Distrib Parallel Databases (2009) 26: 29–65
DOI 10.1007/s10619-009-7043-x

Continuously monitoring top-*k* uncertain data streams: a probabilistic threshold method

Ming Hua · Jian Pei

Published online: 23 June 2009

Abstract Recently, uncertain data processing has become more and more important. Although a significant amount of previous research explores various continuous queries on data streams, continuous queries on uncertain data streams have seldom been investigated. In this paper, we formulate a novel and challenging problem of continuously monitoring top-k uncertain data streams, and propose a probabilistic threshold method. We develop four algorithms systematically: a deterministic exact algorithm, a randomized method, and their space-efficient versions using quantile summaries. An extensive empirical study using real data sets and synthetic data sets is reported to verify the effectiveness and the efficiency of our methods.

Keywords Uncertain streams · Probabilistic threshold top-k queries · Query processing

1 Introduction

In some emerging applications such as large sensor networks, data keep arriving in fast pace and thus can be modeled as data streams [5]. As an effective means of data stream processing, continuous queries on data streams have been investigated extensively (e.g., [7, 8, 24]). Different from conventional one-time queries on static

We are grateful to the anonymous reviewers and Dr. Ihab F. Ilyas, the guest editor, for their constructive comments which help to improve the quality of the paper substantially.
The research of Ming Hua and Jian Pei is supported in part by an NSERC Discovery grant and an NSERC Discovery Accelerator Supplements grant. All opinions, findings, conclusions and recommendations in this paper are those of the authors and do not necessarily reflect the views of the funding agencies.

M. Hua (✉) · J. Pei
Simon Fraser University, Burnaby, Canada
e-mail: mhua@cs.sfu.ca

Table 1 Data for segment *SEGK*715001 for 07/15/2001 in ARTIMIS Data Archives (Number of Lanes: 4)

# Time	Samp	Speed	Volume	Occupancy
00 : 01 : 51	30	47	575	6
00 : 16 : 51	30	48	503	5
00 : 31 : 51	30	48	503	5
00 : 46 : 51	30	49	421	4
01 : 01 : 52	30	48	274	5
01 : 16 : 52	30	42	275	14
...				

data, a continuous query monitors one or multiple data streams and online updates the answers to the query as the underlying data evolve.

Interestingly and importantly, uncertainty is inherent in some data stream applications due to factors like limitations of equipment, delay or loss in data transfer, and complex application semantics.

Example 1 (Motivation) Sensors have been extensively used to monitor traffic in large road networks. In such a network, a speed sensor deployed at a monitoring point can keep sending traffic records to a central server where each record is about the speed when a vehicle passes the monitoring point. For example, the ARTIMIS center in Cincinnati, Ohio/Kentucky reports the speed, volume and occupancy of road segments every 30 seconds [23]. Table 1 is a piece of sample data from ARTIMIS Data Archives, which can be found in http://www.its.dot.gov/JPODOCS/REPTS_TE/13767.html, together with more other applications examples.

Consider a simple continuous query—continuously reporting a list of top-10 monitoring points in the road network of the fastest vehicle speeds in the last 5 minutes. One interesting and subtle issue is how we should measure the vehicle speed at a monitoring point. Can we use some simple statistics like the average/median/maximum/minimum speed in the last 5 minutes? Each of such simple statistics may not capture the distribution of the data well. For example, if the average speeds are used, some outliers (e.g., speedy cars or broken cars) may affect the average speed heavily.

Suppose in the last 5 minutes, monitoring point A has 4 records of speeds $\{85, 75, 65, 60\}$ and monitoring point B has 3 records of speeds $\{100, 60, 55\}$. Which monitoring point has a faster speed? If we use the average speeds, point B is faster than point A.

Instead of using the average speeds, we can model each monitoring point as an uncertain object, and apply the possible worlds semantics [1, 15, 33, 50] to define the semantics of the query. Consider the speed of a monitoring point in the last 5 minutes as a random variable. The traffic records of speed reported at this monitoring point can be considered as a set of random samples drawn from the distribution of this random variable. The set of records at the monitoring point forms an uncertain object. Each record takes a probability to serve as the representative of this uncertain object.

Under the possible worlds semantics, each record in A takes a probability of $\frac{1}{4}$ to be a representative and each one in B takes a probability of $\frac{1}{3}$ to be a representative. The comparison of speeds at A and B is the comparison of two uncertain objects. Each possible combination of a record from A and a record from B is a possible world. For example, $\{A = 85, B = 100\}$ is a possible world, with probability $\frac{1}{4} \times \frac{1}{3} = \frac{1}{12}$. The speed of A is smaller than that of B in this possible world. There are in total 12 possible worlds. Combining the results from all possible worlds, point A takes a probability of $\frac{7}{12}$ to have a speed faster than that at point B. Point B has a probability of $\frac{1}{3}$ to have a speed faster than that at point A. In a probability of $\frac{1}{12}$ the speeds at the two points are the same.

Using the possible worlds semantics, we can refine the continuous query to continuously reporting a list of monitoring points such that each point has a probability of at least p to be in the top-10 lists in all possible worlds in the last 5 minutes, where p is a probabilistic threshold. This is an example of probabilistic threshold top-k queries. Since each sensor keeps reporting traffic records and generates a stream of records, the continuous query monitors uncertain data streams which take a probability passing a user specified threshold to be ranked among top-k.

Although a significant amount of previous research explores various continuous queries on data streams (please see Sect. 7 for a brief review), continuous queries on uncertain data streams have seldom been investigated, except for [11, 35]. As shown in Example 1 where point B has an average speed faster than point A, but point A has a higher probability to be faster than point B in possible worlds semantics, transforming uncertain data into "disguised" certain data by simple aggregation may not provide a meaningful solution in real applications.

Continuous queries on uncertain data streams pose several interesting challenges. First, it is important to develop practically meaningful models for continuous queries on uncertain data streams. The answers to queries should be sound in the sense of probability. Second, it is challenging to develop efficient methods for query answering. As indicated in the recent studies on uncertain data processing (also see Sect. 7 for a brief review), answering queries on uncertain data often involves much heavier cost than on certain data due to the combinatoric nature of the search space.

Top-k queries (also known as ranking queries) are a category of important queries in data analysis. Recently, there have been a few interesting studies on ranking queries on static uncertain data using the possible world semantics [30, 31, 40, 49, 54, 56]. Among them, probabilistic threshold top-k queries [30, 31] compute uncertain records taking a probability of at least p to be in the top-k list where p is a user specified probability threshold. Probabilistic threshold top-k queries capture the intuition of selecting the highly ranked records with good confidence. How can we extend probabilistic threshold top-k queries on static uncertain data to uncertain data streams?

In this paper, we focus on continuous probabilistic threshold top-k queries on uncertain data streams and make three major contributions.

Table 2 Notations frequently used

Notation	Description
o, o_i, v_i	instances in uncertain streams
O	uncertain streams
$\mathcal{O}$	a set of uncertain streams
$\Pr(o)$	membership probability of instance o
W_ω^t, W	a sliding window
w	possible worlds
$\mathcal{W}$	the complete set of possible worlds
$\Pr(w)$	the existence probability of possible world w
$\Pr^k(O)$	the top-k probability of object O
Q_p^k	a top-k query of probability threshold p
R	the ranking order of instances
$o_1 \prec o_2$	o_1 is ranked before o_2
$DS(o)$	the dominant set of o

First, we propose the novel uncertain data stream model and formulate continuous probabilistic threshold top-k queries in Sect. 2 under the possible worlds semantics [1, 15, 33, 50]. We argue that probabilistic threshold top-k queries are practically useful.

Second, we develop four algorithms systematically. A deterministic exact algorithm that computes the exact answer to a continuous probabilistic threshold top-k query is given in Sect. 3. It extends a technique in [30, 31] using Poisson binomial recurrence and incorporates a few stream-specific pruning techniques. A sampling algorithm is proposed in Sect. 4. It estimates the probability that an uncertain object being ranked top-k via sampling. A probabilistic quality guarantee is provided. Then, the approximation answer to a continuous probabilistic threshold top-k query is obtained based on the estimated probabilities. In Sect. 5, we apply the quantile summary techniques to devise space efficient versions of the deterministic algorithm and the sampling algorithm.

Last, an extensive empirical study is reported in Sect. 6 to verify the effectiveness and the efficiency of our methods. We review the related work in Sect. 7. Section 8 concludes the paper. Table 2 summarizes the notations frequently used in the paper.

2 Problem definition

In this section, we first introduce the uncertain stream model. Then, the continuous probabilistic threshold top-k queries are formulated.

2.1 The uncertain stream model

Consider a random variable X. Theoretically, the distribution of X can be characterized by a *probability density function* (*PDF* for short) if X is a continuous random variable, or a *probability mass function* (*PMF* for short) if X is a discrete random variable. In practice, the PDF or PMF of a random variable is often unavailable. Instead, a sample set of instances $x_1, \ldots, x_m$ are used to approximate the distribution of X, where each instance takes a membership probability. For an instance $x_i \in X$ ($1 \leq i \leq m$), the membership probability of x_i measures the likelihood that x_i will occur. The set of samples generated by the random variable can be considered as an uncertain object.

Definition 1 (Uncertain objects) An **uncertain object** is a set of instances $O = \{o_1, \ldots, o_m\}$ such that each instance o_i ($1 \leq i \leq m$) takes a **membership probability** $\Pr(o_i) > 0$, and $\sum_{i=1}^{m} \Pr(o_i) = 1$.

A *temporal random variable* is a random variable whose distribution evolves over time. To approximate the current distribution of a temporal random variable, practically we often use the observations of the variable in a recent time window as the sample instances. Conceptually, an uncertain data stream is a series of (discrete) instances generated by a random variable.

To keep our discussion simple, we assume a synchronous model. That is, each time instant is a positive integer, and at each time instant t ($t > 0$), an instance is collected for an uncertain data stream.

Definition 2 (Uncertain data stream, sliding window) An **uncertain data stream** is a (potentially infinite) series of instances $O = o_1, o_2, \ldots$. Time instants are positive integers. For instant t, $O[t]$ is the instance of stream O.

A **sliding window** W_ω^t is a selection operator defined as $W_\omega^t(O) = \{O[i] | (t - \omega) < i \leq t\}$, where $\omega > 0$ is called the **width** of the window.

For a set of uncertain data streams $\mathcal{O} = \{O_1, \ldots, O_n\}$, sliding window $W_\omega^t(\mathcal{O}) = \{W_\omega^t(O_i) | 1 \leq i \leq n\}$.

The distribution of an uncertain data stream O in a given sliding window W_ω^t is static. Thus, the set of instances $W_\omega^t(O)$ can be considered as an uncertain object. The membership probabilities for instances depend on how the instances are generated from the underlying random variable of $W_\omega^t(O)$. For example, if the instances are drawn using simple random sampling [36], then all instances take the same probability $\frac{1}{\omega}$. On the other hand, using other techniques like particle filtering [21] can generate instances with different membership probabilities. In this paper, we assume that the membership probabilities of all instances are identical. Some of our developed methods can also handle the case of different membership probabilities, which will be discussed in Sect. 8.

Definition 3 (Uncertain object in a sliding window) Let O be an uncertain data stream. At time instant $t > 0$, the set of instances of O in a sliding window W_ω^t is an uncertain object denoted by $W_\omega^t(O)$ ($1 \leq i \leq n$), where each instant $o \in W_\omega^t(O)$ has the membership probability $\Pr(o) = \frac{1}{\omega}$.

In this paper, we assume that the distributions of uncertain data streams are independent from each other. Handling correlations among uncertain data streams is an important direction that we plan to investigate as future study. It will be discussed in Sect. 8. The uncertain data in a sliding window carries the possible worlds semantics [1, 15, 33, 50].

Table 3 An uncertain data stream. (Sliding window width $\omega = 3$. W_3^t contains time instant $t-2$, $t-1$ and t. W_3^{t+1} contains time instant $t-1$, t and $t+1$)

Time instant	# Time	Speeds at A	Speeds at B	Speeds at C	Speeds at D
$t-2$	00 : 01 : 51	$a_1 = 15$	$b_1 = 6$	$c_1 = 14$	$d_1 = 4$
$t-1$	00 : 16 : 51	$a_2 = 16$	$b_2 = 5$	$c_2 = 8$	$d_2 = 7$
t	00 : 31 : 51	$a_3 = 13$	$b_3 = 1$	$c_3 = 2$	$d_3 = 10$
$t+1$	00 : 46 : 51	$a_4 = 11$	$b_4 = 6$	$c_4 = 9$	$d_4 = 3$
...					

Definition 4 (Possible worlds) Let $\mathcal{O} = \{O_1, \ldots, O_n\}$ be a set of uncertain data streams. A **possible world** $w = \{v_1, \ldots, v_n\}$ in a sliding window W_ω^t is a set of instances such that one instance is taken from the uncertain object of each stream in W_ω^t, i.e., $v_i \in W_\omega^t(O_i)$ $(1 \le i \le n)$. The **existence probability** of w is $\Pr(w) = \prod_{i=1}^n \Pr(v_i) = \prod_{i=1}^n \frac{1}{\omega} = \omega^{-n}$. The complete set of possible worlds of sliding window $W_\omega^t(\mathcal{O})$ is denoted by $\mathcal{W}(W_\omega^t(\mathcal{O}))$.

Corollary 1 *For a set of uncertain data streams $\mathcal{O} = \{O_1, \ldots, O_n\}$ and a sliding window $W_\omega^t(\mathcal{O})$, the total number of possible worlds is $\|\mathcal{W}(W_\omega(t))\| = \omega^n$, and*

$$\Pr(\mathcal{W}(W_\omega^t(\mathcal{O})) = \sum_{w \in \mathcal{W}(W_\omega^t(\mathcal{O}))} \Pr(w) = 1.$$

When it is clear from the context, we write $\mathcal{W}(W_\omega^t(\mathcal{O}))$ as $\mathcal{W}$ and $W_\omega^t(\mathcal{O})$ as W or W^t for the sake of simplicity.

Example 2 (Uncertain streams) Consider Example 1 again. Speed sensors are deployed to monitor traffic in a road network. The vehicle speed at each monitoring point can be modeled as a **temporal random variable**. To capture the distribution of such a temporal random variable, a speed sensor at the monitoring point reports the speed readings every 30 seconds. Therefore, the speed readings reported by each speed sensor is an **uncertain stream**. Each reading is an **instance** of the stream. A **sliding window** of length 3 at time t contains the last 3 readings (that is, the readings in the last 90 seconds) of each speed sensor.

