained in Phase 1.

Phase 1 The following cases w.r.t. p^t_x have to be considered:

Case I: $p^t_x = 0$. This case occurs if o_x is a new object or if it is certain that o_x has a lower score than q at time t. Thus nothing has to be done to remove the effect of $p^t_x = 0$: $\hat{P}^t_{PBR}(i) = P^t_{PBR}(i)$.

Case II: $p^t_x = 1$, i.e. if it is certain that o_x has a higher score than q at time t. In this case we just have to decrement C^t by one to remove the effect of p^t_x. Thus $\hat{P}^t_{PBR}(i) = P^t_{PBR}(i)$ and $C^{t+1} = C^t - 1$.

Case III: $0 < p^t_x < 1$, i.e. it is uncertain whether o_x has a higher score than q at time t. To remove the effect of p^t_x on all $P^t_{PBR}(i)$ $(1 \leq i \leq k)$ we look at the last iteration of Equation 1, that was used to obtain $P^t_{PBR}(i)$, $0 \leq i \leq k$. Let o_l be the object that was incorporated in this iteration:

$$P^t_{PBR}(i) = P^{t\prime}_{PBR}(i-1) \cdot p^t_l + P^{t\prime}_{PBR}(i) \cdot (1 - p^t_l),$$

where $P^{t\prime}_{PBR}(i)$ describes the probability that i objects have a score higher than q, if (in addition to all objects o_i such that $P^t_i = 1$) o_l is ignored. Now we observe that the $P^t_{PBR}(i)$'s $(1 \leq i \leq k)$ are not affected by the order in which the objects are processed within the recursion. In particular, the $P^t_{PBR}(i)$'s do not change, if the objects are processed in an order that processes o_x last, thus we obtain:

$$P^t_{PBR}(i) = \hat{P}^t_{PBR}(i-1) \cdot p^t_x + \hat{P}^t_{PBR}(i) \cdot (1 - p^t_x),$$

This can be resolved to

$$\hat{P}^t_{PBR}(i) = \frac{P^t_{PBR}(i) - \hat{P}^t_{PBR}(i-1) \cdot p^t_x}{1 - p^t_x}. \tag{3}$$

With $i = 0$ we obtain

$$\hat{P}^t_{PBR}(0) = \frac{P^t_{PBR}(0)}{1 - p^t_x}, \tag{4}$$

because the probability $\hat{P}^t_{PBR}(-1)$ that exactly -1 objects have a higher score than q is zero. Since the $P^t_{PBR}(i)$'s for $0 \leq i \leq k-1$ are known from the previous stream processing iteration, $\hat{P}^t_{PBR}(0)$ can be easily computed using Equation 4. Now we can inductively compute $\hat{P}^t_{PBR}(i+1)$ by using $\hat{P}^t_{PBR}(i)$ for any i and exploiting Equation 3.

Phase 2 In Phase 2, the same cases have to be considered:

Case I: $p^{t+1}_x = 0$: Object o_x has no influence on the result at time $t+1$. Nothing has to be done. Thus $P^{t+1}_{PBR}(i) = \hat{P}^t_{PBR}(i)$.

Case II: $p^{t+1}_x = 1$: Object o_x certainly has a higher score than q. Thus $C^{t+1} = C^t + 1$ and $P^{t+1}_{PBR}(i) = \hat{P}^t_{PBR}(i)$.

Case III: 0¡$p^{t+1}_x < 1$: We can incorporate the new probability for o_x to be ranked higher than q, i.e. p^{t+1}_x to compute the new probabilistic inverse ranking by an additional iteration of the Poisson Binomial Recurrence:

$$P^{t+1}_{PBR}(i) = \hat{P}^t_{PBR}(i-1) \cdot p^{t+1}_x + \hat{P}^t_{PBR}(i) \cdot (1 - p^{t+1}_x).$$

Example 2. Reconsider Example 1 where, at time t, we obtained $C^t = 1$, $P^t_{PBR}(0) = 0.36$ and $P^t_{PBR}(1) = 0.58$. Now, assume that at time $t+1$ object o_3 changes its probability from 0.6 to 0.2, i.e. $p^t_3 = 0.6$ and $p^{t+1}_3 = 0.2$. Phase 1 starts using case III. Using Equation 4 we get:

$$\hat{P}^t_{PBR}(0) = \frac{P^t_{PBR}(0)}{1 - p^t_3} = \frac{0.36}{0.4} = 0.9$$

Using Equation 3 we also get:

$$\hat{P}^t_{PBR}(1) = \frac{P^t_{PBR}(1) - \hat{P}^t_{PBR}(0) \cdot p^t_3}{1 - p^t_3} = \frac{0.58 - 0.9 \cdot 0.6}{0.4} = 0.1$$

