Noname manuscript No.
(will be inserted by the editor)

Probabilistic n-of-N Skyline Computation over Uncertain Data Streams

Wenjie Zhang[1] · **Aiping Li**[2] ✉ · **Muhammad Aamir Cheema**[1] · **Ying Zhang**[1] · **Lijun Chang**[1]

the date of receipt and acceptance should be inserted later

Abstract Skyline operator is a useful tool in multi-criteria decision making in various applications. Uncertainty is inherent in real applications due to various reasons. In this paper, we consider the problem of efficiently computing probabilistic skylines against the most recent N uncertain elements in a data stream seen so far. Specifically, we study the problem in the n-of-N model; that is, computing the probabilistic skyline for the most recent n ($\forall n \leq N$) elements, where an element is a probabilistic skyline element if its skyline probability is not below a given probability threshold q. Firstly, an effective pruning technique to minimize the number of uncertain elements to be kept is developed. It can be shown that on average storing only $O(\log^d N)$ uncertain elements from the most recent N elements is sufficient to support the precise computation of all probabilistic n-of-N skyline queries in a d-dimension space if the data distribution on each dimension is independent. A novel encoding scheme is then proposed together with efficient update techniques so that computing a probabilistic n-of-N skyline query in a d-dimension space is reduced to $O(d \log\log N + s)$ if the data distribution is independent, where s is the number of skyline points. A trigger based technique is provided to process continuous n-of-N skyline queries. Extensive experiments demonstrate that the new techniques on uncertain data streams can support on-line probabilistic skyline query computation over rapid data streams.

1 Introduction

Skyline analysis has been shown as a useful tool in multi-criterion decision making. Given a certain data set D, an element $s_1 \in D$ dominates another element $s_2 \in D$ if s_1 is better

Wenjie Zhang was partially supported by ARC DE120102144 and DP120104168. Aiping Li (Corresponding Author) was partially supported by State Key Development Program of Basic Research of China (No. 2013CB329601) and National Key Technology R&D Program (No. 2012BAH38B-04). Muhammad Aamir Cheema was partially supported by ARC DE130101002 and DP130103405. Ying Zhang was partially supported by ARC DP110104880, DP130103245 and UNSW ECR grant PSE1799.

[1]University of New South Wales, Sydney, NSW, Australia
[2]National University of Defense Technology, China
E-mail: zhangw@cse.unsw.edu.au, apli1974@gmail.com (corresponding author), macheema@cse.unsw.edu.au, yingz@cse.unsw.edu.au, ljchang@cse.unsw.edu.au

than s_2 in at least one aspect and not worse than s_2 in all other aspects. The skyline on D comprises of elements in D that are not dominated by any other element from D.

Uncertain data analysis is an important issue in many emerging important applications, such as sensor networks, trend prediction, moving object management, data cleaning and integration, economic decision making, and market surveillance. Uncertainty is inherent in such applications due to various factors such as data randomness and incompleteness, limitation of equipment, and delay or loss in data transfer. In many scenarios, uncertain data is collected in a streaming fashion. Uncertain streaming data computation has attracted significant research attention and the existing work mainly focuses on aggregations, top-k queries [1–3], etc.

Skyline computation over uncertain streaming data has many applications and has been studied in [4,5]. For instance, in an on-line shopping system products are evaluated in various aspects such as *price*, *condition* (e.g., brand new, excellent, good, average, etc), and *brand*. A customer may want to select a product, say laptops, based on the multiple criteria such as low price, good condition, and brand preference. In the application, each seller is also associated with a "trustability" value which is derived from customers' feedback on the seller's product quality, delivery handling, etc; the trustability value may be regarded as the "occurrence" probability of the product since it represents the probability that the product occurs exactly as described in the advertisement in terms of delivery and quality. For simplicity, we assume that a customer only prefers a particular brand and remove the brand dimension from ranking. Table 1 lists four qualified results. Both L_1 and L_4 are skyline points regarding (price, condition), L_1 is better than (dominates) L_2, and L_4 is better than L_3. Nevertheless, L_1 is posted long time ago, and the trustability of L_4 is quite low. In such applications, customers may want to continuously monitor on-line advertisements by selecting the candidates for the best deal - skyline points. Clearly, we need to "discount" the dominating ability from offers with too low trustability. Moreover, too old offers may not be quite relevant. We model such an on-line selection problem as probabilistic skyline against sliding windows by treating on-line advertisements as an uncertain data stream (see Section 2 for details) such that each data element (advertisement) has an occurrence probability. Moreover, different users may have different favorite thresholds of the number N of most recent elements to monitor. Therefore, it is important for an information provider (system) to organize the most recent N elements in an effective way, so that any "n-of-N skyline" queries (the computation of the skyline of the most recent n ($\forall n \leq N$) elements) can be processed efficiently.

Table 1 Laptop Advertisements.

Product ID	Time	Price	Condition	Trustability
L_1	107 days ago	\$ 550	excellent	0.80
L_2	5 days ago	\$ 680	excellent	0.90
L_3	2 days ago	\$ 530	good	1.00
L_4	today	\$ 200	good	0.48

[6,7] are the first attempts to investigate skyline computation on certain sliding windows while [6] is the first paper to tackle such a problem in the n-of-N model. To the best of our knowledge, this paper is the first work to study skyline queries in context of n-of-N model over uncertain data streams. Our contribution can be summarized as follows.

1. We formally define the problem of probabilistic skyline computation over uncertain data streams regarding the n-of-N model.
2. An efficient pruning technique has been developed to minimize the number $\mathcal{N}$ ($\mathcal{N} \leq N$) of uncertain elements to be kept in the most recent N elements for processing all probabilistic n-of-N queries. We showed that in a d-dimensional space $\mathcal{N} = O(\log^d N)$ if the data distribution on each dimension is independent.
3. A novel encoding scheme with linear size $O(\mathcal{N})$ on the stored elements is developed, together with the efficient update algorithms based R-tree and interval tree techniques. This encoding scheme effectively reduces the time complexity for processing a probabilistic n-of-N skyline query to $O(\log \mathcal{N} + s)$ from $O(n \log n)$ for $d = 2, 3$ and $O(n \log^{d-2} n)$ for $d \geq 4$, where s is the number of skyline points.
4. A trigger based technique for continuously processing probabilistic skyline query following the n-of-N model is developed. Upon the arrival of a new data element, it guarantees $O(\log \delta)$ time to update the current query result where δ is the number of element changing from the current result to the new result. It takes $O(\log s)$ time to update the triggers list per result change.
5. Extensive experiments indicated that the new techniques can accommodate on-line computation against very rapid data streams.

The rest of the paper is organized as follows. In section 2, we formally define probabilistic skyline queries over uncertain data streams regarding the n-of-N model and provide necessary preliminaries. Section 3 presents the minimum candidate set to process probabilistic n-of-N skyline queries and the encoding scheme. Section 4 provides the techniques for continuously maintaining the indexing structures for the candidate set. Continuous probabilistic n-of-N skylines queries techniques are presented in Section 5. Results of comprehensive performance studies are discussed in section 6. Some extensions of the problem studied in the paper are discussed in Section 7. Related works are summarized in Section 8 and section 9 concludes the paper.

2 Background

We present problem definition and necessary preliminaries in this section.

2.1 Problem Definition

For two exact d-dimensional elements u and v, u dominates v, denoted by $u \prec v$, if $u.i \leq v.i$ for every $1 \leq i \leq d$, and there exists a dimension j with $u.j < v.j$. Given a set of elements, the *skyline* consists of all elements which are not dominated by any other elements. In many applications, a data stream is *append-only*; that is, there is no deletion of data elements involved. In this paper, we study the skyline computation problem restricted to the append-only data stream model. In a data stream, elements are positioned according to their relative arrival ordering and labeled by integers. Note that the position/label $\kappa(a)$ means that the element a arrives $\kappa(a)$-th in the data stream.