Suppose there are four monitoring points A, B, C and D with speed readings shown in Table 3. At time t, sliding window W_3^t contains the records of speeds at time $t-2, t-1$ and t. $W_3^t(A) = \{a_1, a_2, a_3\}$ can be modeled as an **uncertain object**. So are $W_3^t(B)$, $W_3^t(C)$ and $W_3^t(D)$. Each instance in W_3^t takes **membership probability** $\frac{1}{3}$. There are $3^4 = 81$ possible worlds. Each possible world takes one instance from each object. For example, $\{a_1, b_3, c_2, d_1\}$ is a possible world. The existence probability of each possible world is $(\frac{1}{3})^4 = \frac{1}{81}$.

2.2 Continuous probabilistic threshold top-k queries

Top-k queries (also known as ranking queries) [18, 22, 46, 48] are a category of important queries in data analysis and data stream monitoring. In this paper, we consider the top-k selection query model [32].

Definition 5 (Top-k selection queries) For a set of instances S, each instance $o \in S$ is associated with a set of attributes A. Given a predicate P on A, a ranking function $f: S \rightarrow \mathbb{R}$ and a integer $k > 0$, a **top-k selection query** $Q^k_{P,f}$ returns a set of instances $Q^k_{P,f}(S) \subseteq S_P$, where S_P is the set of instances satisfying P, $|Q^k_{P,f}(S)| = \min\{k, |S_P|\}$ and $f(o) > f(o')$ for any instances $o \in Q^k_{P,f}(S)$ and $o' \in S_P - Q^k_{P,f}(S)$.

To keep our presentation simple, we assume that the top-k selection queries in our discussion select all instances in question. That is $S_P = S$. Those selection predicates can be implemented efficiently as filters before our ranking algorithms are applied. Moreover, we assume that the ranking function f in a top-k selection query can be efficiently applied to an instance o to generate a score $f(o)$. When it is clear from context, we also write $Q^k_{P,f}$ as Q^k for the sake of simplicity.

How can we apply a top-k selection query to a set of uncertain objects? Since each object appears as a set of instances, we have to rank the instances in the possible worlds semantics. A top-k selection query can be applied to a possible world directly which consists of a set of instances. In a possible word, a top-k selection query returns k instances. Now, the problem is how to integrate in a meaningful way the results from all possible worlds.

Often, when a user raises a top-k selection query on uncertain objects, the user is interested in the objects which have a high probability to be ranked top-k. Probabilistic threshold top-k queries [30, 31] capture this intuition.

Definition 6 (Probabilistic threshold top-k queries) Let Q^k be a top-k selection query and $Q^k(w)$ (called the **top-k list**) be the top-k instances in a possible world $w \in \mathcal{W}$. Then, the **top-k probability** of an uncertain object O is

$$\Pr^k(O) = \sum_{w \in \mathcal{W},\ O \text{ has an instance in } Q^k(w)} \Pr(w).$$

Given a **probability threshold** p $(0 < p \leq 1)$, the **probabilistic threshold top-k query** Q^k_p returns the complete set of uncertain objects $\{O | \Pr^k(O) \geq p\}$.

Probabilistic threshold top-k queries can be applied on a sliding window of multiple uncertain data streams. We treat the instances of an uncertain data stream falling into the current sliding window as an uncertain object, and rank the streams according to their current sliding window.

Problem definition. Given a probabilistic threshold top-k query Q^k_p, a set of uncertain data streams $\mathcal{O}$, and a sliding window width ω, the **continuous probabilistic**

threshold top-k query is to, for each time instant t, report the set of uncertain data streams whose top-k probabilities in the sliding window $W^t_\omega(\mathcal{O})$ are at least p.

Hereafter, for the sake of simplicity we call a probabilistic threshold top-k query a top-k query when it is clear from context.

Example 3 (Continuous probabilistic threshold top-k queries) Consider the uncertain streams in Table 3 with sliding window size $\omega = 3$ and continuous probabilistic threshold top-2 query with threshold $p = 0.5$.

At time instant t, the sliding window contains uncertain objects $W^t_3(A)$, $W^t_3(B)$, $W^t_3(C)$ and $W^t_3(D)$. The top-k probabilities of those uncertain objects are: $Pr^2(W^t_3(A)) = 1$, $Pr^2(W^t_3(B)) = \frac{2}{27}$, $Pr^2(W^t_3(C)) = \frac{5}{9}$ and $Pr^2(W^t_3(D)) = \frac{10}{27}$. Therefore, the probabilistic threshold top-k query returns $\{A, C\}$ at time instant t.

At time instant $t + 1$, the top-k probabilities of the uncertain objects are: $Pr^2(W^{t+1}_3(A)) = 1$, $Pr^2(W^{t+1}_3(B)) = \frac{2}{27}$, $Pr^2(W^{t+1}_3(C)) = \frac{4}{9}$ and $Pr^2(W^{t+1}_3(D)) = \frac{13}{27}$. The probabilistic threshold top-k query returns $\{A\}$ at time instant $t + 1$.

The methods of answering probabilistic threshold top-k queries will be discussed in Sects. 3, 4, and 5.

3 Exact algorithms

In this section, we discuss deterministic algorithms to give exact answers to probabilistic threshold top-k queries. First, we extend a technique in [30, 31] using Poisson binomial recurrence in answering a query in one sliding window. Then, we discuss how to share computation among overlapping sliding windows.

3.1 Top-k probabilities in a sliding window

Consider a set of uncertain data streams $\mathcal{O} = \{O_1, \ldots, O_n\}$ and sliding window $W^t_\omega(\mathcal{O})$. $W^t_\omega(O_i) = \{O_i[t-\omega+1], \ldots, O_i[t]\}$ is the set of instances of O_i $(1 \le i \le n)$ in the sliding window. In this subsection, we consider how to rank the data streams according to their instances in sliding window $W^t_\omega(\mathcal{O})$. When it is clear from context, we write $W^t_\omega(\mathcal{O})$ and $W^t_\omega(O_i)$ simply as $W(\mathcal{O})$ and $W(O_i)$, respectively.

We reduce computing the top-k probability of a stream O into computing the top-k probabilities of instances of O.

Definition 7 (Top-k probability of instance) For an instance o and a top-k query Q^k, the *top-k probability* of o, denoted by $Pr^k(o)$, is the probability that o is ranked in the top-k lists in possible worlds. That is, $Pr^k(o) = \frac{\|\{w \in \mathcal{W} | o \in Q^k(w)\}\|}{\|\mathcal{W}\|}$.

Following with Definitions 6 and 7, we have the following.

Corollary 2 (Top-k probability) *For an uncertain data stream O, a sliding window $W^t_\omega(\mathcal{O})$ and a top-k query Q^k,*

$$Pr^k(O) = \sum_{o \in W^t_\omega(O)} Pr^k(o)Pr(o) = \frac{1}{\omega} \sum_{o \in W^t_\omega(O)} Pr^k(o).$$

We sort all instances according to their scores. Let R denote the ranking order of instances. For two instances o_1, o_2, we write $o_1 \prec o_2$ if o_1 is ranked before (i.e., better than) o_2 in R. Clearly, the rank of an instance o of stream O in the possible worlds depends on only the instances of other streams that are ranked better than o. We capture those instances as the dominant set of o.

Definition 8 (Dominant set) Given a set of streams $\mathcal{O}$, a sliding window W, and a top-k query Q^k, for an instance o of stream $O \in \mathcal{O}$, the **dominant set** of o is the set of instances of streams in $\mathcal{O} - \{O\}$ that are ranked better than o, denoted by $DS(o) = \{o' \in W(\mathcal{O} - O) | o' \prec o\}$.

In a possible world w, an instance o is ranked the i-th place if and only if there are $(i-1)$ instances in $DS(o)$ appearing in w, and each of those instances is from a unique stream.

Based on this observation, for instance o and stream O' such that $o \notin O'$, we denote by $O' \prec o$ in a possible world w if there exists $o' \in W(O')$, $o' \prec o$, and o' and o appear in w. Apparently, we have

$$\Pr(O' \prec o) = \sum_{o' \in DS(o), o' \in O'} \Pr(o')\Pr(o) = \frac{1}{\omega^2} \|DS(o) \cap W(O')\|.$$

Let $\Pr(DS(o), i)$ be the probability that i instances in $DS(o)$ from unique streams appear in a possible world. Then, the top-k probability of o can be written as

$$\Pr^k(o) = \Pr(o) \sum_{i=0}^{k-1} \Pr(DS(o), i) = \frac{1}{\omega} \sum_{i=0}^{k-1} \Pr(DS(o), i).$$

For an instance o, since the events $O' \prec o$ for $O' \in \mathcal{O} - \{O\}$ are independent, we can view $DS(o)$ as a set of independent random binary trials, where each trial $X_{O'}$ is corresponding to an uncertain object O', $\Pr(X_{O'} = 1) = \Pr(O' \prec o)$, and $\Pr(X_{O'} = 1) = 1 - \Pr(X_{O'} = 1)$. The event that a trial takes value 1 is called a success. Since the probability that each trial takes value 1 is not identical, the total number of successes in $DS(o)$ follows the Poisson binomial distribution [38]. Thus, $\Pr(DS(o), i)$ can be computed using the Poisson binomial recurrence as follows.

Theorem 1 (Poisson binomial recurrence [38]) *Let $p_1, \ldots, p_n$ be the probabilities of independent events $X_1, \ldots, X_n$ and $S_i = \{X_1, \ldots, X_i\}$ $(1 \le i \le n)$. $\Pr(S_i, k)$ $(k \ge 0)$ denotes the probability that k events in S_i happen. Then,* $\Pr(S_1, 0) = 1 - p_1$ *and* $\Pr(S_1, 1) = p_1$. *Moreover, for i, j $(1 \le i < j \le n)$,*

$$\Pr(S_j, 0) = (1 - p_j)\Pr(S_{j-1}, 0);$$
$$\Pr(S_j, i) = p_j \Pr(S_{j-1}, i-1) + (1 - p_j)\Pr(S_{j-1}, i);$$
$$\Pr(S_j, j) = p_j \Pr(S_{j-1}, j-1).$$

The cost of sorting all instances in a sliding window is $O(n\omega \log(n\omega))$. To compute the top-k probability of each instance, the Poisson binomial recurrence is run and takes cost $O(kn)$ in time. Since there are $n\omega$ instances in the sliding window, the overall time complexity is $O(kn^2\omega + n\omega\log(n\omega))$.

Example 4 (Poisson binomial recurrence) Table 3 shows 4 uncertain streams A, B, C, and D. For each instance, a ranking score is given. The ranking order is the ranking score descending order: the larger the ranking score, the better the instance is ranked.

Let us consider the sliding window W_3^t (i.e., the first three columns of instances in the figure), and compute the top-2 probability of c_2. The dominant set is $DS(c_2) = \{a_1, a_2, a_3, d_3\}$. Thus, $p_1 = \Pr(A \prec c_2) = \Pr(a_1) + \Pr(a_2) + \Pr(a_3) = 1$, $p_2 = \Pr(B \prec c_2) = 0$, and $p_3 = \Pr(D \prec c_2) = \Pr(d_3) = \frac{1}{3}$.

Using Theorem 1, let $S_1 = \{A\}$, $S_2 = \{A, B\}$ and $S_3 = \{A, B, D\}$. For S_1, we have $\Pr(S_1, 0) = 1 - p_1 = 0$ and $\Pr(S_1, 1) = p_1 = 1$.

For S_2, we have $\Pr(S_2, 0) = (1 - p_2)\Pr(S_1, 0) = 0$ and $\Pr(S_2, 1) = p_2\Pr(S_1, 0) + (1 - p_2)\Pr(S_1, 1) = 1$.

For S_3, we have $\Pr(S_3, 0) = (1 - p_3)\Pr(S_2, 0) = 0$ and $\Pr(S_3, 1) = p_3\Pr(S_2, 0) + (1 - p_3)\Pr(S_2, 1) = \frac{2}{3}$.

Thus, $\Pr^2(c_2) = \Pr(c_2)(\Pr(S_3, 0) + \Pr(S_3, 1)) = \frac{2}{9}$.

If we sort all the instances in sliding window $W(\mathcal{O})$ in the ranking order, then by one scan of the sliding window we can calculate the top-k probabilities for all instance. For each stream O, we only need to keep the following two pieces of information during the scan. First, we keep the number of instances in O that have been scanned. Suppose there are l such instances, then the probability of O in the Poisson recurrence is $\frac{l}{\omega}$. Second, we maintain the sum of the top-k probabilities of those scanned instances of O.

In practice, when a top-k query is raised, $k \ll n$ often holds where n is the total number of streams. In such a case, some streams can be pruned in the computation.

Theorem 2 (Pruning instances in a stream) *For an uncertain stream O, a top-k query Q_p^k with probability threshold p, and all instances in $W(O)$ sorted in the ranking order $o_1 \prec \cdots \prec o_\omega$, if there exists i $(1 \le i \le \omega)$ such that*

$$\Pr^k(o_i) < \frac{p - \sum_{j=1}^{i-1} \Pr^k(o_j)}{\omega - i + 1} \tag{1}$$

then $\Pr^k(O) < p$.

Moreover, $\Pr^k(O) \ge p$ *if there exists i $(1 \le i \le \omega)$ such that $\sum_{j=1}^{i-1} \Pr^k(o_j) \ge p$.*

Proof The first part: If there exists i such that $\Pr^k(o_i) < \frac{p - \sum_{j=1}^{i-1} \Pr^k(o_j)}{\omega - i + 1}$, then

$$(\omega - i + 1)\Pr^k(o_i) < p - \sum_{j=1}^{i-1} \Pr^k(o_j).$$

Apparently, $\Pr^k(o_1) \geq \cdots \geq \Pr^k(o_\omega)$. Thus, we have

$$(\omega - i + 1)\Pr^k(o_i) \geq \sum_{j=i}^{\omega} \Pr^k(o_j).$$

Combining the above two inequalities, we have

$$\sum_{j=i}^{\omega} \Pr^k(o_j) < p - \sum_{j=1}^{i-1} \Pr^k(o_j).$$

Thus, $\Pr^k(O) = \sum_{j=1}^{i-1} \Pr^k(o_j) + \sum_{j=i}^{\omega} \Pr^k(o_j) < p$.