This completes Phase 1. In Phase 2, case III is chosen and we get:

$$P^{t+1}_{PBR}(0) = \hat{P}^t_{PBR}(-1) \cdot p^{t+1}_3 + \hat{P}^t_{PBR}(0) \cdot (1 - p^{t+1}_3) = 0 \cdot 0.2 + 0.9 \cdot 0.8 = 0.72$$

$$P^{t+1}_{PBR}(1) = \hat{P}^t_{PBR}(0) \cdot p^{t+1}_3 + \hat{P}^t_{PBR}(1) \cdot (1 - p^{t+1}_3) = 0.9 \cdot 0.2 + 0.1 \cdot 0.8 = 0.26$$

This completes the update step (C^t remains unchanged, i.e. $C^{t+1} = C^t$). The result is obtained analogously to Example 1 using Equation 2:

$$P^{t+1}(1) = P^{t+1}_{PBR}(1 - 1 - 1) = P^{t+1}_{PBR}(-1) = 0$$

$$P^{t+1}(2) = P^{t+1}_{PBR}(2-1-1) = P^{t+1}_{PBR}(0) = 0.72$$

Now assume, that at time $t+2$ object o_4 changes its probability from 1 to 0: In Phase 1, case II is used and C_t is decremented from 1 to 0 to obtain $C^{t+1} = 0$. In Phase 2, case I is used and nothing is done. We get:

$$P^{t+2}_{PBR}(0) = \hat{P}^{t+1}_{PBR}(0) = P^{t+1}_{PBR}(0) = 0.72$$

$$P^{t+2}_{PBR}(1) = \hat{P}^{t+1}_{PBR}(1) = P^{t+1}_{PBR}(1) = 0.26$$

We obtain the result using Equation 2:

$$P^{t+2}(1) = P^{t+2}_{PBR}(1-1-0) = P^{t+2}_{PBR}(0) = 0.72$$

$$P^{t+2}(2) = P^{t+2}_{PBR}(2-1-0) = P^{t+2}_{PBR}(0) = 0.36$$

The later example shows why we need to maintain k probability values at each point of time: Even though some of the k probabilities may not be required to obtain the result, they may be required to obtain the result at a later time.

Regarding the computational complexity, the following holds for both Phase 1 and Phase 2: Case I and II have a cost of $O(1)$ since either nothing has to be done, or only C^t has to be incremented or decremented. Case III has a total cost of $O(k)$ leading to a total runtime of $O(k)$ in the update step.

6 Uncertain Query

In the previous section we have assumed that the query object q is fixed, i.e. has a certain position in $\mathbb{R}^d$. We now consider the case in which the query is also given as an uncertain stream object. Similar to the database objects, we now assume that the query object Q^t is represented by a set of m alternative instances $Q = \{q_1^t, ..., q_m^t\}$ at time t. The probabilistic inverse ranking query PIR(Q) w.r.t. an uncertain query object Q can be computed by aggregating the probabilistic inverse ranking query results w.r.t. each instance q_i of Q. Formally,

$$P^t_Q(i) = \sum_{j=1..m} P^t_{q_j}(i) \cdot p(q_j)$$

for all $j \in \{1, \ldots, m\}$, where $p(q_j)$ denotes the probability that the query object is located at q_j and $P^t_{q_j}(i)$ is the probability that instance q_j us located at rank i. $P^t_{q_j}(i)$ can be computed and updated as proposed in Section 5.

In this scenario, the stream may return new position information of the query object as well. Generally, when the stream returns new position information of the query q, the probabilities of all objects being ranked before q may change. Consequently, the inverse ranking result usually needs to be recomputed from scratch, using the technique shown in Section 5.1. However, in most applications, the position of an object only changes slightly. Therefore, the probability of other objects to have a higher score than q normally does not change for most objects. We exploit this property as follows.