In an uncertain data stream DS, each element $a \in DS$ has a probability $P(a)$ ($0 < P(a) \leq 1$) to occur where $a.i$ (for $1 \leq i \leq d$) denotes the i-th dimension value. Given a sequence DS of uncertain data elements, a *possible world* W is a subsequence of DS. The probability of W to appear is $P(W) = \Pi_{a \in W} P(a) \times \Pi_{a \notin W}(1 - P(a))$. Let ω be the set of all possible worlds, then $\sum_{W \in \omega} P(W) = 1$. We use $SKY(W)$ to denote the set of elements

in W that form the skyline of W. The probability that an element a appears in the skylines of the possible worlds is $P_{sky}(a) = \sum_{a \in SKY(W), W \in \omega} P(W)$. $P_{sky}(a)$ is called the *skyline probability* of a. Equation 1 below can be immediately verified.

$$P_{sky}(a) = P(a) \times \Pi_{a' \in DS, a' \prec a}(1 - P(a')) \tag{1}$$

Denote N as the size of the sliding window, namely we only keep the recent N elements in the data stream. In the rest of the paper, we abuse $P_{sky}(a)$ to denote the skyline probability for a to be a skyline element within the recent N elements. Note that a is also within the most recent N elements P_N. Namely,

$$P_{sky}(a) = P(a) \times \Pi_{a' \in P_N, a' \prec a}(1 - P(a')) \tag{2}$$

In n-of-N model, skyline computation is supported for any window length n with $n \leq N$. Suppose an element a is within the most recent n ($n \leq N$) elements, we denote $P_{sky,n}(a)$ as the skyline probability of a computed regarding the most recent n elements only. Namely,

$$P_{sky,n}(a) = P(a) \times \Pi_{\kappa(a), \kappa(a') \geq M-n+1, a' \prec a}(1 - P(a')) \tag{3}$$

where M is the totaly number of elements in the data stream so far and $\kappa(a), \kappa(a') \geq M - n + 1$ implicates that a and a' are within the most recent n elements. Clearly we have $P_{sky,n}(a) \geq P_{sky}(a)$, which implies that even a is not a skyline element in the sliding window with size N, it may still be a skyline element in the most recent n elements for some $n \leq N$. We denote $\mathcal{P}$*n-of-N* query as the query to retrieve probabilistic skyline elements against any most recent n ($n \leq N$) elements in the data stream regarding a given probability threshold.

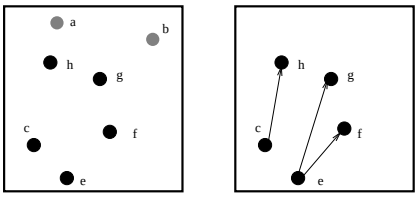
Problem Statement. Given a data stream DS in which each uncertain element $a \in DS$ is associated with an occurrence probability $P(a)$ ($0 < P(a) \leq 1$) indicating the likelihood that a exists in DS. We say an element a is a probabilistic skyline element within the most recent n elements if $P_{sky,n}(a) \geq q$. A $\mathcal{P}$*n-of-N* query retrieves the probabilistic skyline elements within the most recent n ($\forall n \leq N$) data elements in the data stream DS.

2.2 Preliminaries

n-of-N model. As an important method to support query processing over different thresholds of window size, n-of-N model is firstly proposed in [8] to efficiently maintain quantile summaries. We will investigate the problem of effectively organising the most recent N elements in an uncertain data stream seen so far, so that the computation of probabilistic skyline against any most recent n ($n \leq N$) elements can be processed efficiently. Note that a *sliding window* model [9] is a special case of the n-of-N model where $n = N$.

Stabbing queries. Given a set of m intervals and a *stabbing* point p in the 1-dimensional space, the *stabbing query* is to find all intervals which contain p. By the interval tree techniques in [10], a stabbing query can be processed in $O(\log m + l)$ where l is the number of intervals in the result. By storing an interval only in the tree node that is the lowest common ancestor (LCA) of the two end points of the interval, the space complexity of the interval tree is $O(m)$. It has been also shown that the time complexity of an update (insertion or deletion) to an interval tree is amortized to $O(\log m)$ per deletion or insertion. Note that the intervals here can be closed, half closed, or open at both ends.

n-of-N Skyline Query over Exact Data Streams. [6] studies skyline computation over exact sliding windows following the n-of-N model. It is observed that over exact data streams, if an element a is dominated by a newer element a', then a will never be a skyline for any recent n ($n \in N$) elements since a' expires later than a. It is also proved that in such a case removing a from the data stream will not affect computation of n-of-N skyline queries processing. Thus, minimum candidate set R_N comprises of elements in the data stream which are not dominated by newer elements.



(a) Stream S (b) Dominance Graph

Fig. 1 Dominance Graph

Example 1 As shown Figure 1 (a), assume the elements a, b, ..., h arrive at time 1, 2, ..., 7, respectively. Since a and b are both dominated by elements newer than them, the candidate set is $\{c, e, f, g, h\}$.

In Figure 1 (a), g is dominated by c and e. It is noticed that if the dominance relation $e \rightarrow g$ is released due to the expiration of e then the dominance $c \rightarrow g$ has already been released since c expires earlier than e. Therefore, it is only necessary to keep $e \rightarrow g$ to hold a "lock" on g. In R_N, a dominance relation $e' \rightarrow e$ is *critical* if and only if e' is the youngest one (but older than e) in R_N, which dominates e; that is, $\kappa(e')$ is maximized among all the elements (other than e), in R_N, dominating e. A dominance graph G_{R_N} is constructed where the edge set consists of all critical dominance relations. Figure 1 (b) depicts the dominance graph of Figure 1(a). The encoding scheme is as follows: 1) every edge $e' \rightarrow e$ in G_{R_N} is represented by the interval $(\kappa(e'), \kappa(e)]$, and 2) each root e in G_{R_N} is represented by the interval $(0, \kappa(e)]$. Thus, an element $e \in R_N$ is in the answer of an n-of-N query ($n \leq N$) if and only if $\kappa(e)$ is the right end of an interval $(a, \kappa(e)]$ that contains $M - n + 1$. Based on such a scheme, the problem of computing an n-of-N query is converted to the *stabbing query* problem with the stabbing point $M - n + 1$. Namely, stab the intervals by $M - n + 1$, and then return the data elements e such that each $\kappa(e)$ is the right end of a stabbed interval.

Example 2 Regarding the example in Figure 1, the dominant graph can be encoded by the following intervals: $(0, 3]$, $(0, 4]$, $(3, 7]$, $(4, 5]$, and $(4, 6]$. When $n = 6$, $M - n + 1 = 2$ as $M = 7$. Clearly, the intervals $(0, 3]$ and $(0, 4]$ are the results of stabbing query; consequently, c and e are the skyline elements for the most recent 6 elements among the 7 already arrived elements.

Various Dominating Probabilities in Uncertain Data Streams. For each element a in the sliding window DS, we use $P_{new}(a)$ to denote the probability that none of the elements in the sliding window which are newer than a (i.e., arrives later than a) dominates a; that is,

$$P_{new}(a) = \Pi_{a' \in DS, a' \prec a, \kappa(a') > \kappa(a)}(1 - P(a')) \quad (4)$$

Note that $\kappa(a') > \kappa(a)$ means that a' arrives after a. We use $P_{old}(a)$ to denote the probability that none of the elements which are older than a (i.e., arrives earlier than a) dominates a; that is,

$$P_{old}(a) = \Pi_{a' \in P_N, a' \prec a, \kappa(a') < \kappa(a)}(1 - P(a')) \quad (5)$$

The following equation (6) can be immediately verified.

$$P_{sky}(a) = P(a) \times P_{old}(a) \times P_{new}(a). \quad (6)$$

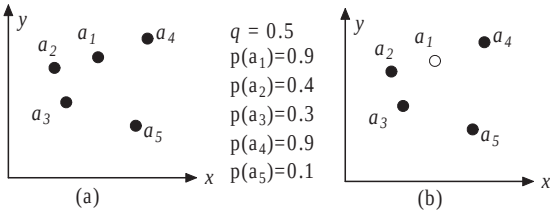


Fig. 2 A Sequence of Data Elements

Example 3 Regarding the example in Figure 2(a) where the occurrence probability of each element is as depicted, assume that $N = 5$, and elements arrive according to the element subindex order; that is, a_1 arrives first, a_2 arrives second, ..., and a_5 arrives last. $P_{new}(a_4) = 1 - P(a_5) = 0.9$ and $P_{old}(a_4) = (1 - P(a_2))(1 - P(a_3))(1 - P(a_1)) = 0.042$, and $P_{sky}(a_4) = P(a_4)P_{new}(a_4)P_{old}(a_4) = 0.034$. If N = 4, a_1 expires once a_5 arrives as shown in Figure 2 (b). Then $P_{old}(a_4)$ = $(1 - P(a_2))(1 - P(a_3))$ = 0.42 and $P_{sky}(a_4)$ = 0.34.