To prove the second part, we only need to notice $\Pr^k(O) = \sum_{j=1}^{\omega} \Pr^k(o_j) \geq \sum_{j=1}^{i} \Pr^k(o_j) \geq p$. □

To use Theorem 2, for each stream O, if the last scanned instance in O satisfies one of the conditions in the theorem, the top-k probabilities of the remaining instances of O do not need to be computed.

For an object uncertain stream O whose top-k probability in sliding window W is smaller than the threshold p, we can derive the maximum number of instances scanned according to Theorem 2 as follows.

Corollary 3 (Maximum number of scanned instances) *For an uncertain stream O, a top-k query Q_p^k with probability threshold p, and all instances in $W(O)$ sorted in the ranking order $o_1 \prec \cdots \prec o_\omega$, the maximum number of instances scanned according to Theorem 2 is $\lceil \frac{\Pr^k(O)}{p} \omega \rceil + 1$.*

Proof Let o_t $(t > 1)$ be the first instance in $W(O)$ that satisfy Inequality 1. Then, o_i $(1 \leq i < t)$ does not satisfy Inequality 1. That is,

$$\Pr^k(o_1) \geq \frac{p}{\omega}, \quad \text{and} \quad \Pr^k(o_i) \geq \frac{p - \sum_{j=1}^{i-1} \Pr^k(o_j)}{\omega - i + 1} \quad \text{for } 1 < i < t.$$

Using mathematical induction, it is easy to show that for $1 \leq i < t$

$$\sum_{j=1}^{i} \Pr^k(o_j) \geq i \times \frac{p}{\omega}.$$

Since $\Pr^k(O) = \sum_{j=1}^{t-1} \Pr^k(o_j) + \sum_{m=t}^{\omega} \Pr^k(o_m)$, we have

$$\sum_{j=1}^{t-1} \Pr^k(o_j) = \Pr^k(O) - \sum_{m=t}^{\omega} \Pr^k(o_m) \geq (t-1) \times \frac{p}{\omega}.$$

Therefore, $t \leq \lceil \frac{\Pr^k(O)}{p} \omega \rceil + 1$. □

Our second pruning rule is based on the following observation.

Lemma 1 (Sum of top-k probabilities) *For a set of uncertain data streams $\mathcal{O}$, a top-k query Q^k, and a sliding window $W^t_\omega(\mathcal{O})$, $\sum_{O \in \mathcal{O}} \Pr^k(O) = k$.*

Proof Using the definitions of top-k probabilities, we have

$$\sum_{O \in \mathcal{O}} \Pr^k(O) = \sum_{w \in \mathcal{W}, o \in Q^k(w)} \Pr(w).$$

In a possible world w, $\sum_{o \in Q^k(w)} \Pr(w) = k \cdot \Pr(w)$. Using Corollary 1, we have $\sum_{O \in \mathcal{O}} \Pr^k(O) = k \sum_{w \in \mathcal{W}} \Pr(w) = k$. □

Theorem 3 (Pruning by top-k probability sum) *Consider a set of uncertain data streams $\mathcal{O}$, a top-k query Q^k_p with probability threshold p, and a sliding window $W^t_\omega(\mathcal{O})$. Assume all instances in $W^t_\omega(\mathcal{O})$ are scanned in the ranking order, and $S \subset W^t_\omega(\mathcal{O})$ is the set of instances that are scanned. For a stream $O \in \mathcal{O}$, $\Pr^k(O) < p$ if*

$$\sum_{o \in O \cap S} \Pr^k(o) < p - \left(k - \sum_{o' \in S} \Pr^k(o')\right).$$

Proof Following with Lemma 1, we have

$$\sum_{o \in W^t_\omega(\mathcal{O}) - S} \Pr^k(o) = k - \sum_{o' \in S} \Pr^k(o').$$

If $\sum_{o \in O \cap S} \Pr^k(o) < p - (k - \sum_{o' \in S} \Pr^k(o'))$, then

$$\begin{aligned}
\Pr^k(O) &= \sum_{o \in O} \Pr^k(o) \\
&= \sum_{o \in O \cap S} \Pr^k(o) + \sum_{o \in O \cap (W^t_\omega(\mathcal{O}) - S)} \Pr^k(o) \\
&< k - \sum_{o' \in S} \Pr^k(o') + p - \left(k - \sum_{o' \in S} \Pr^k(o')\right) \\
&= p.
\end{aligned}$$
□

In summary, by sorting the instances in a sliding window in the ranking order and scanning the sorted list once, we can compute the top-k probability for each stream, and thus the exact answer to the top-k query on the window can be derived. The two pruning rules can be used to prune the instances and the streams.

3.2 Sharing between sliding windows

Using the method described in Sect. 3.1, we can compute the exact answer to a top-k query Q^k in one sliding window W^t. In the next time instant $(t + 1)$, can we reuse some of the results in window W^t to compute the answer to Q^k in window W^{t+1}?

In this subsection, we first observe the compatible dominant set property, and then we explore sharing in computing answers to a top-k query on two consecutive sliding windows.

3.2.1 Compatible dominant sets

For an instance $o \in O$ that is in a window W^t, the top-k probability of o depends on only the number of instances from streams other than O that precede o in the ranking order. The ordering among those instances does not matter. Therefore, for an instance $o \in W^{t+1}$, if we can identify an instance o' in either W^t or W^{t+1} such that o and o' are compatible in terms of number of other preceding instances, then we can derive the top-k probability of o using that of o' directly. Technically, we introduce the concept of compatible dominant sets.

Definition 9 (Compatible dominant sets) Let $o \in O$ be an instance that is in window W^{t+1} and $DS^{t+1}(o)$ be the dominant set of o in W^{t+1}. For an instance $o_1 \in O$ and dominant set $DS(o_1)$, if for any stream $O' \neq O$, the number of instances from O' in $DS^{t+1}(o)$ and that in $DS(o)$ are the same, $DS^{t+1}(o)$ and $DS(o_1)$ are called **compatible dominant sets**. Please note that o may be the same instance as o_1, and $DS(o_1)$ can be in W^t or W^{t+1}. We consider $DS^{t+1}(o)$ and itself trivial compatible dominant sets.

Following with the Poisson binomial recurrence (Theorem 1), we immediately have the following result.

Theorem 4 (Compatible dominant sets) *If $DS^{t+1}(o)$ and $DS(o_1)$ are compatible dominant sets, for any $j \geq 0$, $\Pr(DS^{t+1}(o), j) = \Pr(DS(o_1), j)$ and $\Pr^k(o) = \Pr^k(o_1)$.*

Proof If $DS^{t+1}(o)$ and $DS(o_1)$ are compatible dominant sets, then for any stream O' ($o, o_1 \notin O'$), $\Pr[O' \in DS^{t+1}(o)] = \Pr[O' \in DS(o_1)]$. Thus, following with Theorem 1, we have $\sum_{i=0}^{k-1} \Pr(DS^{t+1}(o), i) = \sum_{j=0}^{k-1} \Pr(DS(o_1), j)$. Therefore, $\Pr^k(o) = \Pr^k(o_1)$. □

Compatible dominant sets can be employed directly to reduce the computation in window W^{t+1} using the results in window W^t and those already computed in window W^{t+1}. For any instance o, if the dominant set of o in W^{t+1} is compatible to some dominant set of o_1, then the top-k probability of o in W^{t+1} is the same as o_1. No recurrence computation is needed for o in W^{t+1}.

When the data streams evolve slowly, the instances from a stream may have a good chance to be ranked in the compatible places. Using compatible dominant sets can capture such instances and save computation.

Now, the problem becomes how to find compatible dominant sets quickly. Here, we give a fast algorithm which can be integrated to the top-k probability computation.

For each sliding window $W^t(\mathcal{O})$, we maintain the sorted list of instances in the window. When the window slides, we update the sorted list in two steps. First, we insert the new instances into the sorted list, but still keep the expired instances. We call the sorted list after the insertions the *expanded sorted list*.

We use an n-digit bitmap counter $c[1], \ldots, c[n]$, where n is the number of streams. At the beginning, $c[i] = 0$ for $1 \leq i \leq n$. We scan the expanded sorted list in the ranking order. If an expired instance or a new instance $o \in O_i$ is met, we set $c[i] = c[i] \oplus 1$.

For an instance $o \in O_i$ in the expanded list such that o is in both W^t and W^{t+1}, if all the bitmap counters, except for $c[i]$, are 0 right before o is read, then, for every instance $o' \in O_j$ $(i \neq j)$, $o' \prec o$ in the expanded sorted list, one of the following three cases may happen: (1) o' appears in both W^t and W^{t+1}; (2) $o' = O'_j[t - \omega + 1]$ (i.e., o' appears in W^t only) and the new instance $O_j[t+1] \prec o$; or (3) $o' = O'_j[t+1]$ (i.e., appears in W^{t+1} only) and the expired instance $O'_j[t - \omega + 1] \prec o$. In all the three cases, $DS^t(o)$ and $DS^{t+1}(o)$ are compatible if o does not arrive at time $t+1$.

If o arrives at time $t+1$, then we check the left and the right neighbors of o in the expanded sorted list. If one of them o' is from the same stream as o, then $DS(o)$ and $DS(o')$ are compatible.

We conduct Poisson recurrence for only instances which are in $W^{t+1}(\mathcal{O})$ and do not have a compatible dominance set. Otherwise, they are expired instances or their top-k probabilities can be obtained from the compatible dominant sets immediately. After one scan of the expanded sorted list, we identify all compatible dominant sets and also compute the top-k probabilities. Then, we remove from the expanded sorted list those expired instances. The current sliding window is processed. We are ready to slide the window to the next time instant $(t+2)$.

Example 5 (Compatible dominant set) Figure 1(a) shows the expanded sorted list of instances in sliding windows W_3^t and W_3^{t+1} in Table 3. At time $t+1$, the instances a_1, b_1, c_1, d_1 expire, and new instances a_4, b_4, c_4, d_4 arrive.

In Fig. 1(b), we show the values of the bitmap counters during the scan of the expanded sorted list. Each instance in W_3^{t+1}, except for d_3, can find a compatible dominant set. We only need to conduct the Poisson recurrence computation of d_3 in W_3^{t+1}.

3.2.2 *Pruning using the highest possible rank*

Consider an instance o in a sliding window W^t. As the window slides towards future, new instances arrive and old instances expire. As a result, the rank of o in the sliding windows may go up or down.

However, the instances arriving later than o or at the same time as o would never expire before o. In other words, the possible rank of o in the future sliding windows is bounded by those instances "no older" than o.

Lemma 2 (Highest possible rank) *For an instance $O[i]$ arriving at time i, in a sliding window $W_\omega^t(\mathcal{O})$ such that $t - \omega + 1 < i \leq t$, let $\mathcal{R}_{O[i]} = \{O'[j] \| O' \in \mathcal{O}, O' \neq O, j \geq i\}$. In any sliding window $W_\omega^{t'}$ such that $t' > t$, the rank of $O[i]$ cannot be less than $\|\mathcal{R}_{O[i]}\| + 1$.*

Example 6 (Highest possible rank) Consider again the uncertain streams in Table 3. In window W_3^t, the rank of c_2 is 6. Among the 8 instances with time-stamp $t-1$

Ranked list of the instances at time t and $t+1$
a_2, a_1, c_1, a_3, a_4, d_3, c_4, c_2, d_2, b_1, b_4, b_2, d_1, d_4, c_3, b_3

(a) The expanded sorted list.

Instance	Counter=[A,B,C,D]
a_2	[0, 0, 0, 0]
a_1	[1, 0, 0, 0]
c_1	[1, 0, 1, 0]
a_3	[1, 0, 1, 0]
a_4	[0, 0, 1, 0]
d_3	[0, 0, 1, 0]
c_4	[0, 0, 0, 0]
c_2	[0, 0, 0, 0]
d_2	[0, 0, 0, 0]
b_1	[0, 1, 0, 0]
b_4	[0, 0, 0, 0]
b_2	[0, 0, 0, 0]
d_1	[0, 0, 0, 1]
d_4	[0, 0, 0, 0]
c_3	[0, 0, 0, 0]
b_3	[0, 0, 0, 0]

(b) The bitmap counters.

Fig. 1 The sorted lists of instances in $SW(t-1)$ and $SW(t)$

and t, there are 3 instances ranked better than c_2. The highest possible rank of c_2 in the future windows is 4. In window W^{t+1}, there are 5 instances arriving no earlier than c_2 and ranked better than c_2. The highest possible rank of c_2 in the future windows is 6.

The highest possible rank of o can be used to derive an upper bound of the top-k probability of o in the future sliding windows.

Theorem 5 (Highest possible top-k probability) *For an instance o in sliding window W_ω^t with the highest possible rank $r \geq k\omega$, let $\rho = \frac{r-1}{\omega(n-1)}$, where n is the number of streams, in any window $W_\omega^{t'}$ $(t' \geq t)$,*

$$\Pr^k(o) \leq \frac{1}{\omega} \sum_{j=0}^{k-1} \binom{n}{j} \rho^j (1-\rho)^{n-j}.$$

Proof To prove the theorem, we need the following lemma [29].

Lemma 3 (Extrema of Poisson trials) *Let $p_1, \ldots, p_n$ be the success probabilities of n independent Poisson trials $X_1, \ldots, X_n$, respectively. Let $\rho = \frac{1}{n}\sum_{i=1}^{n} p_i$, and*

$X=\sum_{i=1}^{n} X_i$. *If* $0 \le c \le n \times \rho - 1$, *then*

$$\Pr(X \le c) \le \sum_{j=0}^{c} \binom{n}{j} \rho^j (1-\rho)^{n-j}.$$

Given o with the highest possible rank r, there are $r-1$ instances from other objects ranked better than o. The sum of membership probabilities of those instances is $\frac{r-1}{\omega}$. The theorem follows with Lemma 3 directly. □

Corollary 4 (Pruning using highest possible rank) *For any instance* $o \in O$, *if* $\sum_{o \in O} p_o < p$ *and there exists* p_o *such that* $\Pr^k(o) \le p_o$, *then* $\Pr^k(O) < p$.

We need $O(1)$ space to maintain the highest possible rank for an instance. The overall space consumption is $O(n\omega)$ for a sliding window. Each time when new instances arrive, the highest possible ranks of all old instances are updated. The highest possible top-k probability of each stream is updated accordingly. This can be integrated into the top-k probability computation. For a stream O, once the upper bound of $\Pr^k(O)$ fails the threshold, all instances in O do not need to be checked in the current window.