Let Q be the query object with alternative instances $q_1^t, ..., q_m^t \in Q$ at time t and let $S^t_{min}(Q)$ and $S^t_{max}(Q)$ denote the minimum and maximum among all possible scores derived from the instances of Q at time t. In the following we assume that new query object instances are reported from the stream at time $t+1$:

Lemma 1. *If $S^t_{min}(Q) \leq S^{t+1}_{min}(Q)$, then for any object x with $P^t_x = 0$ it holds that $P^{t+1}_x = 0$ assuming x has not been updated at time $t+1$.*

Proof.

$$\text{Assumption: } S^t_{min}(Q) \leq S^{t+1}_{min}(Q) \tag{5}$$

$$\text{Assumption: } \forall i : S^t(x_i) = S^{t+1}(x_i) \tag{6}$$

$$\text{Assumption: } P^t_x = 0 \tag{7}$$

$$(7) \Leftrightarrow \forall q \in Q, \forall x_i \in x : S^t(q) > S^t(x_i) \Leftrightarrow \forall x_i \in x : S^t_{min}(Q) > S^t(x_i) \tag{8}$$

$$Def : \forall q \in Q, \forall x_i \in x : S^{t+1}(q) \geq S^{t+1}_{min}(Q)$$

$$\overset{5}{\geq} S^t_{min}(Q)$$

$$\overset{8}{\geq} S^t(x_i)$$

$$\overset{6}{=} S^{t+1}(x_1)$$

$$\Rightarrow \forall q \in Q, \forall x_i \in x : S^{t+1}(q) \geq S^{t+1}(x_1)$$

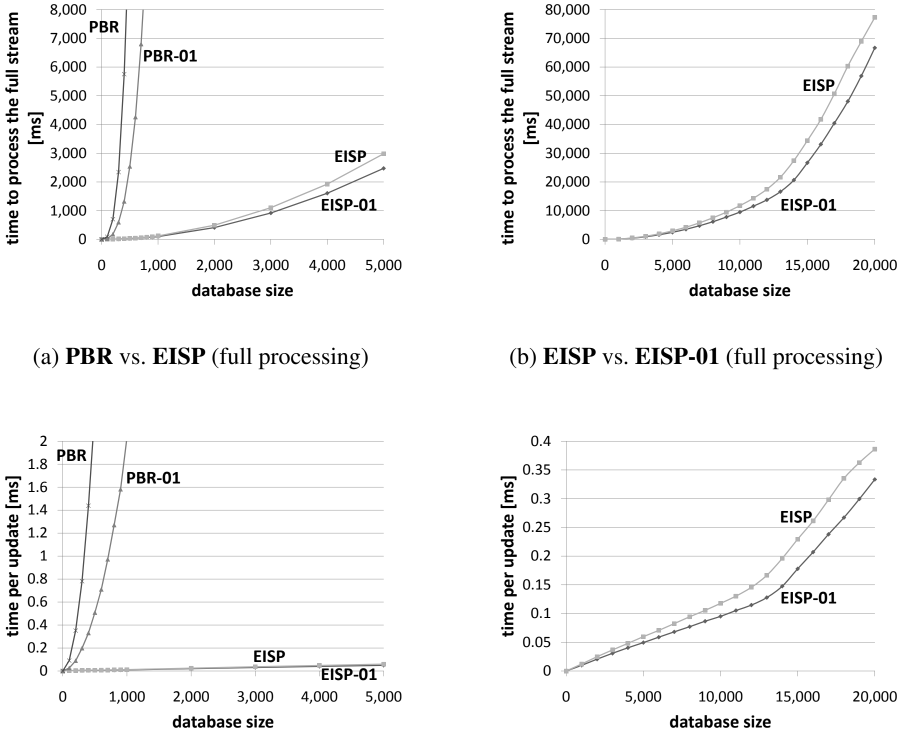
$$\Leftrightarrow P^{t+1}_x = 0$$

Lemma 2. *If $S^t_{max}(Q) \geq S^{t+1}_{max}(Q)$, then for any object x with $P^t_x = 1$ it holds that $P^{t+1}_x = 1$.*

Proof. Proof analogous to Lemma 1.