3 Minimizing the Number of Uncertain Elements and the Encoding Scheme

In this section, we first minimize the number of uncertain elements to be kept for processing all $\mathcal{P}$*n-of-N* queries. Then, we present an effective encoding scheme on the stored elements to support efficient $\mathcal{P}$*n-of-N* query processing.

3.1 Minimizing the Number of Elements

As introduced in Section 2.2, in an exact data stream, an element e is "redundant" if it is dominated by a $newer$ element e'. In an uncertain data stream DS, if an uncertain element e is dominated by a $newer$ uncertain element e', e could still be a probabilistic skyline point regarding a given probability threshold q.

Example 4 In Figure 3, there are 7 uncertain elements a, b, c, d, f, g, and h. The order of the elements in the stream is the alphabetic order. The occurrence probability of each element and the probability threshold q are as illustrated in the figure. As shown, element c is dominated by a newer element h, however, as $P_{new}(c)$ = 1 - $P(h)$ = 0.9, $P_{old}(c)$ = 1, $P_{sky}(c)$ = 0.63 $> q$. Thus, g is a probabilistic skyline in the sliding window within the most recent 7 elements.

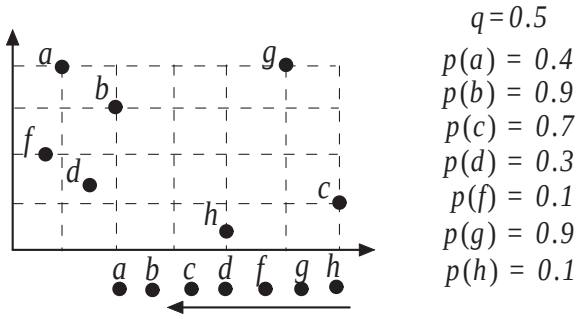


Fig. 3 Uncertain Data Stream

Example 4 shows that modeling redundant elements in an uncertain data stream requires further analysis on the probabilities associated with each element. Remind that $P_{new}(e)$ refers to the probability that uncertain element e is not dominated by any elements newer than it. Let P_N denote the most recent N elements, and $R_{N,q}$ denote the set of elements in the most recent N elements with P_{new} values not smaller than q; that is,

$$R_{N,q} = \{e | e \in P_N, P_{new}(e) \geq q\} \tag{7}$$

In [5], $R_{N,q}$ is proved to be the minimum set of elements to be maintained to correctly answer probabilistic skyline queries over the most recent N elements in the data stream.

Theorem 1 *$R_{N,q}$ is the minimum set of elements to be maintained to correctly compute probabilistic skyline queries for sliding window size of N regarding probability threshold q.*

The proofs of Theorem 1 can be found in [5] based on the following properties of $R_{N,q}$.

- Processing skyline query based on $R_{N,q}$ only will not miss any skyline points.
- For a skyline point, its skyline probability computed based on $R_{N,q}$ only is equal to that computed based on all most recent N elements.
- $R_{N,q}$ is the minimum set of points to guarantee correct retrieval of skylines within the most recent N elements.

The following theorem states that $R_{N,q}$ is also the minimum set of elements to be maintained to correctly compute skyline queries for any recent n elements where $n \leq N$, namely, to correctly retrieve results for $\mathcal{P}n$*-of-N* queries.

Theorem 2 *$R_{N,q}$ is the minimum set of elements to be maintained to correctly compute $\mathcal{P}n$-of-N queries regarding probability threshold q.*

Proof When $n = N$, the n-of-N skyline query equals to skylines over the entire sliding window, namely the case in [5]. Since $\mathcal{P}n$*-of-N* queries also support the case when $n = N$, $R_{N,q}$ is the *minimum* set of elements to maintain for $\mathcal{P}n$*-of-N* queries. Furthermore, for $\forall n < N$, let R_n denote the most recent n elements and $R_{n,q}$ denote element sets in R_n with P_{new} values not smaller than the probability threshold q, namely, $R_{n,q} = \{e | e \in P_n, P_{new,n}(e) \geq q\}$. Based on Theorem 1, $R_{n,q}$ is the minimum set of elements for correct computation of skylines elements within the most recent n elements. Noticing that $R_{n,q}$ is equal to retrieve elements from $R_{N,q}$ which are the within most recent n elements, $R_{n,q} \subseteq R_{N,q}$. Thus using $R_{N,q}$ we can *correctly* retrieve results for $\mathcal{P}n$*-of-N* queries. Thus the theorem holds.

Size of $R_{N,q}$. Elements in the candidate set $R_{N,q}$ can be regarded as the skyline points in the $d+1$ dimensional space by including *time* as an additional dimension. This is because P_{new} can be regarded as the non-dominance probability in such a $d+1$ space. Thus, with the assumption that all points follow uniform distribution, the expected size of $R_{N,q}$ is $ln^d(N)/(d+1)!$.

3.2 Encoding $R_{N,q}$ for $\mathcal{P}n$-of-N Queries

As introduced in Section 2.2, in the candidate set R_N of an exact data stream, we say *an element e' dominates* e is a critical dominance relation if e' is the youngest element (yet older than e) that dominates e. For a value n ($n \leq N$), if e' is not within the most recent n elements (i.e., $\kappa(e') < M - n + 1$ where M is the total number of element seen so far), e is a skyline element.

Similar to the philosophy in encoding the candidate set R_N for n-of-N queries over exact data streams, we aim to identify the most *critical* dominance relationship for elements inside the candidate set $R_{N,q}$ for uncertain sliding windows. Remind that the skyline probability of an element e within the most recent N elements consists of two parts besides its own occurrence probability, $P_{old}(e)$ representing the probability that e is not dominated by any element older than e and $P_{new}(e)$ representing the probability that e is not dominated by any elements newer than it. Furthermore, similar to $P_{sky,n}(a)$ which refers to the skyline probability of an element a computed regarding the most recent n elements in Equation 3, P_{old} value of an element could also be defined for the most recent n ($n \leq N$) elements as follows, given that $\kappa(a) \geq M - n + 1$, namely a is also within the most recent n elements.

$$P_{old,n}(a) = \Pi_{a' \in DS, \kappa(a) \geq M-n+1, \kappa(a') \geq M-n+1, \kappa(a) > \kappa(a'), a' \prec a}(1 - P(a')) \qquad (8)$$

The following equation is immediate since all elements newer than a are within the most recent n elements if a is within the most recent n elements. Hereafter, if discussing $P_{sky,n}$ or $P_{old,n}$ values for an element a it is assumed that a is within the most recent n elements

$$P_{sky,n}(a) = P(a) \times P_{old,n}(a) \times P_{new}(a). \qquad (9)$$

Example 5 Continue with the example in Figure 2, assume $N = 5$ and $n = 3$, namely we are interested in only the most recent three elements a_3, a_4 and a_5. $P_{old,3}(a_4) = 1 - P(a_3) = 0.7$, $P_{new}(a_4) = 1 - P(a_5) = 0.9$, so $P_{sky,3} = 0.567$.