The complete exact algorithm is shown in Fig. 2. Compatible dominant sets can help to reduce the computation cost, however, although it works well in practice, in the worst case, the new instances may be ranked far away from the expired instances of the same stream, and thus no compatible dominant sets can be found. Thus, the time complexity of processing a sliding window, except for the first one, is $O(kn^2\omega + n\log(n\omega))$, where $O(n\log(n\omega))$ is the cost to insert the n new instances into the sorted list.

4 A sampling method

In this section, we propose a sampling method to estimate the top-k probability of each stream with a probabilistic quality guarantee.

For a stream O in a sliding window $W^t(\mathcal{O})$, we are interested in the event that O is ranked top-k. Let Z_O be the indicator to the event: $Z_O = 1$ if O is ranked top-k in $W^t(\mathcal{O})$; $Z_O = 0$ otherwise. Then, $\Pr(Z_O = 1) = \Pr^k(O)$.

To approximate the probability $\Pr(Z_O = 1)$, we design a statistic experiment as follows. We draw samples of possible worlds and compute the top-k lists on the samples. That is, in a sample, for each stream O we select an instance o in window $W^t(O)$. A sample is a possible world. Then, we sort all instances in the sample in the ranking order and find the top-k list.

We repeat the experiment m times independently. Let $Z_{O,i}$ be the value of Z_O at the i-th run. Then, $\widetilde{E}[Z_O] = \frac{1}{m}\sum_{i=1}^{m} Z_{O,i}$ is an estimation of $E[Z_O] = \Pr(Z_O = 1)$.

By using a sufficiently large number of samples, we can obtain a good approximation of $\Pr^k(Z_O)$ with high probability. The methods follows the idea of unrestricted random sampling (also known as simple random sampling with replacement) [36]. The following minimum sample size can be derived from the well known Chernoff-Hoffding bound [2].

```
Input: a set of uncertain streams $\mathcal{O}$, a sliding window width $\omega$,
    a time instant $t+1$, a top-$k$ query $Q_p^k$, and a probability
    threshold $p$;
Output: Answer to top-$k$ query in window $W_\omega^{t+1}$;
Method:
1:  insert the new instances arriving at time $t+1$ into the
    sorted list in window $W_\omega^t$;
2:  initialize $Pr^k(O_i) = 0$ for each stream $O_i \in \mathcal{O}$;
3:  compute the highest possible top-$k$ probability for each
    stream $O_i \in \mathcal{O}$, remove $O_i$ from consideration if it fails the
    probability threshold;
4:  set counter $C[i] = 0$ for each stream $O_i \in \mathcal{O}$;
5:  scan the expanded sorted list in the ranking order
    for each $o$ do
6:      update the corresponding counter if $o$ is expired or new;
7:      compute $DS(o)$;
        if $DS(o)$ has a compatible dominant set in
8:      then obtain $Pr^k(o)$ directly;
9:      else compute the probabilities $Pr(DS(I), j)$ $(0 \le j < k)$;
10:     $Pr^k(O) = Pr^k(O) + Pr^k(o)$;
11:     if $Pr^k(O) \ge p$ then output $O$;
12:     check whether $o$ can be used to prune some unread
        instances;
13:     if all objects either are output or fail the probability
        threshold then exit;
    end for
```

Fig. 2 The exact algorithm

Theorem 6 (Minimum sample size) *For any stream O, let $Z_{O,1}, \dots, Z_{O,m}$ be the values of Z_O in m independent experiments, and $\widetilde{E}[Z_O] = \frac{1}{m}\sum_{i=1}^{m} Z_{O,i}$. For any δ* $(0 < \delta < 1)$, ξ $(\xi > 0)$, *if $m \ge \frac{3\ln\frac{2}{\delta}}{\xi^2}$, then* $\Pr\{|\widetilde{E}[Z_O] - E[Z_O]| > \xi\} \le \delta$.

For efficient implementation, we maintain an indicator variable for each stream. To avoid sorting instances repeatedly, we first sort all instances in a sliding window $W^t(\mathcal{O})$. When drawing a sample, we scan the sorted list from the beginning, and select an instance $o \in O$ in probability $\frac{1}{\omega}$ if stream O has no instance in a sample yet. If an instance is chosen, the corresponding stream indicator is set. When the sample already contains k instances, the scan stops since the instances sampled later cannot be in the top-k list. The sample can then be discarded since it will not be used later.

The space complexity of the sampling method is $O(n\omega)$, because all instances in the sliding window have to be stored. The time complexity is $O(mn\omega + n\omega\log(n\omega))$ for the first window and $O(mn\omega + n\log(n\omega))$ for other windows where m is the number of samples drawn, since the n new instances can be inserted into the sorted list in $W^t(\mathcal{O})$ to form the sorted list in $W^{t+1}(\mathcal{O})$.

5 Space efficient methods

In the exact algorithm and the sampling algorithm, we need to store the sliding window in main memory. In some applications, there can be a large number of streams and the window width is non-trivial. In this section, we develop the space efficient methods using approximate quantile summaries. The central idea is to use quantiles to summarize instances in streams. Since computing exact quantiles in data streams is costly, we seek for high quality approximation.

Both the exact algorithm in Sect. 3 and the sampling method in Sect. 4 can be applied on the approximate quantile summaries of uncertain data streams. Using quantiles is a trade-off between space requirement and query answering accuracy. The distribution of an object is represented in a higher granularity level using quantiles, and thus the query results are approximate. However, we show that using approximate quantiles can save substantial space in answering top-k queries on uncertain streams with high quality guarantees.

5.1 Top-k probabilities and quantiles

Definition 10 (Quantile) Let $o_1 \prec \cdots \prec o_\omega$ be the sorted list of instances in the ranking order in a sliding window $W^t_\omega(O)$. The **ϕ-quantile** $(0 < \phi \leq 1)$ of $W^t_\omega(O)$ is instance $o_{\lceil \phi\omega \rceil}$. A **$\phi$-quantile summary** of $W^t_\omega(O)$ is o_1 and a list of instances $o_{i\lceil \phi\omega \rceil}$ $(1 \leq i \leq \lceil \frac{1}{\phi} \rceil)$.

The ϕ-quantile summary of $W^t_\omega(O)$ partitions the instances of $W^t_\omega(O)$ into $\lceil \frac{1}{\phi} \rceil$ **intervals** (in the values of the ranking function), with $\lceil \phi\omega \rceil$ instances in each interval. The first interval $t_1 = [o_1, o_{\lceil \phi\omega \rceil}]$. Generally, the i-th $(1 < i \leq \lceil \frac{1}{\phi} \rceil)$ interval $t_i = (o_{\lceil (i-1)\phi\omega \rceil}, o_{\lceil i\phi\omega \rceil}]$. Since the membership probability of each instance is $\frac{1}{\omega}$, the **membership probability** of each interval t_i is $\Pr(t_i) = \frac{1}{\omega}\phi\omega = \phi$.

Example 7 (A quantile summary) Consider a window $W^t_9(O)$ where the sorted list of instance scores is $(21, 20, 12, 10, 9, 5, 4, 3, 2)$. Then, 12, 5 and 2 are the $\frac{1}{3}$, $\frac{2}{3}$, and 1-quantiles of $W^t_9(O)$, respectively. The $\frac{1}{3}$-quantile summary of O is $(21, 12, 5, 2)$ which partitions the instances into three intervals: $t_1 = [21, 12]$, $t_2 = (12, 5]$, and $t_3 = (5, 2]$. The membership probability of each interval is $\frac{1}{3}$.

We can use quantiles to approximate the top-k probabilities of streams.

Example 8 (Approximating top-k probability) Consider three streams A, B, and C and their quantile summaries in window $W(\{A, B, C\})$, as shown in Fig. 3, where a_i, b_i and c_i $(1 \leq i \leq 3)$ are the intervals of $W(A)$, $W(B)$ and $W(C)$, respectively. The membership probability of each interval is $\frac{1}{3}$.

To compute the upper bound of the top-2 probability of instances falling into b_1, we let all instances in b_1 take the maximum value $b_1.MAX$. Moreover, since intervals a_2 and c_2 cover $b_1.MAX$, we let all instances in a_2 and c_2 take the minimum value $a_2.MIN$ and $c_2.MIN$, respectively. Thus, the probability that A is ranked better than b_1 is at least $\Pr(a_1) = \frac{1}{3}$, and the probability that C is ranked better than b_1

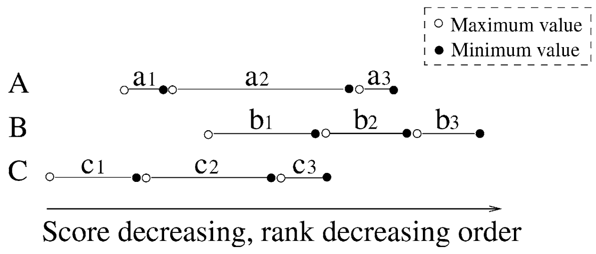


Fig. 3 The quantile summaries of three streams

is at least $\Pr(c_1) = \frac{1}{3}$. The upper bound of the top-2 probability of b_1 is $\Pr(b_1) \times (1 - \frac{1}{3} \times \frac{1}{3}) = \frac{8}{27}$.

Using the similar idea, we can verify that $\frac{1}{9}$ is the lower bound of the top-2 probability of b_1.

Using the idea illustrated in Example 8 and the Poisson binomial recurrence, we can have the general upper and lower bounds of the top-k probabilities of intervals.

Theorem 7 (Upper and lower bounds) *To answer a top-k query on sliding window $W(\mathcal{O})$, for an interval t in a ϕ-quantile summary of $W(O)$ $(O \in \mathcal{O})$,*

$$\Pr^k(t) \le \Pr(t) \sum_{i=0}^{k-1} \Pr(UDS(t), i)$$

and

$$\Pr^k(t) \ge \Pr(t) \sum_{i=0}^{k-1} \Pr(LDS(t), i),$$

where $\Pr^k(t) = \sum_{o \in W(O), o \in t} \Pr^k(o)$, $UDS(t) = \{o' | o' \in t', t'$ is an interval in a ϕ-quantile summary of $W(O')$, $O' \neq O, t'.MIN \ge t.MAX\}$, and $LDS(t) = \{o' | o' \in t', t'$ is an interval in a ϕ-quantile summary of $W(O')$, $O' \neq O, t'.MAX \ge t.MIN\}$.

Proof Since $UDS(t) \subseteq DS(t)$, $\sum_{i=1}^{k-1} \Pr(UDS(t), i) \ge \sum_{j=1}^{k-1} \Pr(DS(t), j)$. Similarly, $DS(t) \subseteq LDS(t)$, so we have $\sum_{j=1}^{k-1} \Pr(DS(t), j) \ge \sum_{i=1}^{k-1} \Pr(LDS(t), i)$. Moreover, since $\Pr^k(t) = \Pr(t) \sum_{j=1}^{k-1} \Pr(DS(t), j)$, the conclusions hold. □

Using Theorem 7, we can easily derive the upper bound and the lower bound of a stream by summing up the upper/lower bounds of all intervals of the quantile summary of the stream. Importantly and interestingly, the difference between the upper bound and the lower bound of the top-k probability is up to 2ϕ, which provides a strong quality guarantee in approximation.

Theorem 8 (Approximation quality) *For a stream O, let $\mathcal{U}(\Pr^k(O))$ and $\mathcal{L}(\Pr^k(O))$ be the upper bound and the lower bound of $\Pr^k(O)$ derived from Theorem 7, respectively. $\mathcal{U}(\Pr^k(O)) - \mathcal{L}(\Pr^k(O)) \le 2\phi$.*

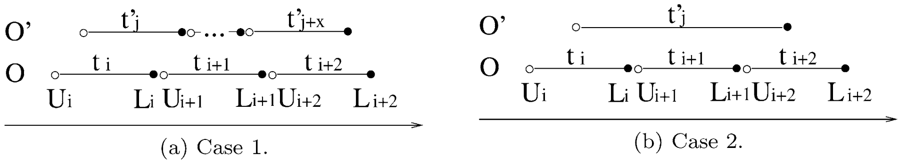


Fig. 4 Two cases in the proof of Theorem 8

Proof To prove Theorem 8, we need the following lemma.

Lemma 4 (Monotonicity) *Let t_i, t_{i+1} and t_{i+2} be three consecutive intervals in the ϕ-quantile summary of $W(O)$. Let $\mathcal{L}(\mathrm{Pr}^k(t_i))$ and $\mathcal{U}(\mathrm{Pr}^k(t_{i+2}))$ be the lower bound of $\mathrm{Pr}^k(t_i)$ and the upper bounds of $\mathrm{Pr}^k(t_{i+2})$, respectively, derived from Theorem 7. If for any interval t' in the ϕ-quantile summary of O' $(O' \neq O)$, $t'.MAX > t_i.MIN$, $t'.MIN > t_{i+2}.MAX$, then $\mathcal{L}(\mathrm{Pr}^k(t_i)) \geq \mathcal{U}(\mathrm{Pr}^k(t_{i+2}))$.*

Proof For intervals t_i and t_{i+2}, we have $LDS(t_i) = \{t' | t' \in O', t'.MAX > t_i.MIN\}$, and $UDS(t_{i+2}) = \{t' | t' \in O', t'.MIN \geq t_{i+2}.MAX\}$. Using the assumption in the lemma, we have $LDS(t_i) \subset UDS(t_{i+2})$. Thus, $\mathcal{L}(\mathrm{Pr}^k(t_i)) \geq \mathcal{U}(\mathrm{Pr}^k(t_{i+2}))$. □

We consider the following two cases:

Case 1. For any interval $t \in W(O)$ and $t' \in W(O')$ $(O' \neq O)$, if $t'.MAX > t.MAX$, then $t'.MIN > t.MIN$, as illustrated in Fig. 4(a). Lemma 4 holds in this case.