With the above Lemmata we can reduce the number of objects that have to be considered for re-computation of the inverse ranking at time $t+1$. Especially, if $S^t_{min}(Q) \leq S^{t+1}_{min}(Q) \wedge S^t_{max}(Q) \geq S^{t+1}_{max}(Q)$, then we have to compute p^{t+1}_x for those objects $x \in \mathcal{D}$ for which $p^t_x \notin \{0, 1\}$. For the remaining objects we have to update p^{t+1} and the inverse ranking probabilities considering the cases outlined in Section 5.2. Let us note, that the effectiveness of this pruning scheme highly depends on the grade of uncertainty of the objects. In our experiments, we show that the number of objects pruned from the computation of the inverse ranking can be very large.

A very drastic change of the position of the query object may, in the worst case, cause all probabilities $P^t_x, x \in \mathcal{D}$ to change. The incremental computation of Section 5 requires two computations: The removal of the effect of P^t_x and the incorporation of P^{t+1}_x for any object $x \in \mathcal{D}$ that changed its probability of having a higher score than q. In contrast, a computation from scratch requires only one computation for each $x \in \mathcal{D}$: the incorporation of P^{t+1}_x. Therefore, it is wise to switch to a full recomputation of the PIR if more than $\frac{n}{2}$ objects change their probability.



(a) **PBR** vs. **EISP** (full processing)

(b) **EISP** vs. **EISP-01** (full processing)

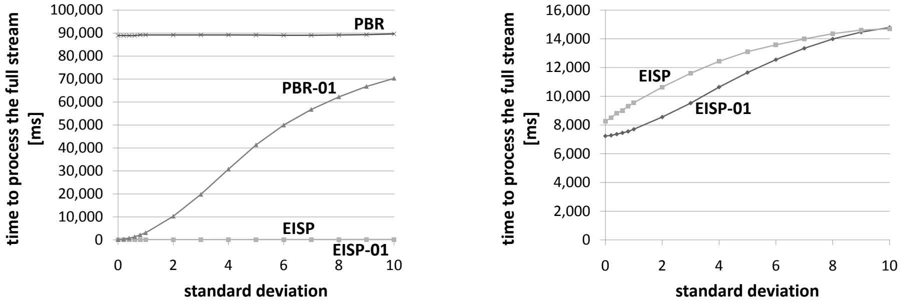
(c) **PBR** vs. **EISP** (single update)

(d) **EISP** vs. **EISP-01** (single update)

Fig. 2. Scalability of the PIR approaches.

7 Experiments

In the bigger part of the experimental evaluation, we use a synthetic dataset modelling a data stream with observations of 2-dimensional objects. The location of an object o_x at time t is modelled by m alternatives of a Gaussian distributed random variable X_{o_x} maintained in an array called *sample buffer*. For each $o_x \in \mathcal{D}$, the mean $E(X_{o_x})$ follows a uniform $[-10, 10]$-distribution in each dimension. The probabilistic stream A contains, for each $o_x \in \mathcal{D}$, exactly 10 alternative positions, that are randomly shuffled into the stream. Once a new alternative position of an object o_x is reported by the stream, it is stored in the sample buffer of o_x by replacing the least recently inserted one. We tune three parameters to evaluate the performance of the incremental PIR method described in Section 5: the database size n (default n=10,000), the standard deviation σ of uncertain object instances (default σ=5), and the sample buffer size m. For the scalability experiments, we chose $m = 3$. The evaluation of σ was performed with $m = 10$. In addition, we experimentally evaluate the influence of the degree of uncertainty



(a) **PBR** vs. **EISP** $n = 1,000$

(b) **EISP** vs. **EISP-01** $n = 10,000$

Fig. 3. Runtime evolution w.r.t. the standard deviation σ.

on the performance of our incremental PIR method (cf. Section 6). Finally, in Section 7.5, we examine the scalability issues on a real-world dataset.