Consider an increase in the value n ($n \leq N$). $P_{old,n}(e)$ is non-increasing with the increase of value n since more elements older than e may be included in the most recent n elements and contribute to the P_{old} value. On the other hand, $P_{new}(e)$ does not change with the value of n because elements which contribute to $P_{new}(e)$ are all newer than e. Thus, to determine the critical dominance relation for e in an uncertain data stream is to locate the element e_c with $\kappa(e_c) = M - n_c + 1$ making $P_{sky,n_c}(e) \geq q$ invalid, where n_c is minimized (or $\kappa(e_c)$ is maximized). We use $e_c \xrightarrow[q]{c} e$ to denote that e_c probabilistically critically dominates e regarding the probability threshold q. Namely,

$$n_c = \arg\min_n P_{sky,n}(e) < q$$

Clearly, for any value $n < n_c$, e is a probabilistic skyline element within the most recent n elements.

Example 6 In Figure 3, for element g, $P_{new}(g)$ = 1 - $P(h)$ = 0.9. When n = 5, namely, within the most recent 5 elements, $P_{old,5}(g) = (1 - P(f)) \times (1 - P(d))$ = 0.63, and $P_{sky,5}(f)$ = 0.5103 $> q$; When n = 6, $P_{old,6}(g) = (1-P(f)) \times (1-P(d)) \times (1-P(b))$ = 0.063 $< q$. Thus, n_c = 6 is the minimum number of elements in the sliding window to make f unqualified to be a probabilistic skyline element. Equally speaking, b is the youngest element in the sliding window which dominates g and after the expiration of which g is a probabilistic skyline element. Element b probabilistically critically dominates g, namely $b \overset{c}{\underset{q}{\rightarrow}} g$, as $\kappa(b) = M - n_c + 1$ = 2.

Once the critical dominance relation is determined for an uncertain element e, we can have the dominance graph $G_{R_{N,q}}$ which is an edge set consisting all probabilistic critical dominance relations. Note that for an element e in the candidate set $R_{N,q}$, if $P(e) \times P_{new}(e) < q$, e does not have a critical dominance relation available since e is not a skyline element for any value of n ($n \leq N$). However, we still need to keep e in $R_{N,q}$ as shown in Theorem 2 because deleting e will affect the skyline probability calculation for other elements in $R_{N,q}$. Based on $G_{R_{N,q}}$, given a value of n ($n \leq N$), e is a skyline element for n if either of the following two conditions hold.

- e is a root in the dominance graph $G_{R_{N,q}}$, or
- there is an edge $e_c \overset{c}{\underset{q}{\rightarrow}} e$ in $G_{R_{N,q}}$, such that e_c arrives earlier than the n-th most recent element (i.e., $\kappa(e_c) < M - n + 1 \leq \kappa(e)$).

The encoding scheme for $G_{R_{N,q}}$ is as follows. 1) Every edge $e_c \overset{c}{\underset{q}{\rightarrow}} e$ in $G_{R_{N,q}}$ is represented by an interval $(\kappa(e_c), \kappa(e)]$. 2) Each root e in $G_{R_{N,q}}$ is represented by the interval $(0, \kappa(e)]$. Let $I_{R_{N,q}}$ denote the interval tree on the intervals obtained by the encoding scheme on $G_{R_{N,q}}$. So, an element e in $G_{R_{N,q}}$ is the answer of a $\mathcal{P}$*n-of-N* query ($n \leq N$) if and only if $\kappa(e)$ is the right end of an interval that contains $M - n + 1$. The problem of computing $\mathcal{P}$*n-of-N* query is thus converted to the *stabbing query* problem with stabbing point $M - n + 1$ as discussed in Section 2.2. Namely, stab the intervals in $I_{R_{N,q}}$ by $M - n + 1$, and then return the data elements e such that $\kappa(e)$ is the right end of a stabbed interval.

Example 7 In Figure 3, M = 7 since there are 7 elements in the stream so far. Suppose N = 6. The candidate set $R_{N,q}$ consists of all recent 6 elements b, c, d, f, g and h since the P_{new} value of each element is not below the threshold q. Only the elements with occurrence probabilities not smaller than q are considered when computing probabilistic dominance relations, i.e., b, c, g. Element b is dominated by two newer elements d and f, and $P_{sky}(b) = P(b) \times (1 - P(d)) \times (1 - P(f))$ = 0.567, so b is a root in the dominance graph $G_{R_{N,q}}$. c is dominated by newer element h with $P_{sky}(c) = P(c) \times (1 - P(h))$ = 0.63, so c is a root in $G_{R_{N,q}}$. g is dominated by newer element h and older elements b, d, f. From Example 6, b probabilistically critically dominates g ($b \overset{c}{\underset{q}{\rightarrow}} g$). So the interval tree $I_{R_{N,q}}$ consists the following intervals by encoding the dominance relations in the dominance graph $G_{R_{N,q}}$: $(0, 2]$, $(0, 3]$ and $(2, 6]$. If $n = 5$ (to retrieve probabilistic skylines within the recent 6 elements), we stab the interval tree $I_{R_{N,q}}$ with $M - n + 1$ = 3 and c, g will be returned as the final results.

Note that in exact data streams, any non-redundant element in R_N is a skyline point for some $n \leq N$. However, in uncertain streams, this statement no longer holds. For instance, an element e may have an occurrence probability lower than q, disabling it from being a skyline point for any n values. However, we still need to keep e in $R_{N,q}$ if its $P_{new}(e)$ value

is above q. This is because as proved in [5], removing such elements may incur incorrect probabilistic skyline results.

Time Complexity. The number of intervals kept in $G_{R_{N,q}}$ is $O(|R_{N,q}|)$ since $G_{R_{N,q}}$ is a forest. Thus, the stabbing query which retrieves the results for $\mathcal{P}n$*-of-N* queries runs in $O(log|R_{N,q}|) + s$ where s is the number of probabilistic skyline points within the most recent n elements.

4 Maintaining $R_{N,q}$ and the Encoding Scheme

In the sliding window model, when a new element e_{new} arrives, the window slides to accommodate e_{new}, and the oldest element e_{old} moves out of the window range and should be removed. These may also trigger updates in the non-redundant element set $R_{N,q}$ and the dominance interval tree $I_{R_{N,q}}$. Algorithm 1 describes the overall framework to handle the key issues while the window slides for the uncertain stream.

Algorithm 1: Framework

```
1 while a new element e_new arrives do
2     if κ(e_new) ≤ N then
3         Updates introduced by e_new;
4     else
5         Updates introduced by e_new;
6         Updates introduced by e_old;
7         Deletion of e_old;
8     Insertion of e_new;
```

As shown in Algorithm 1, when the sliding window is not yet full (i.e., e_{new} is the i-th element and $i \leq N$), we only need to handle the updates introduced by e_{new} and insertion of e_{new}, where insertion of e_{new} identifies the qualification of e_{new} regarding the candidate set $R_{N,q}$ and $I_{R_{N,q}}$. After the window is full, we need to further address the deletion of the oldest element e_{old} (i.e., the element which arrives $(M - N + 1)$-th in the stream) from the sliding window as well as the updates introduced by the deletion of e_{old}. In the following subsections we discuss the three major steps, *insertion of* e_{new}, *updates introduced by* e_{new}, and *updates introduced by* e_{old}, respectively. Deletion of e_{old} is trivial since we only need to delete e_{old} from the candidate set $R_{N,q}$ and interval tree $I_{R_{N,q}}$ if necessary. Naively processing these steps requires a sequential scan of elements in $R_{N,q}$.

4.1 Insertion of e_{new}

Since the most recent element e_{new} is not dominated by any element newer than it, $P_{new}(e_{new})$ = 1 and we insert e_{new} into the aggregate R-tree indexing the candidate set $R_{N,q}$. Next, if $P(e_{new}) \geq q$ the skyline probability of e_{new} should be explored to determine its probabilistic dominance relation. Otherwise (i.e., $P(e_{new}) < q$), the identification of probabilistic dominance relation is not necessary since e_{new} has no chance to be a probabilistic skyline element for any $n \leq N$.