According to the definitions of $\mathcal{U}(\mathrm{Pr}^k(O))$ and $\mathcal{L}(\mathrm{Pr}^k(O))$, $\mathcal{U}(\mathrm{Pr}^k(O)) = \sum_{i=1}^{\frac{1}{\phi}} \mathcal{U}(\mathrm{Pr}^k(t_i))$ and $\mathcal{L}(\mathrm{Pr}^k(O)) = \sum_{i=1}^{\frac{1}{\phi}} \mathcal{L}(\mathrm{Pr}^k(t_i))$. Thus,

$$\begin{aligned}
&\mathcal{U}(\mathrm{Pr}^k(O)) - \mathcal{L}(\mathrm{Pr}^k(O)) \\
&\quad = \sum_{i=1}^{\frac{1}{\phi}-2} (\mathcal{U}(\mathrm{Pr}^k(t_{i+2})) - \mathcal{L}(\mathrm{Pr}^k(t_i))) + \mathcal{U}(\mathrm{Pr}^k(t_1)) \\
&\qquad + \mathcal{U}(\mathrm{Pr}^k(t_2)) - \mathcal{L}(\mathrm{Pr}^k(t_{\frac{1}{\phi}-1})) - \mathcal{L}(\mathrm{Pr}^k(t_{\frac{1}{\phi}})).
\end{aligned} \tag{2}$$

Using Lemma 4, we have

$$\sum_{i=1}^{\frac{1}{\phi}-2} (\mathcal{U}(\mathrm{Pr}^k(t_{i+2})) - \mathcal{L}(\mathrm{Pr}^k(t_i))) < 0.$$

Thus, we have

$$\begin{aligned}
&\mathcal{U}(\mathrm{Pr}^k(O)) - \mathcal{L}(\mathrm{Pr}^k(O)) \\
&\quad < \mathcal{U}(\mathrm{Pr}^k(t_1)) + \mathcal{U}(\mathrm{Pr}^k(t_2)) - \mathcal{L}(\mathrm{Pr}^k(t_{\frac{1}{\phi}-1})) - \mathcal{L}(\mathrm{Pr}^k(t_{\frac{1}{\phi}})).
\end{aligned} \tag{3}$$

Also, for $1 \leq i \leq \frac{1}{\phi}$, $\Pr(t_i) = \phi$ and

$$0 \leq \mathcal{U}(\Pr^k(t_i)) \leq \Pr(t_i) \quad \text{and} \quad 0 \leq \mathcal{L}(\Pr^k(t_i)) \leq \Pr(t_i).$$

Thus, we have

$$\mathcal{U}(\Pr^k(t_1)) + \mathcal{U}(\Pr^k(t_2)) - \mathcal{L}(\Pr^k(t_{\frac{1}{\phi}-1})) - \mathcal{L}(\Pr^k(t_{\frac{1}{\phi}})) \leq 2\phi. \tag{4}$$

Plugging Inequalities 3 and 4 into (2), we get $\mathcal{U}(\Pr^k(O)) - \mathcal{L}(\Pr^k(O)) \leq 2\phi$.

Case 2. If case 1 does not hold, i.e., there is an interval $t \in O$ and an interval $t' \in W(O) - O$ such that, $t'.MAX > t.MAX$ and $t'.MIN \leq t.MIN$. That is, interval t' covers t completely, as illustrated in Fig. 4(b). In that case, $\mathcal{U}(\Pr(O' \prec t_{i+1})) - \mathcal{L}(\Pr(O' \prec t_{i+1})) = \Pr(t_j) = \phi$. Comparing to Case 1 where $\mathcal{U}(\Pr(O' \prec t_{i+1})) - \mathcal{L}(\Pr(O' \prec t_{i+1})) = \Pr(t_j) + \cdots + \Pr(t_{j+x}) = (x-1)\phi$, the difference between the upper bound and the lower bound is smaller. Therefore, in Case 2, $\mathcal{U}(\Pr^k(O)) - \mathcal{L}(\Pr^k(O))$ is even smaller than that in Case 1. The theorem holds. □

For any object O, since $\mathcal{U}(\Pr^k(O)) - \mathcal{L}(\Pr^k(O)) \leq 2\phi$, we can simply use $\frac{\mathcal{U}(\Pr^k(O)) - \mathcal{L}(\Pr^k(O))}{2}$ to approximate $\Pr^k(O)$.

Corollary 5 (Approximation Quality) *For a stream $O \in W(\mathcal{O})$, let $\widetilde{\Pr}^k(O) = \frac{\mathcal{U}(\Pr^k(O)) - \mathcal{L}(\Pr^k(O))}{2}$, then $\|\widetilde{\Pr}^k(O) - \Pr^k(O)\| \leq \phi$.*

5.2 Approximate quantile summaries

Although using quantiles we can approximate top-k probabilities well, computing exact quantiles of streams by a constant number of scans still needs $\Omega(N^{\frac{1}{p}})$ space [47]. To reduce the cost in space, we use ϵ-approximate quantile summary which can still achieve good approximation quality.

Definition 11 (ϵ-approximate quantile) Let $o_1 \prec \cdots \prec o_\omega$ be the sorted list of instances in a sliding window $W(O)$. An **ϵ-approximate ϕ-quantile** $(0 < \phi \leq 1)$ of $W(O)$ is an instance O_l where $l \in [\lceil(\phi - \epsilon)\omega\rceil, \lceil(\phi + \epsilon)\omega\rceil]$.

An **ϵ-approximate ϕ-quantile summary** of $W(O)$ is o_1 and a list of instances $o_{l_1}, \ldots, o_{l_{\lceil 1/\phi \rceil}}$, $l_i \in [\lceil(i\phi - \epsilon)\omega\rceil, \lceil(i\phi + \epsilon)\omega\rceil]$ $(1 \leq i \leq \lceil\frac{1}{\phi}\rceil)$.

The ϵ-approximate ϕ-quantile summary of $W(O)$ partitions the instances of $W(O)$ into $\lceil\frac{1}{\phi}\rceil$ **intervals**. The first interval $t_1 = [o_1, o_{l_1}]$, and generally the i-th $(1 < i \leq \lceil\frac{1}{\phi}\rceil)$ interval $t_i = (q_{i-1}, q_i]$.

The number of instances in each interval is in $[(\phi - 2\epsilon)\omega, (\phi + 2\epsilon)\omega]$. Since the membership probability of each instance is $\frac{1}{\omega}$, the membership probability of each interval is within $[\phi - 2\epsilon, \phi + 2\epsilon]$.

Computing ϵ-Approximate quantiles in data streams is well studied [25, 26, 41, 43]. Both deterministic and randomized methods are proposed. In our implementation, we adopt the method of computing approximate quantile summaries in a sliding window proposed in [41], which is based on the GK-algorithm [26] that finds the approximate quantile over a data steam. The algorithm can continuously output the ϵ-approximate quantiles in a sliding window with space cost of $O(\frac{\log^2 \epsilon\omega}{\epsilon^2})$.

Then, how can we compute the upper bound and the lower bound of the top-k probability of a stream in a sliding window using its ϵ-approximate ϕ-quantile summaries?

Consider an interval $t_i = (o_{i-1}, o_i]$ in an ϵ-approximate ϕ-quantile summary. Suppose $R(o_{i-1})$ and $R(o_i)$ are the actual rank of o_{i-1} and o_i, respectively. Then, the actual number of instances in t_i is $R(o_i) - R(o_{i-1})$. The membership probability of t_i is $\Pr(t_i) = \frac{R(o_i)-R(o_{i-1})}{\omega}$. However, since o_{i-1} and o_i are approximations of the $(i-1)\phi$- and $i\phi$-quantiles, respectively, their actual ranks are not calculated. Instead, we use $\widetilde{\Pr}(t) = \phi$ to approximate the membership probability of t_i. Since $((i-1)\phi - \epsilon)\omega \le R(o_{i-1}) \le ((i-1)\phi + \epsilon)\omega$, and $(i\phi - \epsilon)\omega \le R(o_i) \le (i\phi + \epsilon)\omega$, we have $\|\Pr(t) - \widetilde{\Pr}(t)\| \le 2\epsilon$.

Then, we compute the upper bound and the lower bound of $\Pr^k(t)$, denoted by $\widetilde{\mathcal{U}}(\Pr^k(t))$ and $\widetilde{\mathcal{L}}(\Pr^k(t))$, respectively, using the approximate membership probability $\widetilde{\Pr}(t)$, following with Theorem 7. In sequel, we can further derive the upper bound and the lower bound of a stream by summing up the upper bound and the lower bound of all intervals, respectively. The above approximation method has the following quality guarantee.

Theorem 9 (Approximation quality) *Given a stream O in a sliding window W, let $\widetilde{\mathcal{U}}(\Pr^k(O))$ and $\widetilde{\mathcal{L}}(\Pr^k(O))$ be the upper and lower bounds of* $\Pr^k(O)$ *computed using the ϵ-approximate ϕ-quantile summary of $W(O)$, then,*

$$\|\widetilde{\mathcal{U}}(\Pr^k(O)) - \mathcal{U}(\Pr^k(O))\| \le \epsilon \tag{5}$$

and

$$\|\widetilde{\mathcal{L}}(\Pr^k(O)) - \mathcal{L}(\Pr^k(O))\| \le \epsilon. \tag{6}$$

Proof Consider an ϵ-approximate $i\phi$-quantile $o_i \in O$. To analyze the approximation error introduced by o_i, we first assume that other quantiles o_j $(1 \le j \le \frac{1}{\phi}, j \ne i)$ are exact. Suppose the real rank of o_i is $R(o_i)$, according to the definition of ϵ-approximate quantile, we have $(i\phi - \epsilon)\omega \le R(o_i) \le (i\phi + \epsilon)\omega$.

$t_i = (o_{i-1}, o_i]$ and $t_{i+1} = (o_i, o_{i+1}]$ are two intervals partitioned by o_i. The approximate numbers of instances in t_i and t_{i+1} are both $\phi\omega$.

If $R(o_i) < \phi$, then the actual number of instances in t_i is $R(o_i) - (\phi - 1)\omega < \phi\omega$, and the actual number of instances in t_{i+1} is $(\phi + 1)\omega - R(o_i) > \phi\omega$. That is, there are $\phi\omega - R(o_i) \le \epsilon\omega$ instances that are actually in t_{i+1}, but are counted into t_i due to the ϵ-approximate quantile o_i. Thus, the error introduced by o_i is at most $\|\frac{\epsilon}{\phi}(\mathcal{U}(\Pr^k(t_i)) - \mathcal{U}(\Pr^k(t_{i+1})))\|$. Similarly, if $R(o_i) < \phi$, the error introduced by o_i is at most $\|\frac{\epsilon}{\phi}(\mathcal{U}(\Pr^k(t_{i+1})) - \mathcal{U}(\Pr^k(t_i)))\|$.

Generally, the maximum overall approximation error introduced by ϵ-approximate quantiles $o_1, \dots, o_{\frac{1}{\phi}-1}$ is

$$\|\widetilde{\mathcal{U}}(\Pr^k(O)) - \mathcal{U}(\Pr^k(O))\|$$

$$= \sum_{i=1}^{\frac{1}{\phi}-1} \left\| \frac{\epsilon}{\phi} (\mathcal{U}(\Pr^k(t_i)) - \mathcal{U}(\Pr^k(t_{i+1}))) \right\|$$

$$\leq \frac{\epsilon}{\phi} (\mathcal{U}(\Pr^k(t_1)) - \mathcal{U}(\Pr^k(t_{\frac{1}{\phi}}))) \leq \epsilon.$$

Inequality 6 can be shown similarly. □

For a sliding window $W(O)$ of a stream, we use $\frac{\widetilde{\mathcal{U}}(\Pr^k(O)) - \widetilde{\mathcal{L}}(\Pr^k(O))}{2}$ as an approximation of $\Pr^k(O)$.

Theorem 10 (Approximation Quality) *For a stream O and sliding window $W(O)$, let $\widetilde{\Pr}^k(O) = \frac{\widetilde{\mathcal{U}}(\Pr^k(O)) - \widetilde{\mathcal{L}}(\Pr^k(O))}{2}$, then $\|\widetilde{\Pr}^k(O) - \Pr^k(O)\| \leq \phi + \epsilon$.*

Proof Following with Theorems 9 and 8, we have

$$\widetilde{\mathcal{U}}(\Pr^k(O)) - \widetilde{\mathcal{L}}(\Pr^k(O)) \leq \mathcal{U}(\Pr^k(O)) - \mathcal{L}(\Pr^k(O)) + 2\epsilon \leq 2\phi + 2\epsilon.$$

Theorem 10 follows with the above inequality directly. □

5.3 Space efficient algorithms using quantiles

The deterministic algorithm discussed in Sect. 3 and the sampling algorithm proposed in Sect. 4 can both be extended using approximate quantile summaries. Due to the loss of information in approximate quantile summaries, the extension of the deterministic algorithm only provides approximate answers.

Using approximate quantile summaries, each stream in a sliding window is represented by $\lceil \frac{1}{\phi} \rceil$ intervals. The upper bound and the lower bound of the top-k probability of each interval can be computed using either the deterministic method or the sampling method.

To compute the upper bound and the lower bound using the deterministic method, we first sort the maximum values and the minimum values of all intervals in the ranking order. Then, by scanning the sorted list once, we can compute the approximate upper bound and the approximate lower bound of the top-k probability of each interval. For each stream O, we maintain the upper bound and the lower bound of the number of instances in $W(O)$ that have been scanned, and the upper bound and the lower bound of $\Pr^k(O)$ so far.

The time complexity of query evaluation in a sliding window using the above extended algorithm is $O(\frac{kn^2}{\phi} + \frac{n}{\phi}\log(n\frac{1}{\phi}))$. The upper bound of approximation error is $\phi + \epsilon$.

To compute the upper bound and the lower bound using the sampling method, we draw m sample units uniformly at random as described in Sect. 4. The difference is, each sample unit contains an interval from each stream. For each interval t of stream O, we define an indicator $X_{t_{\mathcal{U}}}$ to the event that $\|\{t' | t'$ is an interval of $O', O' \neq O, t'.MIN > t.MAX\}\| < k$, and an indicator $X_{t_{\mathcal{L}}}$ to the event that $\|\{t' | t'$ is an interval of $O', O' \neq O, t'.MAX > t.MIN\}\| < k$. The indicator is set to 1 if the event happens; otherwise, the indicator is set to 0. Then, $\mathcal{U}(\Pr^k(t)) = E[X_{t_\mathcal{U}}]$, and $\mathcal{L}(\Pr^k(t)) = E[X_{t_\mathcal{L}}]$. Suppose in sample unit s, the value of $X_{t_\mathcal{U}}$ is $X^s_{t_\mathcal{U}}$, then the expectation $E[X_{t_\mathcal{U}}]$ can be estimated by $\frac{1}{m}\sum_{i=1}^m X^{s_i}_{t_\mathcal{U}}$. Similarly, $E[X_{t_\mathcal{L}}]$ can be estimated by $\frac{1}{m}\sum_{i=1}^m X^{s_i}_{t_\mathcal{L}}$.