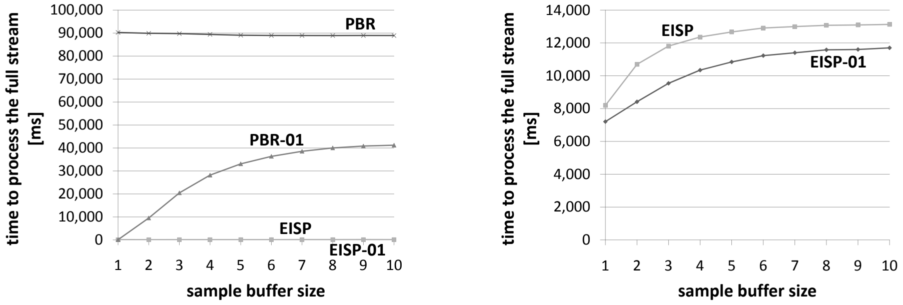
We denote our approach by **EISP** (Efficient Inverse Stream Processing). For comparison, we implemented the Poisson binomial recurrence based algorithm (abbreviated by **PBR**) as proposed by [1] that uses Equation 1, at each point of time where the stream provides a new observation. In addition, we evaluate the effect of the strategy proposed in Section 5 to avoid computation of objects o_x with a probability $p_x^t \in \{0, 1\}$ of having a higher score than q. This strategy will be denoted as *01-Pruning*. **EISP-01** and **PBR-01** denote the versions of **EISP** and **PBR**, respectively, that use *01-Pruning*.

7.1 Scalability

In the first experiment, we evaluate the scalability of **EISP**, **PBR**, **EISP-01** and **PBR-01** w.r.t. the database size n. We choose $k = n$ because if k is chosen constant and n is scaled up, the number of objects that certainly have a higher score than q will eventually reach k. In this case, *01-Pruning* will immediately notice that q cannot possibly be at one of the first k positions and will prune the computation. Then **EISP-01** and **PBR-01** have no further update costs. The results of these experiments are shown in Figure 2.

Figures 2(a) and 2(b) evaluate the total time required to process the whole stream, i.e. all $10{\cdot}n$ object updates. It can be observed that all four algorithms show a superlinear time complexity to process the whole stream (cf. Figure 2(a)). In addition, the utilization of *01-Pruning* leads to an improvement in the runtime. As the number of uncertain objects (i.e. the objects in the database for which it is uncertain whether they have a higher score than q and thus cannot be removed by *01-Pruning*) increases as well as the number of certain objects, we obtain a linear speed-up gain using *01-Pruning*.

For a more detailed evaluation of the update cost in each iteration, consider Figures 2(c) and 2(d): Here, the average time required for an update is shown. Note that the update cost of both **PBR** and **PBR-01** grows fast with n. This is explained by the



(a) **PBR** vs. **EISP** $n = 1,000$

(b) **EISP** vs. **EISP-01** $n = 10,000$

Fig. 4. Runtime evolution w.r.t. the sample buffer size m.

quadratic cost of $O(k \cdot n)$ (recall that we chose $k = n$) of the Poisson binomial recurrence at each update step. On the other hand, the update cost of $O(k)$ of **EISP** is linear to the number of database objects in this experiments (due to $k = n$). Here, *01-Pruning* has high influence on **PBR** but smaller effect on **EISP** especially for $n \leq 5,000$. The effect of *01-Pruning* may seem low for **EISP**, but note that in our experiments we measured the total time required for an update: This includes the time required to fetch a new location from the stream, compute its score, and recompute the total probability that the respective object has a higher score than q. This overhead is naturally required for any approach.

7.2 Standard Deviation σ

In the next experiment, we test the effect of the standard deviation σ on the distribution of location instances. Here, the total time required to process the whole stream was examined. The results are depicted in Figure 3. As **PBR** has to process all objects in each iteration of the inverse ranking, there is no influence of σ when this method is used (cf. Figure 3(a)). *01-Pruning* is able to reduce the runtime complexity having low values for σ, as many uncertain objects do not overlap with the score function and can therefore be neglected in each iteration. However, with an increasing value of σ, the cost of **PBR-01** approaches that of **PBR**, as the uncertainty ranges are spread over a greater range of the data space. **EISP** and **EISP-01** outperform the other methods by several orders of magnitude. Figure 3(b) shows that, for a small value of σ, there is a significant effect of *01-Pruning*. This becomes evident considering that the time overhead required to process the stream is about $7,000$ ms in this experiment. The reason is that for $\sigma = 0$ *01-Pruning* there exists no uncertainty, and thus all objects always have a probability of either 0 or 1 of having a higher score than q. Thus, Case *I* and Case *II* (cf. Section 5 are used in each update step and the Poisson binomial recurrence is never required. For $\sigma > 10$ most objects x have a probability $0 < P_x^t < 1$ of having a higher score than q. Thus Case *III* is used in each iteration and C^t approaches zero.