Remind that to determine the critical dominance relation for e_{new} is to locate the element e_c with $\kappa(e_c) = M - n_c + 1$ making $P_{sky,n_c}(e_{new}) \geq q$ invalid, where n_c is minimized (or $\kappa(e_c)$ is maximized). A naive way to do so is to firstly sort all elements in $R_{N,q}$ decreasingly according to timestamps of elements; then scan the sorted elements and update $P_{sky}(e_{new})$ by multiplying $1 - P(e)$ if an element $e \prec e_{new}$. Each element dominating e_{new} in the process will be kept in a list called critical dominance list of e_{new}, denoted as $L_c(e_{new})$ which is decreasingly sorted based on timestamps. The scan stops when the first element e_c with timestamp $\kappa(e_c) = M - n_c + 1$ making $P_{sky,n_c}(e_{new}) < q$ is encountered.

Considering that elements in $R_{N,q}$ are indexed by an R-tree. We propose to use the best first search paradigm on R-tree to determine the critical dominance relation for e_{new}. For a node v in the R-tree indexing $R_{N,q}$, we record the maximum timestamp $\kappa(v)$ of all descendent elements of v. A max heap H based on $\kappa(v)$ is built to keep the nodes to be expanded. We denote the lower left corner of the minimal bounding box (MBB) of v as v^{lower}. The criteria to expand a node is $v^{lower} \prec e_{new}$. Otherwise (i.e., $v^{lower} \not\prec e_{new}$), no elements from v dominates e_{new} and v will not be expanded. We terminate if the heap is empty, or the current element under investigation e_c with timestamp $\kappa(e_c) = M - n_c + 1$ makes $P_{sky,n_c}(e_{new}) < q$. If such an element is not found, e_{new} is a probabilistic skyline element for the time interval $(0, \kappa(e_{new}]$.

Algorithm 2 depicts above steps. Remind that Algorithm 2 is invoked only when $P(e_{new}) \geq q$. Starting from the root node of $R_{N,q}$, child entries of an intermediate entry v are inserted into the max heap if $v^{lower} \prec e_{new}$ (**Line** 6). If v is a data element and the updated skyline probability of e_{new} remains above q after considering the dominance of v, v is inserted into the critical dominance list of e_{new} (**Line** 10); $P_{sky}(e_{new})$ is also updated accordingly (**Line** 11). Otherwise (i.e., $P_{sky}(e_{new})$ below q), the algorithm terminates with the critical dominance relation identified.

Algorithm 2: Critical Dominance of e_{new}

```
1  max Heap H ← root of R-tree indexing R_{N,q};
2  while H is not empty do
3      v = H.top();
4      H.pop();
5      if v is an intermediate node AND v^{lower} ≺ e_new then
6          insert children entries of v into H;
7      else
8          if v ≺ e_new then
9              if P_sky(e_new) * (1 − P(v)) ≥ q then
10                 insert v into L_c(e_new);
11                 P_sky(e_new)* = 1 − P(v);
12             else
13                 insert (κ(v), κ(e_new)] into I_{R_{N,q}};
14                 found = true; exit;
15 if found = false then
16     insert (0, κ(e_new] into I_{R_{N,q}};
```

4.2 Updates Introduced by e_{new}

Next we handle the updates introduced by e_{new}. If an element e is dominated by e_{new}, we need to update its P_{new} and P_{sky} probability which may render them invalid as a citizen in the non-redundant set $R_{N,q}$ or in the dominance graph $G_{R_{N,q}}$. First of all we retrieve the set of elements dominated by e_{new}, denoted as $D_{e_{new}}$. The following algorithm describes the key update issues related to e_{new}. We maintain a priority query Q initialized to the root node of R-tree R. Q is prioritized according to the levels of nodes, i.e., nodes of higher levels are accessed first.

Algorithm 3: Dealing with $D_{e_{new}}$

```
1 for ∀e ∈ D_{e_new} do
2     if P_new(e) × (1 − P(e_new)) < q then
3         delete e from R_{N,q};
4         if e is the end of an interval in G_{R_{N,q}} then
5             delete the interval;
6     else if P_sky(e) × (1 − P(e_new)) < q then
7         update the probabilistic critical dominance relation of e;
8         P_new(e) = P_new(e) × (1 − P(e_new));
```

Lines 2-5 in Algorithm 3 handle the first case when P_{new} value of e degrades to be smaller than q with the contribution of $1 - P(e_{new})$. In this case we remove e from $R_{N,q}$ and if it has a critical dominance relation captured in $G_{R_{N,q}}$, it is also deleted. **Lines** 6-8 deal with the case where e survives the citizenship test in $R_{N,q}$ but $P_{sky}(e)$ becomes below q after multiplying $1-P(e_{new})$. In this case the probabilistic critical dominance relationship will be re-calculated by visiting the critical dominance list $L_c(e)$ sequentially,

Note that if an element e is deleted from $R_{N,q}$ in the first case ($P_{new}(e)\times(1-P(e_{new})) < q$), we do not need to update the information of elements dominated by e. We formally prove this in the following lemma.

Lemma 1 *If $e_{new} \prec e$ and $P_{new}(e) \times (1 - P(e_{new})) < q$ after the arrival of the new element e_{new}, e could be removed from $R_{N,q}$ without updating the dominating probabilities of elements dominated by e.*

Proof For an element e' and $e \prec e'$, first suppose $\kappa(e') < \kappa(e)$, namely, e' is older than e. Since $e \prec e'$, all elements dominating e also dominates e', so $P_{new}(e') \times (1 - P(e_{new}) < q$. e' should also be removed from $R_{N,q}$; if $\kappa(e') > \kappa(e)$, since $e \prec e'$, the skyline probability of e' computed within the most recent $M - \kappa(e) + 1$ elements, $P_{sky,M-\kappa(e)+1}(e')$, after multiplying $1 - P(e_{new})$, must be smaller than q. Thus, the critical dominance relationship of e', $e_c \xrightarrow[q]{c} e'$, will be re-computed in Algorithm 3 and $\kappa(e_c) > \kappa(e)$, which means deleting e does not affect the critical dominance relationship and dominating probabilities of e'.

4.3 Updates Introduced by e_{old}

For the expired element e_{old}, we first remove it from the candidate set $R_{N,q}$ and the dominance graph if necessary. Next, for each element e dominated by e_{old}, the P_{old} and P_{sky} values of e change and the critical dominance relation might also change. Algorithm 4 illustrates the process. **Line** 1 deletes e_{old} from $R_{N,q}$ and **Line** 3 removes the critical dominance relation from $G_{R_{N,q}}$. Note that we do not need to check every element dominated by e_{old} to remove the effect of P_{old}. Instead, only those which are critically dominated by e_{old} will be actually affected by e_{old}. This is because e_{old} is the oldest element in $R_{N,q}$ and may not contribute to the critical dominance relation of all elements.

Algorithm 4: Expiration of Old Element e_{old}

1 $R_{N,q} := R_{N,q} - \{e_{old}\}$;
2 **if** $(0, \kappa(e_{old})] \in I_{R_{N,q}}$ **then**
3 remove $(0, \kappa(e_{old})]$ from $I_{R_{N,q}}$;
4 **for** $\forall(\kappa(e_{old}), \kappa(e)] \in I_{R_N}$ **do**
5 update $(\kappa(e_{old}), \kappa(e)]$ to $(0, \kappa(e)]$;
6 $P_{sky}(e)$ is updated by discounting $1 - P_{old}$;

5 Continuous $\mathcal{P}$*n-of-N* Queries

A continuous query is issued once and run continuously to generate results along with the updates of underlying streaming datasets. Arrival of a new element in the data stream may invoke update in $\mathcal{P}$*n-of-N* results. A simple way is to re-run the the query processing algorithm (stabbing query in Section 3.2) per arrival of a new element. This takes $O(log\mathcal{N} + s)$ time where $\mathcal{N}$ is the number of intervals in $G_{R_{N,q}}$ and s is the number of elements in the $\mathcal{P}$*n-of-N* result set S_n. In this section, we present a trigger based algorithm which continuously and incrementally updates the $\mathcal{P}$*n-of-N* results. Correctness of our algorithm is based on the following observation.