The time complexity of query evaluation in a sliding window using the above sampling method is $O(mn\frac{1}{\phi})$, where m is the number of samples. If $m \geq \frac{3\ln\frac{2}{\delta}}{\xi^2}$, $(0 < \delta < 1,\ \xi > 0)$, then, the upper bound of approximation error is $\phi + \epsilon + \xi$ with a probability at least $1 - \delta$.

In the above two extended algorithms using approximate quantile summaries, the space complexity of the algorithms is reduced from $O(n\omega)$ to $O(n\frac{\log^2 \epsilon\omega}{\epsilon^2})$, which is the space complexity of computing ϵ approximate quantiles.

6 Experimental results

In this section, we report a systematic empirical study. All the experiments were conducted on a PC computer with a 3.0 GHz Pentium 4 CPU, 1.0 GB main memory, and a 160 GB hard disk, running the Microsoft Windows XP Professional Edition operating system. Our algorithms were implemented in Microsoft Visual Studio 2005.

6.1 Results on real data sets

To illustrate the effectiveness of probabilistic threshold top-k queries over sliding windows in real applications, we use the seismic data collected from the wireless sensor network monitoring volcanic eruptions (http://fiji.eecs.harvard.edu/Volcano). 16 sensors were deployed at Reventador, an active volcano in Ecuador. Each of the 16 sensors continuously sampled seismic data, and the last 60 seconds of data from each node was examined at the base station. To detect the eruption, it is interesting to continuously report the top-k monitoring locations with the highest seismic values in the last 60 seconds.

The seismic data reported by each sensor is treated as an uncertain stream, and each data record is an instance. We tested probabilistic threshold top-k queries with different parameter values on the data set. Limited by space, we only report the answers to a probabilistic threshold top-k query with $k = 5$ and $p = 0.4$ in this paper. We consider a sliding window width of 60 instances per stream. The answers to the query in 10 consecutive sliding windows are reported in Table 4. As comparison, we also compute the average value of each stream in each sliding window and report the top-5 streams with the highest average seismic values.

The answers to the probabilistic threshold top-k query listed in Table 4 reveal the following interesting patterns. First, the seismic values reported by sensors O_2 and O_{16} are consistently among the top-5 with high confidence in the 10 sliding windows. The rankings of seismic values in those locations are stable. Second, the seismic values reported by sensor O_6 is among the top-5 with high confidence in the first 5 sliding windows. The rankings of seismic values reported by sensor O_6 drop after sliding window W_5. Third, the seismic values reported by sensor O_4 is ranked among top-5 with high confidence only in sliding window W_4.

Table 4 The answers to a probabilistic threshold top-k query in 10 consecutive sliding windows ($\omega = 60$) and the answers to a traditional top-k query on average values

Window ID	PTK query ($k = 5$, $p = 0.4$)	Top-5 query on average data
W_1	O_2, O_6, O_{16}	$O_2, O_4, O_6, O_{14}, O_{16}$
W_2	O_2, O_6, O_{16}	$O_2, O_4, O_6, O_{14}, O_{16}$
W_3	O_2, O_6, O_{16}	$O_2, O_4, O_6, O_{14}, O_{16}$
W_4	O_2, O_4, O_6, O_{16}	$O_2, O_4, O_6, O_{14}, O_{16}$
W_5	O_2, O_6, O_{16}	$O_2, O_4, O_6, O_{14}, O_{16}$
W_6	O_2, O_{16}	$O_2, O_4, O_6, O_{14}, O_{16}$
W_7	O_2, O_{16}	$O_2, O_4, O_6, O_{14}, O_{16}$
W_8	O_2, O_{16}	$O_2, O_4, O_6, O_{14}, O_{16}$
W_9	O_2, O_{16}	$O_2, O_4, O_6, O_{14}, O_{16}$
W_{10}	O_2, O_{16}	$O_2, O_4, O_6, O_{14}, O_{16}$

A traditional top-5 query on the average seismic values in each sliding window reports $\{O_2, O_4, O_6, O_{14}, O_{16}\}$ consistently in the 10 sliding windows. They do not reflect the above interesting patterns.

Simple example shows that continuous probabilistic threshold top-k queries on uncertain streams provide meaningful information which cannot be captured by the traditional top-k queries on aggregate data.

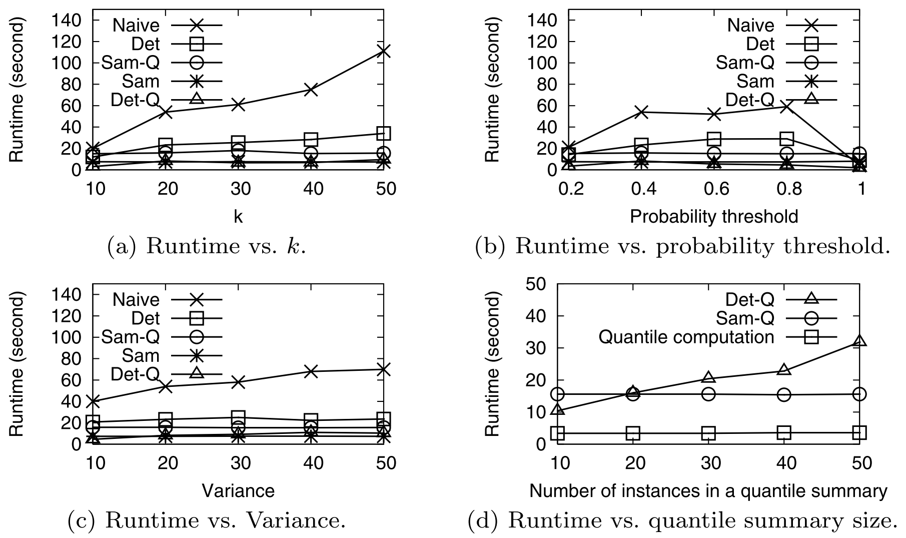
6.2 Synthetic data set Setup

In this performance study, we use various synthetic data sets to evaluate the efficiency of the algorithms and the query evaluation quality. By default, a data set contains 100 uncertain streams, and the sliding window width is set to $\omega = 200$. Thus, there are $20,000$ instances in each sliding window. The data in a sliding window is already held in main memory. The scores of instances from one stream follow a normal distribution. The mean μ is randomly picked from a domain [0, 1000], and the variance σ is randomly picked from [0, 10]. We add 10% noise by using 10σ as the variance. Moreover, the query parameter $k = 20$, and the probability threshold $p = 0.4$. The number of samples drawn in the sampling algorithm is $1,000$. In quantile summaries, $\phi = 0.1$, and $\epsilon = 0.02$. The reported results are the average values in 5 sliding windows.

We test the following four algorithms: the *deterministic exact* method (*Det*) in Sect. 3.2; the *sampling* method (*Sam*) in Sect. 4; the *extended deterministic* method using quantile summaries (*Det-Q*) and the *sampling* method using quantile summaries (*Sam-Q*) in Sect. 5.3.

6.3 Efficiency and approximation quality

Figure 5 shows the runtime of the four algorithms. To show the effectiveness of the compatible dominant set technique and the pruning techniques discussed in Sect. 3.2, we also plot the runtime of the method (*Naive*) discussed in Sect. 3.1, which does



(a) Runtime vs. k.
(b) Runtime vs. probability threshold.
(c) Runtime vs. Variance.
(d) Runtime vs. quantile summary size.

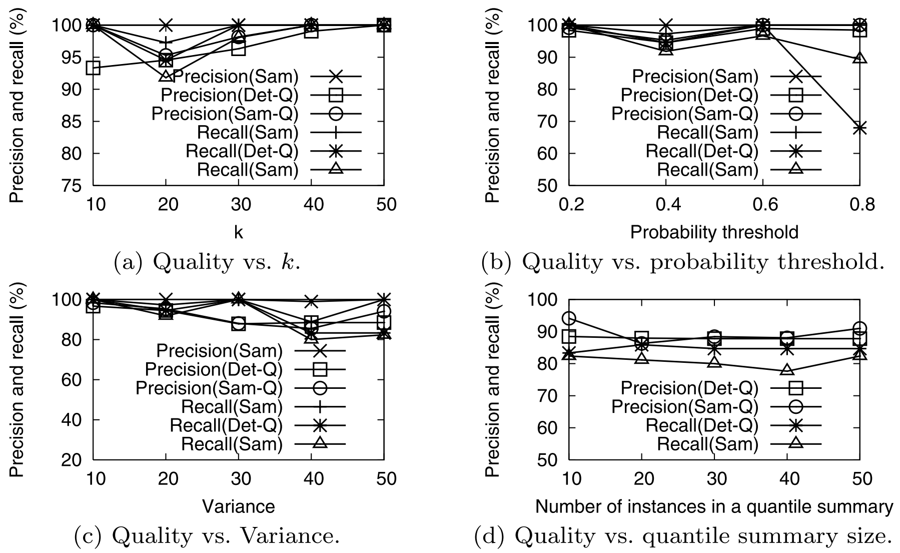
Fig. 5 Efficiency

not explore the sharing between sliding windows. We evaluate the probabilistic top-k queries in 5 consecutive sliding windows. In the first sliding window, the runtime of the "Naive" algorithm and the "Det" algorithm is the same. But in the next 4 sliding windows, the "Naive" algorithm recomputes the top-k probabilities without using the results in the previous sliding windows. But the "Det" algorithm adopts the incremental window maintenance techniques, and thus requires very little computation. Therefore, by average, the runtime of the "Naive" algorithm is approximately 5 times greater than the runtime of the "Det" algorithm. Among all methods, the sampling method and the algorithm using quantiles have much less runtime than the deterministic methods.

Figure 5(a) shows that when parameter k increases, the runtime of the naive method and the deterministic method also increases. With a larger k, more instances are likely to be ranked top-k, and thus more instances have to be read before pruning techniques take effects. Moreover, the Poisson binomial recurrence used by those two methods has a linear complexity with respect to k. However, the deterministic method has a clear advantage over the naive method, which shows the effectiveness of the compatible dominant set technique, and the pruning using highest possible rank. The runtime in the sampling methods and the *Det-Q* method is more sensitive to k, since those techniques have very small overhead increase as k increases.

In Fig. 5(b), as the probability threshold increases, the runtime of the naive method and the deterministic method first increase, and then drop when p is greater than 0.8. As indicated by Theorem 2, if p is small, we can determine that many streams can pass the threshold after checking only a small number of their instances; if p is very large, we can also determine that many streams fail the threshold after checking a small number of instances.

In the synthetic data set, the instances of each stream follow a normal distribution. If the variance of the distribution is larger, then the ranks of instances are more di-



(a) Quality vs. k. (b) Quality vs. probability threshold. (c) Quality vs. Variance. (d) Quality vs. quantile summary size.

Fig. 6 Approximation quality

verse. Thus, we may have to scan more instances in order to determine whether the top-k probability of a stream can pass the probability threshold. Figure 5(c) verifies our analysis.

We test how the two parameters ϕ and ϵ affect the efficiency of the methods using quantiles. $\lceil\frac{1}{\phi}\rceil$ is the number of instances kept in a quantile summary. In Fig. 5(d), we set the value of ϕ to $0.1, 0.05, 0.033, 0.025, 0.02$, and the corresponding number of instances in a quantile summary is $10, 20, 30, 40, 50$, respectively. Only the runtime of *Det-Q* increases when more instances are kept. Since ϵ is typically very small, and does not affect the runtime remarkably, we omit the details here due to limited space.

Figure 6 compares the precision and the recall of the three approximation algorithms using the same settings as in Fig. 5. In general, all three methods have good approximation quality, and the sampling method achieves a higher precision and recall. We notice that, in Fig. 6(b), the recall of the deterministic methods using quantiles decreases when the probability threshold increases. This is because a larger probability threshold reduces the size of answer sets. Moreover, in Fig. 6(c), the precision and recall of all methods decreases slightly as the variance increases. When the variance gets larger, the instances of a stream distribute more sparsely. Thus, we may need more samples to capture the distribution.

We also test our methods on the uncertain data streams generated using the Gamma distribution $\Gamma(k,\theta)$. The mean μ is randomly picked from a domain $[0, 1000]$. We change the variance σ from 10 to 50. The scale parameter θ is set to $\frac{\sigma}{\mu}$ and the shape parameter k is set to $\frac{\mu^2}{\sigma}$. The efficiency and the approximation quality are shown in Fig. 7. The results are very similar to Figs. 5(c) and 6(c). This shows that the performance of our algorithms is not sensitive to the types of score distributions of the data sets.

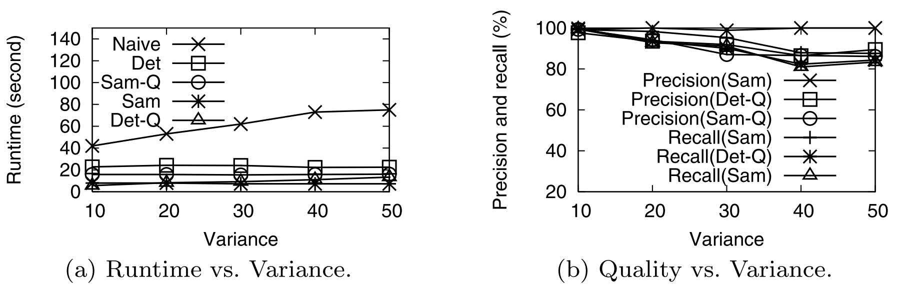


Fig. 7 Efficiency and approximation quality on data sets under the Gamma distribution

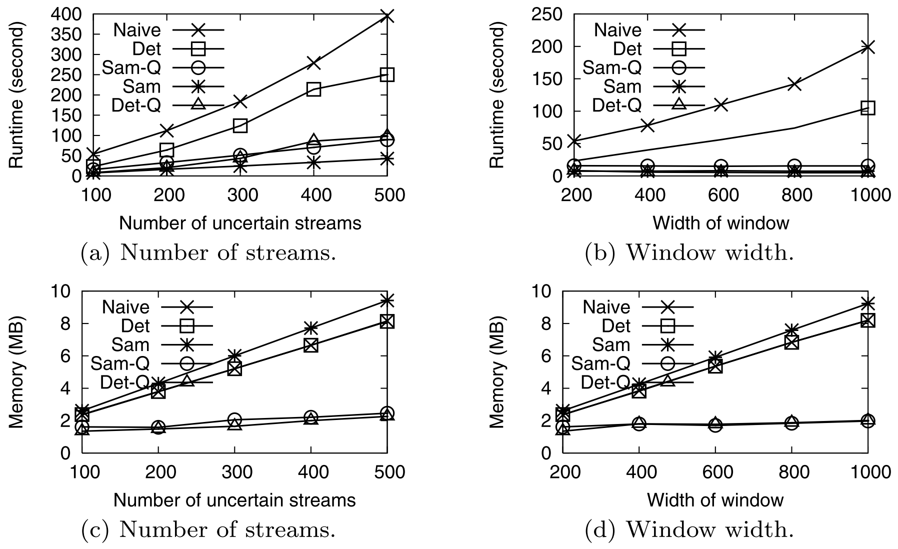


Fig. 8 Scalability

6.4 Scalability

To test the scalability of the algorithms, we first fixed the sliding window width to 200, and change the number of uncertain streams from 100 to 500. The maximum number of instances in a window is 100, 000. As the number of streams increases, the Poisson binomial recurrence takes more time in the deterministic method. Thus, the runtime increases. However, all methods are linearly scalable. The results are shown in Fig. 8(a).