7.3 Sample Buffer Size m

Next, the total stream processing time was evaluated w.r.t. the sample buffer size m. Figure 4 shows that m has an impact on all inverse ranking methods. Again, using **PBR**, the number of considered alternatives only influences the required runtime if we apply *01-Pruning* (cf. Figure 4(a)). If m increases, the probability that an object o has both instances with a higher and smaller score than q increases, i.e. it is uncertain whether $S(q) > S(o)$. Figure 4(b) shows that even for $m = 10$, we obtain a relatively high performance gain using *01-Pruning*, since the alternatives remain in the extent of their probabilistic distribution. Thus, for many objects o, $S(q) > S(o)$ can be decided even for a large m.

7.4 Uncertain Query

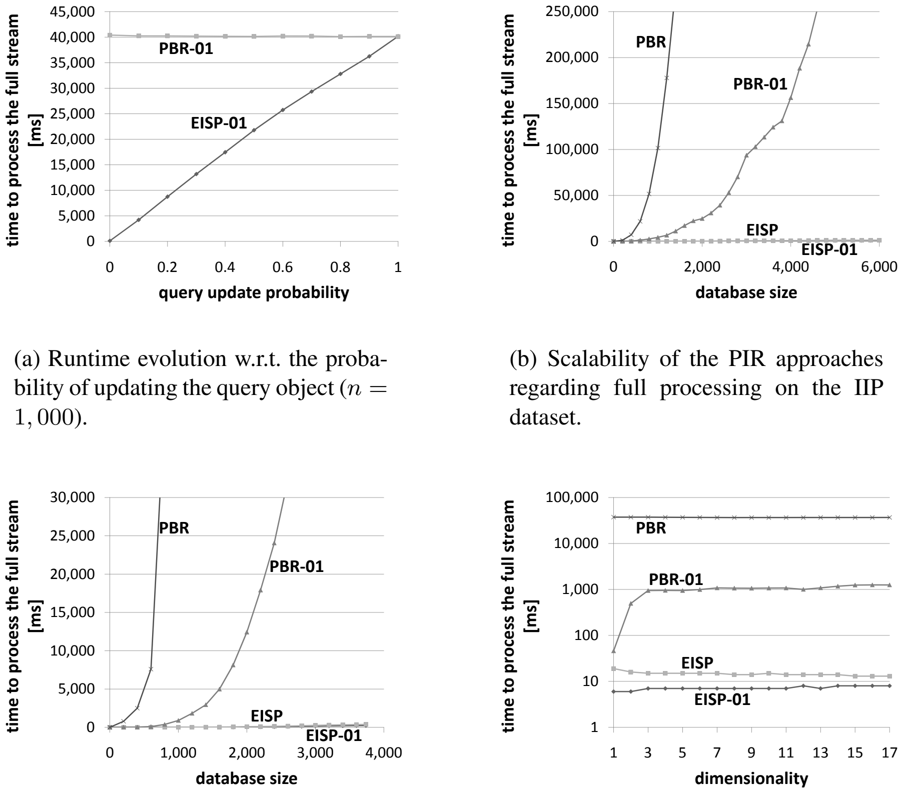
Finally, we evaluate the case that the query q is given as an uncertain stream object, now denoted by Q. As described in Section 6, the whole inverse ranking has to be recomputed by the **PBR** method if a position update of Q occurs. We test the performance of our adapted **EISP** method for this case.

For each time stamp t, we vary a probability value for Q of being updated and compare the versions of **PBR** with **EISP** that use *01-Pruning* in Figure 5(a). A value of 0 corresponds to the case that Q is certain, whereas a value of 1 assumes an update of Q in each iteration and thus forces **EISP-01** to always recompute the actual inverse ranking. It can be observed that the runtime required for processing the whole stream when using **EISP-01** increases linearly with a growing probability of the query object of being uncertain. This effect is due to the fact that the number of updates of Q and thus the number of complete re-computations have to be done according to the chosen probability value. As **PBR-01** does not depend on the uncertainty of Q because it recomputes the inverse ranking in each iteration anyway, its curve defines an upper asymptote to the curve of **EISP-01**.