Proposition 1 *Once a new element e_{new} arrives, the current result S_n of a $\mathcal{P}$n-of-N query may have the following changes after we apply Algorithm 3 to reflect the updates introduced by e_{new}.*

Deletion: An element $e \in S_n$ is removed from S_n, if either e expires, or the updated critical dominance relation for e is: $e' \xrightarrow[q]{c} e$, where $\kappa(e') \geq \mathcal{M} - n + 1$.

Insertion: An element $e \in R_N$ is added to S_n in the following two cases. (1) e is e_{new} and either e_{new} is a root node in $G_{R_{N,q}}$ or for the critical dominance relation of e_{new} $e' \xrightarrow[q]{c} e_{new}$, $\kappa(e') < \mathcal{M} - n + 1$; (2) e is critically dominated by the element e' which just expired from the most recent n elements; namely, $\kappa(e') = \mathcal{M} - n$.

Algorithm 5 below describes the process for continuous $\mathcal{P}$*n-of-N* queries. Note that Algorithm 3 for handling updates incurred by arrival of e_{new} is invoked prior to Algorithm

5. On the other hand, Algorithm 4 may not be necessary unless $n = N$. Suppose elements in the result set S_n are maintained by a min-heap according to arrival order of elements and e_{top} is the top element of the min-heap.

Algorithm 5: Continuous $\mathcal{P}n$-*of-N* Queries

```
1  while new element e_new do
2      M = M + 1;
3      for e ∈ S_n & e ∈ D_{e_new} do
4          if ∃e' ->^c_q e with κ(e') ≥ M − n + 1 then
5              remove e from S_n;
6      if ∄e' ->^c_q e_new with κ(e') ≥ M − n + 1 then
7          insert e_new into S_n;
8      while κ(e_top) < M − n + 1 do
9          remove e_top from S_n;
10         for e_top ->^c_q e ∈ G_{R_{N,q}} do
11             insert e into S_n;
```

In the algorithm, we handle unqualified results in S_n in Lines 3 to 5 and Line 9 with two different cases, 1) after updating the probabilistic critical dominance relation, an element e is critically dominated by an element e' within the most recent n element, namely, e is no longer a probabilistic skyline point after e_{new} arrives, or, 2)e_{top} is no longer within the most recent n elements after e_{new} arrives The new results are added to S_n in Lines 6 to 7 and Lines 10 to 11. There are also two cases to add new results, 1) if e_{new} is a probabilistic skyline within the most recent n element or 2) an element e is critically dominated by e_{top} which just slides out of the most recent n elements.

Time Complexity. Let δ be the number of elements changing from the current result to the new result. It takes $O(\delta)$ time to change the result set. Considering the min-heap keeping the result set S_n. The update cost per element change is $O(\log s)$.

6 Performance Evaluation

In this section, we present the results of a comprehensive performance evaluation of our techniques. As mentioned earlier, there is no existing technique specifically designed to support efficient computation of n-of-N skyline queries over uncertain sliding windows. In our performance study, we implement the most efficient main-memory algorithm for skyline queries over uncertain data streams [5] and use it as a benchmark algorithm to evaluate our techniques.

All algorithms proposed in the paper are implemented in standard C++ with STL library support and compiled with GNU GCC. Experiments are conducted on a PC with Intel Xeon 2.4GHz dual CPU and 4G memory under Debian Linux. In our implementation, MBBs of the uncertain objects are indexed by an R-tree with page size 4096 bytes.

Real Dataset. The real dataset is extracted from the stock statistics from NYSE (New York Stock Exchange). We choose 2 million stock transaction records of Dell Inc. from December 1st 2000 to May 22nd 2001. For each transaction, the average price per volume and total volume are recorded. This 2-dimensional dataset is referred to as *stock* in the following. To evaluate the techniques over uncertain sliding windows, we randomly assign a probability value between 0 and 1 to each transaction; that is, probability values follows *uniform* distribution. Elements arrival order is based on their transaction time.

Synthetic Dataset. We evaluate our techniques against the 3 most popular synthetic benchmark data, *correlated*, *independent*, and *anti-correlated* [11]. We evaluate our techniques against the space dimensions from 2 to 5. To evaluate the techniques over uncertain sliding windows, we use two models *uniform* and *normal* to assign occurrence probability to each element. In *uniform* distribution, the occurrence probability of each element takes a random value between 0 and 1, while in the *normal* distribution, the mean value of occurrence probabilities P_μ varies from 0.1 to 0.9 and standard deviation S_d is set to 0.3. The occurrence probability distribution follows *uniform* distribution by default unless otherwise specified. We assign a random order for elements arrival in a data stream.

The following algorithms are evaluated in this subsection for $\mathcal{P}$*n-of-N* queries.

q-sky : The query processing algorithm for probabilistic skyline queries over uncertain sliding windows in [5].

pnN: Our query processing algorithm (Section 3.2) for $\mathcal{P}$*n-of-N*; that is, the stabbing query processing algorithm.

pmnN : Our algorithms (Section 4)for continuously maintaining the data structures for supporting $\mathcal{P}$*n-of-N* queries.

pcnN : The continuous query processing algorithm in Section 5 for continuously outputting $\mathcal{P}$*n-of-N* results.

6.1 Evaluating query algorithm: pnN

In this set of experiments, we fix $N = 10^6$ and randomly choose 1000 different n values varying from 1000 to 10^6. Each n is thus mapped to a $\mathcal{P}$*n-of-N* query with $N = 10^6$ to evaluate the query processing algorithm pnN. The processing time reported in Figure 4 is the average of the bucket of 1000 queries. As there is no existing work supporting skyline query processing over uncertain sliding windows with variable length, we naively search each candidate kept in the q-sky algorithm [5] and test if it is a probabilistic element over the recent n elements; pnN utilizes the stabbing query processing algorithm over the interval set $I_{R_{N,q}}$. In Figure 4, we vary dimensionality from 2 to 5, and evaluate both q-sky and pnN over the three synthetic datasets anti-correlated, independent, and correlated. As shown, both q-sky and pnN have a better performance over corr dataset and pnN is up to 2 orders of magnitude faster than q-sky. In the more challenging anti dataset, pnN is up to 5 orders of magnitude faster. In the remaining of this subsection, we no longer evaluate the performance of q-sky in our performance study since pnN significantly outperforms q-sky.

Figure 5 reports the impact of different n values on query processing time. The space dimensionality is fixed to 2 and 5 respectively. We also record the average query processing time of 1000 queries. The results in Figure 5 show that the query techniques for $\mathcal{P}$*n-of-N* queries are not very sensitive to the value of n. On the other hand, dimensionality and data distribution have a greater impact over the efficiency.