Then, we fix the number of streams to 100, and vary the sliding window width from 200 to 1, 000. The runtime in the deterministic method increases substantially faster than the other methods. The sampling method is more stable, because its runtime is related to only the sample size and the number of streams. For the methods using quantile summaries, after compressing the instances in to a quantile summary, the increase of sliding window width does not affect the runtime noticeably. The results are shown in Fig. 8(b).

In terms of memory usage, Figs. 8(c) and 8(d) show the scalability of each algorithm with respect to the number of uncertain streams and the sliding window width, respectively. The memory used by the deterministic exact algorithm and the sampling algorithm increases linearly, since it is proportional to the number of instances in a sliding window. The memory used by the extended deterministic method using quantiles and the sampling method using quantiles does not change dramatically, because the number of instances for each object in the sliding window only depends on parameter ϕ.

7 Related work

In this section, we briefly review the existing work highly related to our study and point out the differences.

7.1 Continuous queries on probabilistic streams

To the best of our knowledge, [11, 34, 35, 37] are the only existing studies on continuous queries on probabilistic data streams, which are highly related to our study.

In [11], a probabilistic data stream is a (potentially infinite) sequence of uncertain tuples, where each tuple is associated with a membership probability p $(0 < p \leq 1)$, meaning that the tuple takes a probability p to appear in an instance (i.e., a possible world) of the probabilistic stream. It is assumed that tuples are independent from each other. Conventional stream sketching methods are extended to such probabilistic data streams to approximate answers to complex aggregate queries.

Jayram et al. [34, 35] adopt a different probabilistic data stream model. A probabilistic data stream contains a (potentially infinite) sequence of uncertain objects, where each uncertain object is represented by a set of instances and each instance carries a membership probability. An uncertain object arrives in whole at a time and does not change after the arrival. In other words, uncertain objects do not evolve over time. New uncertain objects keep arriving. Several one pass streaming algorithms are proposed to estimate the statistical aggregates of the probabilistic data.

Most recently, Kanagal and Deshpande [37] propose a probabilistic sequence model that considers the temporal and spatial correlations among data. Given a set of uncertain attributes $(A_1, \ldots, A_m)$, each uncertain attribute A_i $(1 \leq i \leq m)$ is a discrete random variable in domain $dom(A_i)$ whose distribution is evolving over time. A probabilistic sequence contains, for each time instant t, an instance $(v_1^t, \ldots, v_m^t)$ for $(A_1, \ldots, A_m)$, where each v_i^t $(1 \leq i \leq m)$ is a random variable in domain $dom(A_i)$ with certain probability distribution. It is a Markov sequence since the random variables at t only depends on the random variables at $t - 1$. Graphical models are used to describe the correlations among the random variables in two consecutive instants. Query answering are considered as inferences over the graphical models.

Our study is different from [11, 34, 35, 37] in following two important aspects. First, the uncertain stream model proposed in this paper is substantially different from the ones proposed before. In the probabilistic sequence model proposed in [37], each element in the stream is a random variable (distribution). While we model an uncertain stream as a series of sample instances generated by a temporal random variable. The set of random variables (i.e., uncertain objects) are fixed. The distributions of those random variables evolve over time. Our model handles some application scenarios that are not covered by the models in [11, 34, 35, 37].

Second, we focus on continuous probabilistic threshold top-k queries on sliding windows, a novel type of queries on uncertain data streams that have not been addressed before. [11, 34, 35] deal with aggregates on a whole stream. The operators discussed in [37] cannot be directly used to answer continuous probabilistic threshold top-k queries.

7.2 Top-k queries on uncertain data

There are numerous existing methods for answering top-k queries on static data. The threshold algorithm (TA) [48] is one of the fundamental algorithms. Several variants of TA have been proposed, such as [22].

Recently, ranking queries are extended to uncertain data. In [54], Soliman et al. proposed U-Top$k$ queries and U-kRanks queries. A U-Top$k$ query returns a k-tuple sorted list which has the highest probability to be the top-k list in possible worlds. Therefore, the co-existence probability of tuples in the top-k lists affects the results heavily. However, when k is large, the probability of the answers to a U-Top$k$ query may become very small. A U-kRanks query finds the tuple of the highest probability at each ranking position. Thus, the tuples returned by a U-kRanks query may not be a valid top-k tuple list in any possible world, and a tuple may appear more than once in the answer set. Lian and Chen developed the spatial and probabilistic pruning techniques for U-kRanks queries [40].

In [49], Ré et al. consider arbitrary SQL queries and the ranking is on the probability that a tuple satisfies the query instead of using a ranking function. Zhang and Chomicki developed the global top-k semantics on uncertain data which returns k tuples having the largest probability in the top-k list, and gave a dynamic programming algorithm [56].

Hua et al. give efficient algorithms for probabilistic threshold top-k queries (PTK queries for short) on static uncertain data in [30, 31]. The continuous probabilistic threshold top-k queries discussed in this paper are based on the study of PTK queries in [30, 31]. Compared to the U-Top$k$ queries and U-kRanks queries, the PTK queries address a different application scenario.

Example 9 (Comparison among U-Top$k$, U-kRanks and PTK queries) Consider the speed monitoring scenario in Example 1. A traffic analyst wants to know the top-2 speeding locations so that more efforts can be placed at those locations for speeding control. If there are three sensors A, B and C deployed at different locations. At time $9:00$AM, three records r_A, r_B, and r_C are reported from those sensors with associated confidences: $r_A = 110$ km/h with $\Pr(r_A) = 0.1$, $r_B = 100$ km/h with $\Pr(r_B) = 0.4$, and $r_C = 90$ km/h with $\Pr(r_C) = 0.8$. What are the top-2 speeding locations?

A U-Top2 query reports C as the answer, since $\langle r_C \rangle$ is the most probably top-2 list in all possible worlds, whose probability is 0.432.

A U-2Ranks query reports C as the most probably 1-st speeding location with confidence 0.432. For the 2-nd speeding location, C is reported again with confidence 0.288.

A probabilistic threshold top-2 query with probability threshold $p = 0.3$ returns B and C as the 2 locations whose top-2 probabilities are no smaller than p. Their top-2 probabilities are $\Pr^2(r_B) = 0.4$ and $\Pr^2(r_C) = 0.72$.

Therefore, location B has a probability of 0.4 of being ranked in the top-2 speeding locations. But it cannot be reported by U-Top$k$ queries or U-kRanks queries.

In the speed monitoring application, users are more interested in the individual locations with a high probability of being ranked top-k. The co-occurrence of speeding locations in the top-k list or the speeding location at certain ranking position are not important. Therefore, probabilistic threshold top-k queries are more appropriate than U-Top$k$ queries and U-kRanks queries in this application scenario.

Although top-k queries in various forms are explored for uncertain static data, all the existing studies do not consider streaming uncertain data which is the focus of this paper.

7.3 Distributed top-k query processing

Distributed top-k query processing focuses on reducing communication cost while providing high quality answers. [6] studies top-k monitoring queries which continuously report the k largest values from data streams produced at physically distributed locations. In their model, there are multiple logical data objects and each object is associated with an overall logical data value. Updates to overall logical data values arrive incrementally over time from distributed locations. Efficient techniques are proposed to compute and maintain the top-k logical data objects over time with low communication cost among distributed locations and a bounded error tolerance. In [45], an algorithmic framework is proposed to process distributed top-k queries, where the index lists of attribute values are distributed across a number of data peers. The framework provides high quality approximation answers and reduces network bandwidth, local peer load, and query response time.

There are two major differences between our study and distributed top-k query processing. First, different data models are adopted. In our study, each object is associated with a set of instances representing the distribution of the object; while in [6], each object has only one overall logical data value, and [45] considers a set of tuples with distributed attribute index lists. Second, the queries are different. In our study, a probabilistic threshold top-k query finds all objects whose probability of being ranked top-k is greater than a threshold; while in [6], a top-k query finds the top-k objects having the highest overall logical data values, and in [45], tuples are ranked according to a monotonic aggregate function on a set of attributes. The techniques developed in [6] and [45] cannot be used to answer probabilistic threshold top-k queries efficiently.

7.4 Continuous ranking and quantile queries on data streams

A rank or quantile query is to find a data entry with a given rank against a monotonic order specified on the data. Rank queries have several equivalent variations [13, 27, 57] and play important roles in many data stream applications.

It has been shown in [34] that an exact computation of rank queries requires memory size linear to the size of a data set by any one-scan technique, which may be impractical in on-line data stream computation where streams are massive in size and fast in arrival speed. Approximately computing rank queries over data streams has been investigated in the form of quantile computation.

A ϕ-quantile ($0 < \phi \leq 1$) of a collection of N data elements is the element with rank $\lceil \phi N \rceil$ against a monotonic order specified on the data set. The main paradigm is to continuously and efficiently maintain a small data structure (sketch/summary) in space over data elements for online queries. It has been shown in [4, 25, 26, 44] that a space-efficient ϕ-approximation quantile sketch can be maintained so that, for a quantile ϕ, it is always possible to find an element at rank r' with the uniform precision guarantee $\|r' - \lceil \phi N \rceil\| \leq \epsilon N$. Due to the observation that many real data sets often exhibit skew towards heads (or tails depending on a given monotonic order), relative rank error (or biased) quantile computation techniques have been recently developed [12, 13, 57], which give better rank error guarantees towards heads.

Top-k queries have been extended to data streams. In [46], Mouratidis el al. study the problem of continuous monitoring top-k queries over sliding windows. Very recently, [17] improves the performance of the algorithms.

All the existing studies on continuous ranking or quantile queries on data streams do not consider uncertain data. Those methods cannot be extended to probabilistic threshold top-k queries on uncertain data directly due to the complexity of possible worlds semantics. In this paper, we investigate native methods for uncertain data streams.

7.5 Uncertain data processing

Recently, modeling and querying uncertain data has become an active research direction [3, 20, 39, 50].

In this study, we adopt the working model for uncertain data proposed in [50], which describes the existence probability of an instance tuple in an uncertain data set and the constraints (i.e., exclusiveness).

There have been a few important models of uncertain data and queries. Cheng et al. [9] propose a general classification of probabilistic queries and evaluation algorithms over uncertain data sets. In [10], uncertain data are modeled as a set of ranges and the associated probability density functions. Probabilistic threshold queries are proposed on the above model, which return the results whose confidence is higher than a user defined threshold. Tao et al. [55] proposed a U-tree index to facilitate probabilistic range queries on uncertain objects represented by multi-dimensional probability density functions. Singh et al. [52] extended the inverted index and signature tree to index uncertain categorical data. Dalvi and Suciu [16] developed an efficient algorithm to evaluate arbitrary SQL queries on probabilistic databases and rank the results by their probability. In [14], they showed that the complexity of evaluating conjunctive queries on a probabilistic database is either PTIME or $\#P$-complete.

On the system level, Orion [51] is an uncertain database management system that supports the attribute and tuple level uncertainty with arbitrary correlations. Three basic operations are defined to perform selection and to compute marginal distributions and joint distributions. Other relational operations can be defined based on the three basic operations. Both continuous and discrete uncertainty is handled in Orion [53].

7.6 Continuous sensor stream monitoring

Sensor stream monitoring focuses on maintaining the answers to deterministic queries in sensor networks, while reducing the energy consumption as much as possible. Deshpande et al. [19] build a correlation-aware probabilistic model from the stored and current data in a sensor network, and use the probabilistic model to answer SQL queries. Only approximate answers with certain confidence bounds are provided, but the cost of data maintenance and query answering is significantly reduced. More specifically, Liu et al. [42] study Min/Max query monitoring over distributed sensors. In their scenario, queries are submitted to a central server, and the major cost in query answering is the communication cost between the central server and distributed sensors. The authors model the reading of each sensor as a random variable, whose probability distribution can be obtained from historical data. Those distributions are used to estimate the answer to any Min/Max query. The server also contacts a small number of sensors for their exact readings, in order to satisfy the user specified error tolerance. [28] considers the applications where multiple sensors are deployed to monitor the same region. A sampling method is used to answer continuous probabilistic queries. The values of sensors that have little effect on the query results are sampled at a lower rate.

There are three differences between our study and [19, 28, 42]. First, the uncertain data models adopted in the above work are different from the uncertain stream model discussed in this paper, due to different application requirements. Second, the monitored queries are different: [19, 28, 42] deal with general SQL queries, Min/Max queries, and probabilistic queries, respectively, but our study focuses on top-k queries on uncertain streams specifically. Last, while the above work only provides approximate answers, our study can provide a spectrum of methods including an exact algorithm, a random method and their space efficient versions.

8 Conclusions

In this paper, we proposed a novel uncertain data stream model and continuous probabilistic threshold top-k queries on uncertain streams, which are different from the existing probabilistic stream models and queries. A deterministic method and a sampling method, as well as their space efficient versions using quantiles were developed to answer those queries. Experimental results on real data sets and synthetic data sets were reported, which show the effectiveness and efficiency of the methods.

As future work, it is interesting to extend our methods to other probabilistic data stream models and other continuous preferences queries. Particularly, there are four important directions.

Uncertain data streams with non-uniformly distributed instances. In this paper, we assume that the membership probabilities of all instances in a sliding window are identical. This is suitable for the applications where instances are generated using simple random sampling. However, in other applications, instances with non-identical membership probabilities may be generated. Therefore, it is important to investigate how to adapt the methods developed in this paper for the case of non-identical membership probabilities.