7.5 Scalability Evaluation on Real-World Data

For an experimental evaluation of the scalability on real-world data, we first utilize the International Ice Patrol (IIP) Iceberg Sightings Dataset[6]. This dataset contains information about iceberg activity in the North Atlantic from 2001 to 2009. The latitude and longitude values of sighted icebergs serve as 2-dimensional values positions up to 6216 probabilistic objects, where each iceberg has been sighted at different positions. The stream consists of up to 10 positions of each iceberg which are ordered chronologically. Here again, we chose $m = 3$. Figure 5(b) indicates that the observations made for synthetic data can be transferred to real-world data. Note that for this dataset, *01-Pruning* is very effective, since the position of an iceberg has a very small variance. Many icebergs even appear to hold their position over time.

[6] The IIP dataset is available at the National Snow and Ice Data Center (NSIDC) web site (*http://nsidc.org/data/g00807.html*).



(a) Runtime evolution w.r.t. the probability of updating the query object ($n = 1,000$).

(b) Scalability of the PIR approaches regarding full processing on the IIP dataset.

(c) Scalability of the PIR approaches regarding full processing on the NBA dataset.

(d) Scalability of the PIR approaches w.r.t. the data dimensionality regarding full processing on the NBA dataset.

Fig. 5. Additional experiments.

The next set of experiments uses the NBA Dataset[7], containing information about North American basketball players. Each of the 3738 records in this dataset corresponds to the performance of one player in one season. In particular, each record contains a total of 17 dimensions representing the number of games played, the number of points scored, etc. in one given season between the years 1946 and 2006. For our experiments, we model players by uncertain stream objects, using a sliding window model of size $m = 3$, that is, a player is described by his performance in the last three years. The probabilistic stream contains all records of the dataset. For simplicity, the score function $s(x)$ we use is simply the sum of all (normalized) attributes. In this scenario, the semantic of a PIR query is to compute, for any given time, the rank of player Q with

[7] The NBA dataset was derived from *www.databasebasketball.com*.

respect to all NBA players. First, we evaluated the scalability of our PIR algorithm in Figure 5(c) using all 17 dimensions. It can be observed that the scalability is very similar to the IIP dataset, despite of the increased dimensionality. This is further evaluated in Figure 5(d) where we scale the number of dimensions. For the approach that do not utilize *01-Pruning*, the runtime appears to be constant in the number of dimensions. This can be explained by the fact that the dimensionality only affects the computation of the score of an object. Since we use the sum of all dimensions, we theoretically expect the algorithm to scale linearly in the number of dimensions, but the impact of this linear computation can be neglected. It can also be observed that, using *01-Pruning*, the runtime increases for low dimensions, and then becomes constant for higher dimensions. This can be explained by the uncertainty of the individual dimensions: The first dimension represents the number of games played by a player, which is a variable with a rather low deviation for each player. Even if a player has a very volatile performance, the number of games he played may be about the same. Therefore, the one-dimensional dataset has a rather low uncertainty, and thus, a lower runtime (cf. Section 7.2). However, a bad player may be replaced, and thus not play the full time, which is covered by the second dimension, that aggregates the number of minutes played in a year and has a higher deviation. The third dimension has the highest uncertainty, as it describes the number of points scored by a player in a year. After the third dimension, adding further dimensions does not significantly increase the total deviation of the sum (i.e. the score) of a player. In summary, increasing the dimensionality has no significant effect on the runtime, but may increase the uncertainty of the object, thus indirectly increasing the runtime.

8 Conclusions

In this paper, we proposed a general solution to efficiently answering probabilistic inverse ranking queries on streams. State-of-the-art approaches solving the PIR query problem for static data are not applicable for stream data due to the $O(k \cdot n)$ complexity of the Poisson binomial recurrence. We have shown theoretically and experimentally that the update cost of our approach is $O(k)$ and thus our approach is suitable for stream databases. Let us note that our framework can be easily adapted to tackle further variants of inverse ranking/top-k queries on streams: the threshold probabilistic inverse ranking query, that returns exactly those ranking positions i for which $P_q^t(i)$ is greater than a user-specified parameter $\tau \in [0,1]$, as proposed in [1], and the (threshold) probabilistic top-k query, that returns the probability that q is one of the best k objects in the database. The latter has many applications in decision-making environments.