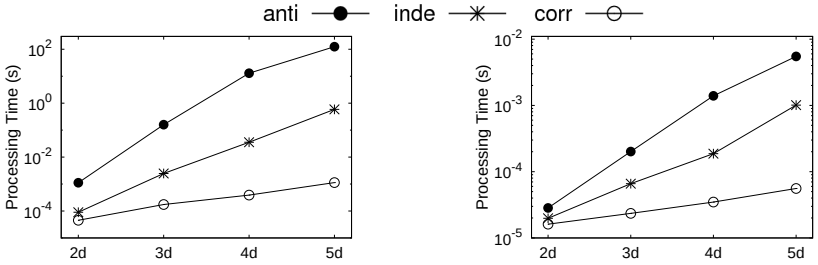
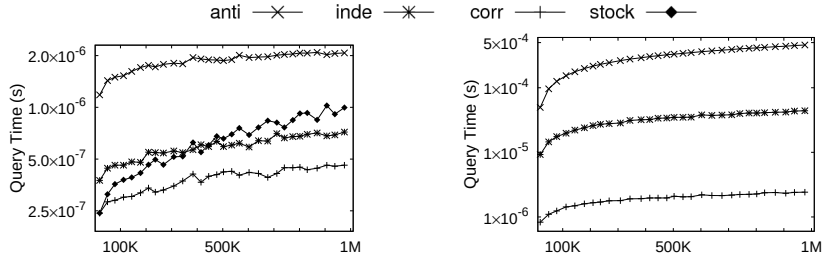


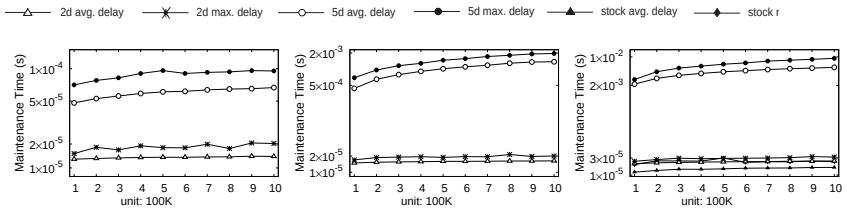
Fig. 4 q-sky vs pnN



(a) 2d (b) 5d

Fig. 5 Performance of pnN against Different n

6.2 Efficiency of Maintenance Techniques: pmnN



(a) correlated (b) independent (c) anti-correlated

Fig. 6 pmnN Performance against Different N

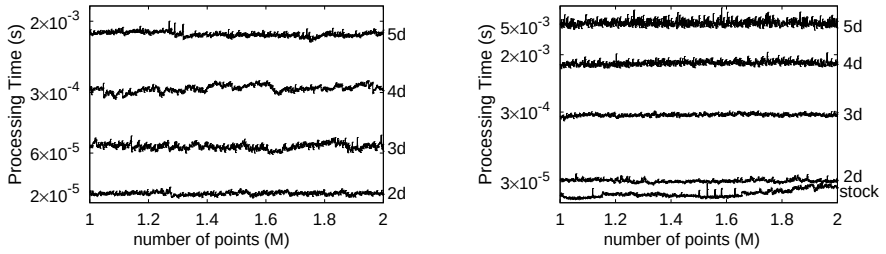
In this subsection, we report the efficiency of the maintenance techniques for $\mathcal{P}n$-*of*-N queries over uncertain sliding windows. The dimensionality is fixed to 2 and 5, and N varies from 100k to 1M. For each of the space dimensions, we generate three data streams where the spatial distribution follows correlated, independent, and anti-correlated, respectively. The real data stream *stock* is also studied along with the anti-correlated data stream. In Figure 6, we report the maximum and average cost of processing one element against different N values. As shown, when dimensionality is high the maintenance time per element is also high as the size of candidate set $R_{N,q}$ is larger. As illustrated, the correlated data has the best performance and the anti-correlated data is the most challenging. This is because

correlated data leads to the smallest size of $R_{N,q}$ on average while the anti-correlated data set generates the largest $R_{N,q}$ on average. The results demonstrate that our continuous maintenance techniques are very efficient and can support on-line update against a very rapid data stream. For the most challenging case of 5d anti-correlated data set, in average pmnN can handle the stream speed of about 500 elements per second.

6.3 Scalability Evaluation

We evaluate the scalability for the proposed techniques to handle a number of $\mathcal{P}$*n-of-N* queries regarding various parameters. We choose $N = 10^6$, and limit the data set size to 2×10^6.For each space dimension d ($1 \leq d \leq 5$), we generate two streams (independent and anti-correlated) with 2×10^6 data elements. The *stock* data is reported along with anti-correlated data.

The scalability of our algorithms is recorded as follows. We randomly generate 2×10^6 $\mathcal{P}$*n-of-N* queries and randomly assign them among the most recent 1M elements. Then, we run the pmnN algorithm to continuously maintain the data structures and run pnN for processing $\mathcal{P}$*n-of-N* queries. We record the processing time between two consecutive data elements which includes both the time of processing the queries and the time to maintain the data structures. Since such time is too short to be captured, we use average time for processing 1000 elements as the processing time. In Figure 7, we vary the number of points from 10^6 to 2×10^6. As illustrated, the proposed techniques could support queries over very rapid data streams with the arrival speed higher than 10K per second when dimensionality is lower (2 and 3). When dimensionality is higher (4 and 5), our techniques could still support data stream with a medium arrival speed at 200 elements per second even in the most challenging scenario of 5d anti correlated data set.



(a) independent (b) anti-correlated

Fig. 7 Overall Performance

We also report the impact of the expected occurrence probability (P_μ) in normal distribution and the probability threshold (q) on the scalability of $\mathcal{P}$*n-of-N* query processing. Figure 8 illustrates that the query processing techniques perform better with the increase of P_μ. This is because when the occurrence probabilities of uncertain elements are large, it is less likely for an element to be a probabilistic skyline point and thus the size of candidate set $R_{N,q}$ is smaller. Figure 9 shows that the processing time decreases with the increase of probability threshold q also because less elements are in the candidate set $R_{N,q}$.

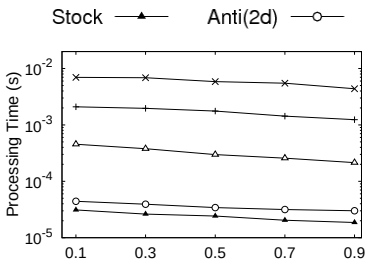


Fig. 8 Processing Time vs P_μ

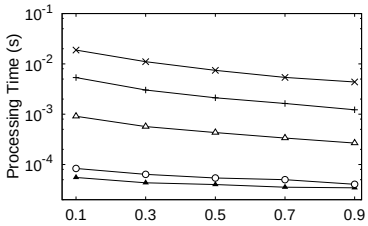
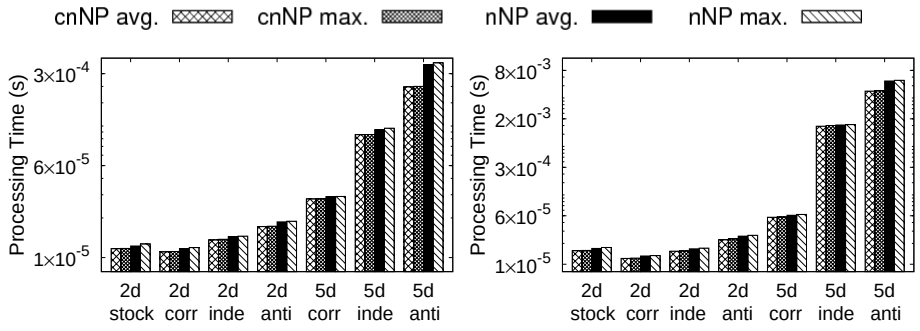


Fig. 9 Processing Time vs q

6.4 Evaluation of continuous query processing algorithm: pcnN

In this subsection we evaluate the performance of continuous query processing techniques - pcnN. To make a comparison, we also run our pnN algorithm once per new data item arrival to continuously process a $\mathcal{P}$*n-of-N* query. We use 2d and 5d data for the evaluation. We choose N = 10K and 1M. In the system, 20 $\mathcal{P}$*n-of-N* queries are generated such that 10 for N = 1M and 10 for N = 10K. For N = 10K (N = 1M), these 10 queries are with $n = i \times \frac{N}{10}$ (for $1 \leq i \leq 10$), respectively. We record the average delay (processing time) and maximum delay of an element, respectively. Note that a delay of an element e means the processing time involving processing e before processing next element; this includes the data structure maintenance costs and query processing costs. Again to record precisely such a delay per element, we use the average delay per 1000 elements instead. The performance of pcnN for continuous $\mathcal{P}$*n-of-N* queries is shown in Figure 10. As shown in the gure, both pnN and pcnN algorithms are quite efcient, while pcnN technique can support a data stream against medium arrival speed even for the most challenging 5d anti-correlated datasets.



