Probabilistic k-Skyband Operator over Sliding Windows*

Xing Feng[1], Wenjie Zhang[2], Xiang Zhao[2], Ying Zhang[2], and Yunjun Gao[1]

[1] Zhejiang University
[2] University of New South Wales
{xingfeng,zhangw,xzhao,yingz}@cse.unsw.edu.au, gaoyj@zju.edu.cn

Abstract. Given a set of data elements $\mathcal{D}$ in a d-dimensional space, a k-skyband query reports the set of elements which are dominated by at most $k-1$ other elements in $\mathcal{D}$. k-skyband query is a fundamental query type in data analyzing as it keeps a minimum candidate set for all top-k ranking queries where the ranking functions are monotonic. In this paper, we study the problem of k-skyband over uncertain data streams following the possible world semantics where each data element is associated with an occurrence probability. Firstly, a dynamic programming based algorithm is proposed to identify k-skyband results for a given set of uncertain elements regarding a pre-specified probability threshold. Secondly, we characterize the minimum set of elements to be kept in the sliding window to guarantee correct computing of k-skyband. Thirdly, efficient update techniques based on R-tree structures are developed to handle frequent updates of the elements over the sliding window. Extensive empirical studies demonstrate the efficiency and effectiveness of our techniques.

1 Introduction

In a d dimensional numerical space, we say an element e dominates another element e' if e is not worse than e' in every dimension and is better than e' in at least one dimension. Given a dataset $\mathcal{D}$, the k-skyband of $\mathcal{D}$ contains the set of points which are dominated by at most $k-1$ other points. k-skyband is a useful tool in data analyzing as it retrieves the objects which are definitely worse than (i.e., dominated by) at most $k-1$ other objects. The concept of k-skyband is also closely related to the following two concepts, *skyline* and *top-k ranking* queries. Given a dataset $\mathcal{D}$, the *skyline* of $\mathcal{D}$ contains all elements which are not dominated by any other elements from $\mathcal{D}$. Clearly, skyline is a special case of k-skyband query where $k = 1$. Given a dataset $\mathcal{D}$ and a ranking function f, a top-k query retrieves the k elements in $\mathcal{D}$ with the highest scores according to f. An important property of k-skyband is that it keeps a minimum candidate set to answer multiple ad-hoc top-k queries given that the ranking functions of these queries are monotonic. Thus, keeping k-skyband of a dataset is sufficient to answer all top-k

* Wenjie Zhang was partially supported by ARC DE120102144 and DP120104168. Ying Zhang was partially supported by DP110104880 and UNSW ECR grant PSE1799. Yunjun Gao was supported in part by NSFC 61003049 and the Fundamental Research Funds for the Central Universities under Grant 2012QNA5018.

J. Wang et al. (Eds.): WAIM 2013, LNCS 7923, pp. 190–202, 2013.

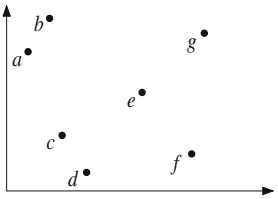


Fig. 1. Skyband Example

ranking queries with monotonic ranking functions. In Figure 1, the skyline elements (1-skyband) are $\{a, c, d\}$ as neither of them is dominated by other elements; 2-skyband consists $\{a, c, d, b, f\}$ which are dominated by at most 1 other object. Note that without loss of generality, we assume smaller values are preferred in the paper.

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 arises in these applications due to various reasons including data randomness and incompleteness, delay or loss in data transfer, limitation of equipments, privacy preservation, etc. In many scenarios of such applications, uncertain data is collected in a streaming fashion. Uncertain streaming data computation has been studied and the existing work mainly focuses on aggregate queries, top-k queries, skyline queries, etc [3,6,15,16]. In this paper, we will investigate the problem of efficient skyband computation over uncertain streaming data where each data element has a probability to occur.

Skyband computation over uncertain streaming data has many applications. For instance, in an on-line shopping system products are evaluated in various aspects such as *price* and *condition*. In addition, each seller is associated with a "trustability" value which is derived from customers' feedback on the seller's product quality, delivery handling, etc. This "trustability" value can also be regarded as 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. A customer may want to select a product, say laptops, according to multi-criteria based ranking, such as low price, good condition, etc, and require that the retrieved laptop cannot be definitely worse than (i.e., dominated by) $k-1$ other laptops where k is specified by the user. In Table 1, L_1 and L_4 are skyline points, $\{L_1, L_4, L_2, L_3\}$ form 2-skybands. Nevertheless, we should also consider that L_1 is posted long time ago; L_4 is better than (dominates) L_3 but the trustability of the seller of L_4 is low.