One aspect of future work is to develop an approximate approach, which is able to efficiently cope with continuous data models. The idea is to derive for each database object O, a lower and an upper bound of the probability that O has a higher score than Q. Using these approximations, we can apply the concept of uncertain generating functions [19] in order to obtain an (initial) approximated result of a PIR query, which guarantees that the true result is bounded correctly. The problem at hand is to update these uncertain generating functions efficiently when an update is fetched from the stream.

References

1. Lian, X., Chen, L.: Probabilistic inverse ranking queries over uncertain data. In: Database Systems for Advanced Applications, 14th International Conference, DASFAA 2009, Brisbane, Australia, April 21-23, 2009. (2009) 35–50
2. Li, C.: Enabling data retrieval: By ranking and beyond. In: Ph.D. Dissertation, University of Illinois at Urbana-Champaign. (2007)
3. Yi, K., Li, F., Kollios, G., Srivastava, D.: Efficient processing of top-k queries in uncertain databases with x-relations. IEEE Trans. Knowl. Data Eng. **20**(12) (2008) 1669–1682
4. Cheng, R., Kalashnikov, D., Prabhakar, S.: Evaluating probabilistic queries over imprecise data. In: Proc. ACM SIGMOD Int. Conf. on Management of Data (SIGMOD'03), San Diego, CA). (2003)
5. Böhm, C., Pryakhin, A., Schubert, M.: Probabilistic ranking queries on gaussians. In: SSDBM. (2006) 169–178
6. Cormode, G., Li, F., Yi, K.: Semantics of ranking queries for probabilistic data and expected results. In: Proceedings of the 25th International Conference on Data Engineering, ICDE 2009, March 29-April 2, 2009, Shanghai, China. (2009)
7. Soliman, M., Ilyas, I.: Ranking with uncertain scores. In: Proceedings of the 25th International Conference on Data Engineering, ICDE 2009, March 29-April 2, 2009, Shanghai, China. (2009) 317–328
8. Tao, Y., Cheng, R., Xiao, X., Ngai, W., Kao, B., Prabhakar, S.: Indexing multi-dimensional uncertain data with arbitrary probability density functions. In: Proc. 31th Int. Conf. on Very Large Data Bases (VLDB'05), Trondheim, Norway. (2005) 922–933
9. Agrawal, P., Benjelloun, O., Das Sarma, A., Hayworth, C., Nabar, S., Sugihara, T., Widom, J.: Trio: A system for data, uncertainty, and lineage. In: Proc. Int. Conf. on Very Large Databases (VLDB'06). (2006)
10. Babcock, B., Babu, S., Datar, M., Motwani, R., Widom, J.: Models and issues in data stream systems. In: PODS, New York, NY, USA, ACM (2002) 1–16
11. Muthukrishnan, S.: Data streams: algorithms and applications. Found. Trends Theor. Comput. Sci. **1**(2) (2005) 117–236
12. Jayram, T.S., Kale, S., Vee, E.: Efficient aggregation algorithms for probabilistic data. In: SODA, Philadelphia, PA, USA, Society for Industrial and Applied Mathematics (2007) 346–355
13. Aggarwal, C.C., Yu, P.S.: A framework for clustering uncertain data streams. In: ICDE, Washington, DC, USA, IEEE Computer Society (2008) 150–159
14. Ré, C., Letchner, J., Balazinksa, M., Suciu, D.: Event queries on correlated probabilistic streams. In: SIGMOD, New York, NY, USA, ACM (2008) 715–728
15. Cormode, G., Garofalakis, M.: Sketching probabilistic data streams. In: SIGMOD, New York, NY, USA, ACM (2007)
16. Lee, M.C.K., Ye, M., Lee, W.C.: Reverse ranking query over imprecise spatial data. In: COM.GEO. (2010)
17. Lange, K.: Numerical analysis for statisticians. In: Statistics and computing. (1999)
18. Bernecker, T., Kriegel, H.P., Mamoulis, N., Renz, M., Züfle, A.: Scalable probabilistic similarity ranking in uncertain databases. IEEE Trans. Knowl. Data Eng. **22**(9) (2010) 1234–1246
19. Bernecker, T., Emrich, T., Kriegel, H.P., Mamoulis, N., Renz, M., Züfle, A.: A novel probabilistic pruning approach to speed up similarity queries in uncertain databases. to appear in ICDE'11. preliminary version availabe at the corr archive (www.arxiv.org). article-id: arxiv:1101.2613v1. (2011)