(a) N=10K (b) N=1M

Fig. 10 Performance Evaluation of cnNP

7 Extensions

Techniques proposed in this paper can also be applied to other extensions of skyline queries over uncertain data streams under n-of-N model. In this section, we introduce several variations and briefly discuss the techniques.

7.1 Uncertain Object Model

In this paper we focus on the *existential uncertainty* of objects, namely, a probability value is associated with an object to indicate the likelihood of its existence. In some applications, each uncertain object may have several possible values (instances) and we call it the *uncertain object* model. Below we formally define the $\mathcal{P}n$*-of-N* in the uncertain object model and briefly discuss the techniques.

In uncertain object model, an *uncertain* object U is represented by a set of *instances* such that each instance $u \in U$ is a point in a d-dimensional numeric space $D = \{D_1, ..., D_d\}$ with the probability $P(u)$ to occur where $0 < P(u) \leq 1$ and $\sum_{u \in U} P(u) = 1$. Given a set of uncertain objects $\mathcal{U} = \{U_1, \cdots, U_n\}$, a *possible world* $W = \{u_1, \cdots, u_n\}$ is a set of instances with one instance from each uncertain object. The probability of W to appear is $Pr(W) = \prod_{i=1}^{n} p_{u_i}$. Let Ω be the set of all possible worlds, then $\sum_{W \in \Omega} Pr(W) = 1$.

We use $SKY(W)$ to denote the set of objects such that for each object $U \in SKY(W)$, U has an instance in the skyline of a possible world W. The probability that U appears in the skylines of the possible worlds is $P_{sky}(U) = \sum_{U \in SKY(W), W \in \Omega} P(W)$. $P_{sky}(U)$ is called the *skyline probability* of U.

Our framework still works for the uncertain object model based $\mathcal{P}n$*-of-N* queries. It could be verified that the candidate set for $\mathcal{U}$ is the set of objects with skyline probability no less than the given probability q. The probabilistic critical dominance relationship could be identified based on the techniques in Section 3.2. To manage the possibly large volume of instances for each uncertain object, an in memory R-tree may be built to index all of its instances.

7.2 Probabilistic Top-k Skyline Elements

Based on the definition of $\mathcal{P}n$*-of-N* queries, a probabilistic top-k query retrieves the k skyline elements with the highest skyline probability (but not smaller than q). In case there are less than k elements with probabilities not smaller than q, only these elements are output.

Our techniques could be directly applied to support probabilistic top-k skyline elements in the n-of-N model. After processing $\mathcal{P}n$*-of-N* queries, the skyline probability of each element e in the result set could be computed by simply scanning the critical dominance list (Section 4.1). The scanning is according to a decreasing order of elements in the critical dominance list, till the constraint of "most n elements" is met. The k elements with the largest skyline probabilities form the result set.

8 Related Work

Börzsönyi *et al* [11] first study the skyline operator in the context of databases and propose an SQL syntax for the skyline query. They also develop two computation techniques

based on *block-nested-loop* and *divide-and-conquer* paradigms, respectively. Another *block-nested-loop* based technique SFS (*sort-filter-skyline*) is proposed by Chomicki *et al* [12], which takes advantage of a pre-sorting step. SFS is then significantly improved by Godfrey *et al* [13]. The *progressive* paradigm that aims to output skyline points without scanning the whole dataset is firstly proposed by Tan *et al* [14]. It is supported by two auxiliary data structures, *bitmap* and *search tree*. Kossmann *et al* [15] present another progressive technique based on the nearest neighbor search technique. Papadias *et al* [16] develop a *branch-and-bound* algorithm (BBS) to progressively output skyline points based on R-trees with the guarantee of minimal I/O cost.

Skyline queries processing in exact data streams is investigated by Lin *et al* [6] following the n-of-N model. Tao *et al* [7] independently develop efficient techniques to compute sliding window skylines.

The skyline query processing on uncertain data is firstly approached by Pei *et al* [17] where *Bounding-pruning-refining* techniques are developed for efficient computation. Efficient pruning techniques are developed to reduce the search space for query processing. While [17] solves the case of probabilistic skyline computation with a pre-given threshold, [18] studies the problem of computing skyline probabilities for every object in the uncertain database. In [19], instead of a pregiven probability threshold, k uncertain objects from the data set with the highest skyline probabilities are retrieved. Stochastic skyline operators are proposed in [20, 21] to retain a minimum set of candidates for all ranking functions in the light of expected utility principles.

9 Conclusions

In this paper, we presented novel techniques for on-line skyline computation over the most recent n elements (for any $n \leq N$) in an uncertain data stream in a probability threshold fashion. Each element in the data stream is associated with an occurrence probability. We identify the minimum candidate set to maintain and propose efficient query processing and index maintaining techniques. Our experiment results demonstrated that the techniques can be used to process rapid data streams in lower dimensional spaces with the space dimension not greater than 5.

References

1. G. Cormode and M. Garofalakis, "Sketching probabilistic data streams," in *SIGMOD*, 2007.
2. T. S. Jayram, A. McGregor, S. Muthukrishan, and E. Vee, "Estimating statistical aggregrates on probabilistic data streams," in *PODS*, 2007.
3. C. Jin, K. Yi, L. Chen, J. X. Yu, and X. Lin, "Sliding-window top-k queries on uncertain streams," in *VLDB*, 2008.
4. X. Ding, X. Lian, L. Chen, and H. Jin, "Continuous monitoring of skylines over uncertain data streams," in *Information Sciences*, 2012.
5. W. Zhang, X. Lin, Y. Zhang, W. Wang, and J. X. Yu, "Probabilistic skyline operator over sliding windows," in *ICDE*, 2009.
6. X. Lin, Y. Yuan, W. Wang, and H. Lu, "Stabbing the skye: Efficient skyline computation over sliding windows," in *ICDE*, 2005.
7. Y. Tao and D. Papadias, "Maintaining sliding window skylines on data streams," in *TKDE*, 2006.
8. X. Lin, H. Lu, J. Xu, and J. X. Yu, "Continuously maintaining quantile summaries of the most recent n elements over a data stream." in *ICDE*, 2004, pp. 362–374.
9. B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom, "Models and issues in data stream systems." in *PODS*, 2002, pp. 1–16.

10. K. Mehlhorn, *Data Structures and Algorithms: 3. Multidimensional Searching and Computational Geometry*. Springer, 1984.
11. S. Börzsönyi, D. Kossmann, and K. Stocker, "The skyline operator." in *ICDE*, 2001, pp. 421–430.
12. J. Chomicki, P. Godfrey, J. Gryz, and D. Liang, "Skyline with presorting." in *ICDE*, 2003, pp. 717–816.
13. P. Godfrey, R. Shipley, and J. Gryz, "Maximal vector computation in large data sets," in *VLDB*, 2005.
14. K.-L. Tan, P.-K. Eng, and B. C. Ooi, "Efficient progressive skyline computation." in *VLDB*, 2001, pp. 301–310.
15. D. Kossmann, F. Ramsak, and S. Rost, "Shooting stars in the sky: An online algorithm for skyline queries." in *VLDB*, 2002, pp. 275–286.
16. D. Papadias, Y. Tao, G. Fu, and B. Seeger, "An optimal and progressive algorithm for skyline queries." in *SIGMOD*, 2003, pp. 467–478.
17. J. Pei, B. Jiang, X. Lin, and Y. Yuan, "Probabilistic skylines on uncertain data," in *VLDB 2007*.
18. M. J. Atallah and Y. Qi, "Computing all skyline probabilities for uncertain data," in *PODS*, 2009.
19. Y. Zhang, W. Zhang, X. Lin, B. Jiang, and J. Pei, "Ranking uncertain sky: the probabilistic top-k skyline operator." in *Information Systems*, 2011.
20. X. Lin, Y. Zhang, W. Zhang, and M. A. Cheema, "Stochastic skyline operator," in *ICDE 2011*.
21. W. Zhang, X. Lin, Y. Zhang, M. A. Cheema, and Q. Zhang, "Stochastic skylines," in *TODS*, 2012.