In the exact algorithm, all techniques can be used except for the "compatible dominant set" technique. The "compatible dominant set" technique reuse the dominant set of an instance in the previous sliding window, as long as the number of instances from each object in the dominant set does not change. This only holds when the membership probabilities of instances are identical. Therefore, new techniques that can reuse the dominant set of instances in the case of non-identical membership probabilities need to be developed.

Second, the sampling method can be used without any change. We simply draw samples according to the distribution of instances for each object in the sliding window.

Last, in order to use the space efficient algorithms developed in this paper, the ϕ-quantile summary needs to be redefined. The major idea is to partition the instances ranked in the value ascending order into intervals, so that the sum of membership probabilities of instances in each interval is at most ϕ. For an uncertain object $W_\omega^t(O)$ containing a set of instances $o_1, \dots, o_\omega$, where each instance o_i is associated with a membership probability $\Pr(o_i)$ $(1 \le i \le \omega)$. The instances are ranked in the value ascending order. We partition $o_1, \dots, o_\omega$ into b exclusive intervals $t_i = [o_{z_i}, o_{z'_i}]$, where

$$\begin{cases} z_i = 1, & i = 1; \\ z_i = z'_{i-1} + 1, & 1 < i \le b; \\ z'_i = \max\limits_{j \ge z_i} \left\{ j \,\middle|\, \sum\limits_{x=z_i}^{j} \Pr(o_x) \le \phi \right\}, & 1 \le i \le b. \end{cases}$$

Therefore, the probability of each interval is at most ϕ. Moreover, there are at most $2\lceil \frac{1}{\phi} \rceil$ intervals. This is because, in the worst case, each interval only contains one instance, which means that the sum of membership probabilities of any two consecutive instances is greater than ϕ. Then, if the number of instances is greater than $2\lceil \frac{1}{\phi} \rceil$, the sum of membership probabilities of all instances will be greater than 1, which conflicts with the fact that the sum of membership probabilities of all instances of one object is 1.

Though the approximate top-k probabilities can be computed similarly as discussed in the paper, details need to be worked out as future work.

Uncertain data streams with correlations. Correlations may exist among uncertain data streams. For example, in the speed monitoring application, if two speed sensors are deployed at the same location, then the readings of the two sensors are mutually exclusive. That is, only the reading of one sensor can exist in a possible world. More generally, the complex correlation among two or more uncertain data streams can be represented by the joint distribution of their readings. How to continuously monitor the top-k uncertain data streams in such cases is highly interesting.

Continuously monitoring probabilistic threshold top-k queries with different parameter values. There are two parameters, the query parameter k and the probability threshold p, in probabilistic threshold top-k queries. To achieve the best analysis results, users may be interested in how query results change as parameters vary. To support the interactive analysis, it is highly desirable to monitor probabilistic threshold top-k queries on uncertain data streams with different parameters.

To achieve this goal, we plan to design an index that stores the top-k probabilities of instances and objects in the current sliding window. Once constructed, the index should be easy to update as the sliding window advances. Moreover, the index should be memory efficient.

Continuously monitoring probabilistic threshold top-k aggregate queries. In this paper, we focus on probabilistic threshold top-k selection queries, where the ranking function is applied on a single instance. Another category of top-k queries is top-k aggregate queries [32], where the ranking function is applied on a group of instances. The top-k groups with highest scores are returned as results. It is interesting to investigate how to extend the techniques discussed in this paper to handle top-k aggregate queries on uncertain data streams.

References

1. Abiteboul, S., Kanellakis, P., Grahne, G.: On the representation and querying of sets of possible worlds. In: Proceedings of the 1987 ACM SIGMOD International Conference on Management of Data (SIGMOD'87), pp. 34–48. ACM Press, New York (1987)
2. Angluin, D., Valiant, L.G.: Fast probabilistic algorithms for Hamiltonian circuits and matchings. In: Proceedings of the Ninth Annual ACM Symposium on Theory of Computing (STOC'77), pp. 30–41. ACM Press, New York (1977)
3. Antova, L., Koch, C., Olteanu, D.: 10^{10^6} worlds and beyond: efficient representation and processing of incomplete information. In: Proceedings of the 2007 IEEE 23rd International Conference on Data Engineering (ICDE'07), 2007
4. Arasu, A., Manku, G.S.: Approximate counts and quantiles over sliding windows. In: Proceedings of the Twenty-third ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS'04), Paris, France, 2004
5. Babcock, B., Babu, S., Datar, M., Motwani, R., Widom, J.: Models and issues in data stream systems. In: Proc. 21st ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS'02), Madison, WI, 2002
6. Babcock, B., Olston, C.: Distributed top-k monitoring. In: SIGMOD Conference, pp. 28–39, 2003
7. Babu, S., Widom, J.: Continuous queries over data streams. SIGMOD Rec. **30**, 109–120 (2001)
8. Chen, J., DeWitt, D., Tian, F., Wang, Y.: NiagraCQ: A scalable continuous query system for internet databases. In: Proc. 2000 ACM-SIGMOD Int. Conf. Management of Data (SIGMOD'00), Dallas, TX, 2000
9. Cheng, R., Kalashnikov, D.V., Prabhakar, S.: Evaluating probabilistic queries over imprecise data. In: Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data (SIGMOD'03), pp. 551–562. ACM Press, New York (2003)
10. Cheng, R., Xia, Y., Prabhakar, S., Shah, R., Vitter, J.S.: Efficient indexing methods for probabilistic threshold queries over uncertain data. In: VLDB, pp. 876–887, 2004
11. Cormode, G., Garofalakis, M.N.: Sketching probabilistic data streams. In: SIGMOD Conference, pp. 281–292, 2007
12. Cormode, G., Korn, F., Muthukrishnan, S., Srivastava, D.: Effective computation of biased quantiles over data streams. In: ICDE'05, 2005
13. Cormode, G., Korn, F., Muthukrishnan, S., Srivastava, D.: Space- and time-efficient deterministic algorithms for biased quantiles over data streams. In: PODS'06, 2006

14. Dalvi, N., Suciu, D.: The dichotomy of conjunctive queries on probabilistic structures. In: Proceedings of the Twenty-sixth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS'07), pp. 293–302. ACM Press, New York (2007)
15. Dalvi, N., Suciu, D.: Management of probabilistic data: foundations and challenges. In: Proceedings of the Twenty-sixth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS'07), pp. 1–12. ACM Press, New York (2007)
16. Dalvi, N.N., Suciu, D.: Efficient query evaluation on probabilistic databases. In: Proceedings of the Thirtieth International Conference on Very Large Data Bases, Toronto, Canada, pp. 864–875, 2004
17. Das, G., Gunopulos, D., Koudas, N., Sarkas, N.: Ad-hoc top-k query answering for data streams. In: VLDB, pp. 183–194, 2007
18. Das, G., Gunopulos, D., Koudas, N., Tsirogiannis, D.: Answering top-k queries using views. In: VLDB, pp. 451–462, 2006
19. Deshpande, A., Guestrin, C., Madden, S., Hellerstein, J.M., Hong, W.: Model-driven data acquisition in sensor networks. In: VLDB, pp. 588–599, 2004
20. Dey, D., Sarkar, S.: A probabilistic relational model and algebra. ACM Trans. Database Syst. **21**(3), 339–369 (1996)
21. Doucet, A., Vo, B.N., Andrieu, C., Davy, M.: Particle filtering for multi-target tracking and sensor management. In: Proceedings of the Fifth International Conference on Information Fusion, pp. 474—481, 2002
22. Fagin, R., Lotem, A., Naor, M.: Optimal aggregation algorithms for middleware. In: PODS '01: Proceedings of the Twentieth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, pp. 102–113. ACM Press, New York (2001)
23. Gardner, D., Bostrom, R.: Ohio and Kentucky approach to data archiving in Cincinnati. http://trb.mtc.ca.gov/urban/adus/Ohio.pdf
24. Gehrke, J., Korn, F., Srivastava, D.: On computing correlated aggregates over continuous data streams. In: Proc. 2001 ACM-SIGMOD Int. Conf. Management of Data (SIGMOD'01), Santa Barbara, CA, pp. 13–24, 2001
25. Gilbert, A.C., Kotidis, Y., Muthukrishnan, S., Strauss, M.: How to summarize the universe: Dynamic maintenance of quantiles. In: VLDB, pp. 454–465, 2002
26. Greenwald, M., Khanna, S.: Space-efficient online computation of quantile summaries. In: SIGMOD Conference, pp. 58–66, 2001
27. Gupta, A., Zane, F.X.: Counting inversions in lists. In: SODA '03: Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 253–254. Society for Industrial and Applied Mathematics, Philadelphia (2003)
28. Han, S., Chan, E., Cheng, R., yiu Lam, K.: A statistics-based sensor selection scheme for continuous probabilistic queries in sensor networks. Real-Time Syst. **35**(1), 33–58 (2007)
29. Hoeffding, W.: On the distribution of the number of successes in independent trials. Ann. Math. Stat. **27**, 713–721 (1956)
30. Hua, M., Pei, J., Zhang, W., Lin, X.: Efficiently answering probabilistic threshold top-k queries on uncertain data (extended abstract). In: Proc. International Conference on Data Engineering (ICDE'08), Cancun, Mexico, 2008
31. Hua, M., Pei, J., Zhang, W., Lin, X.: Ranking queries on uncertain data: a probabilistic threshold approach. In: Proc. ACM International Conference on Management of Data (SIGMOD'08), Vancouver, Canada, 2008
32. Ilyas, I.F., Beskales, G., Soliman, M.A.: A survey of top-k query processing techniques in relational database systems. ACM Comput. Surv. **40**(4), 1–58 (2008)
33. Imielinski, T., Witold Lipski, J.: Incomplete information in relational databases. J. ACM **31**(4), 761–791 (1984)
34. Jayram, T.S., Kale, S., Vee, E.: Efficient aggregation algorithms for probabilistic data. In: SODA '07: Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 346–355. Society for Industrial and Applied Mathematics, Philadelphia (2007)
35. Jayram, T.S., McGregor, A., Muthukrishnan, S., Vee, E.: Estimating statistical aggregates on probabilistic data streams. In: PODS, pp. 243–252, 2007
36. Kalton, G.: Introduction to Survey Sampling, 1st edn. Sage Publications, Thousand Oaks (1983)
37. Kanagal, B., Deshpande, A.: Efficient query evaluation over temporally correlated probabilistic streams. In: ICDE, 2009
38. Lange, K.: Numerical Analysis for Statisticians. Springer, New York (1999)

39. Lee, S.K.: Imprecise and uncertain information in databases: an evidential approach. In: Proceedings of the Eighth International Conference on Data Engineering (ICDE'92), pp. 614–621. IEEE Computer Society, Washington (1992)
40. Lian, X., Chen, L.: Probabilistic ranked queries in uncertain databases. In: Proc. 2008 International Conference on Extended Data Base Technology (EDBT'08), 2008
41. Lin, X., Lu, H., Xu, J., Yu, J.X.: Continuously maintaining quantile summaries of the most recent n elements over a data stream. In: ICDE, pp. 362–374, 2004
42. Liu, Z., Sia, K.C., Cho, J.: Cost-efficient processing of min/max queries over distributed sensors with uncertainty. In: SAC, pp. 634–641, 2005
43. Manku, G.S., Rajagopalan, S., Lindsay, B.G.: Approximate medians and other quantiles in one pass and with limited memory. In: SIGMOD Conference, pp. 426–435, 1998
44. Manku, G.S., Rajagopalan, S., Lindsay, B.G.: Random sampling techniques for space efficient online computation of order statistics of large datasets. In: SIGMOD '99: Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data, pp. 251–262. ACM Press, New York (1999)
45. Michel, S., Triantafillou, P., Weikum, G.: Klee: A framework for distributed top-k query algorithms. In: VLDB, pp. 637–648, 2005
46. Mouratidis, K., Bakiras, S., Papadias, D.: Continuous monitoring of top-k queries over sliding windows. In: SIGMOD Conference, pp. 635–646, 2006
47. Munro, J.I., Paterson, M.: Selection and sorting with limited storage. Theor. Comput. Sci. **12**, 315–323 (1980)
48. Nepal, S., Ramakrishna, M.: Query processing issues in image(multimedia) databases. In: Proceedings of the 15th International Conference on Data Engineering, pp. 22–29. IEEE Computer Society, Washington (1999)
49. Ré, C., Dalvi, N., Suciu, D.: Efficient top-*k* query evaluation on probabilistic data. In: Proceedings of the 23nd International Conference on Data Engineering (ICDE'07), Istanbul, Turkey. IEEE, New York (2007)
50. Sarma, A.D., Benjelloun, O., Halevy, A., Widom, J.: Working models for uncertain data. In: ICDE '06: Proceedings of the 22nd International Conference on Data Engineering (ICDE'06), p. 7. IEEE Computer Society, Washington (2006)
51. Singh, S., Mayfield, C., Mittal, S., Prabhakar, S., Hambrusch, S., Shah, R.: Orion 2.0: native support for uncertain data. In: SIGMOD '08: Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data, pp. 1239–1242. ACM Press, New York (2008)
52. Singh, S., Mayfield, C., Prabhakar, S., Shah, R., Hambrusch, S.: Indexing uncertain categorical data. In: Proceedings of the 2007 IEEE 23rd International Conference on Data Engineering (ICDE'07), 2007
53. Singh, S., Mayfield, C., Shah, R., Prabhakar, S., Hambrusch, S.E., Neville, J., Cheng, R.: Database support for probabilistic attributes and tuples. In: ICDE, pp. 1053–1061, 2008
54. Soliman, M.A., Ilyas, I.F., Chang, K.C.C.: Top-*k* query processing in uncertain databases. In: Proceedings of the 23nd International Conference on Data Engineering (ICDE'07), Istanbul, Turkey. IEEE, New York (2007)
55. Tao, Y., Cheng, R., Xiao, X., Ngai, W.K., Kao, B., Prabhakar, S.: Indexing multi-dimensional uncertain data with arbitrary probability density functions. In: Proceedings of the 31st International Conference on Very Large Data Bases (VLDB'05), VLDB Endowment, pp. 922–933, 2005
56. Zhang, X., Chomicki, J.: On the semantics and evaluation of top-*k* queries in probabilistic databases. In: Proc. the Second International Workshop on Ranking in Databases (DBRank'08), 2008
57. Zhang, Y., Lin, X., Xu, J., Korn, F., Wang, W.: Space-efficient relative error order sketch over data streams. In: ICDE '06: Proceedings of the 22nd International Conference on Data Engineering (ICDE'06), p. 51. IEEE Computer Society, Washington (2006)