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

The challenges for skyband computation over uncertain sliding windows are threefold. Firstly, computing probabilistic k-skyband for a given set of uncertain elements is costly since naively we need to enumerate all the possible worlds which is exponential to the number of total elements. Secondly, a critical requirement in data stream computation is to have small space so it is important to identify a candidate set with minimum size for correct k-skyband computation in the sliding window. Thirdly, due to high arriving speed and high data volume, it is essential to develop time efficient techniques to handle frequent updates (i.e., insertions and deletions) in the sliding window.

To the best of our knowledge, this paper is the first to study the problem of probabilistic k-skyband operator over uncertain sliding windows. Our main contributions are summarized as follows.

- We formally define the probabilistic k-skyband operator in uncertain sliding windows following the possible world semantics in a probability threshold fashion.
- A dynamic programming based algorithm is proposed to efficiently retrieve k-skyband elements for a given set of uncertain elements. The algorithm runs in $O(nk)$ time for an element a where n is the number of elements dominating a. An optimization technique is further proposed to reduce the time complexity to $O(k)$ when a new element is added to the dominating element set of a.
- We characterize the minimum information needed in continuously computing k-skyband regarding a sliding window and show that the expected size of the minimum candidate set is $kln^d\frac{N}{k}$ when the data set follows uniform distribution.
- We develop novel, efficient, incremental techniques to continuously compute probabilistic k-skyband over sliding windows.
- Space and time efficiency are demonstrated by an extensive empirical study on both real and synthetic datasets.

The rest of the paper is organized as follows. In Section 2, we formally define the problem of sliding-window k-skyband computation and provide some necessary background information. Section 3 presents the theoretical foundations. We develop techniques for k-skyband probabilities computation in Section 4 and continuous updates of probabilistic k-skyband queries in Section 5. Results of comprehensive performance evaluation are presented in Section 6. Section 7 summaries related work and Section 8 concludes the paper.

2 Background Information

We use DS to represent a sequence (stream) of data elements in a d-dimensional numeric space such that each element a 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. For two 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 *k-skyband* consists of all points which are dominated by at most $k-1$ other elements.

2.1 Problem Definition

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 $SKYBAND(W)$ to denote the set of elements in W that form the k-skyband of W. The probability that an element a appears in the k-skyband of the possible worlds is $P_{skyband}(a) = \sum_{a \in SKYBAND(W), W \in \Omega} P(W)$. $P_{skyband}(a)$ is called the *k-skyband probability* of a.

In many applications, a data stream DS is *append-only* [7,9,14]; that is, there is no deletion of data element involved. In this paper, we study the k-skyband computation problem restricted to the append-only data stream model. In a data stream, elements are positioned according to their relative arrival ordering and labelled by integers. Note that the position/label $\kappa(a)$ means that the element a arrives $\kappa(a)$th in the data stream.

Problem Statement. In this paper, we study the problem of efficiently retrieving k-skyband elements from the most recent N elements, seen so far, with the k-skyband probabilities not smaller than a given probability threshold q ($0 < q \leq 1$).

2.2 Preliminaries

An Assisted Probability. Let DS_N denote the most recent N tuples. For each element a, we use $P_{new}(a)$ to denote the k-skyband probability of a restricted to the elements newer than a only.

Example 1. Suppose 7 uncertain elements a, b, c, d, e, f, g arrive according to alphabetic order. For an element c, $P_{skyband}(c)$ denote the probability that c is dominated by at most $k-1$ elements among a, b, d, e, f, g. On the other hand, $P_{new}(c)$ is the probability that c is dominated by at most $k-1$ elements among d, e, f, g (i.e., newer than c) only.

3 Framework

Given a probability threshold q and a sliding window with length N, Alogrithm1 shows the framework for continuously processing probabilistic k-skyband queries where a_{old} is the oldest element in the current window DS_N. Functions $insert(a_{new})$ and $remove(a_{old})$ will be elaborated in the following sections.

Algorithm 1. Probabilistic k-skyband Computation over a Sliding Window

```
1 while a new element a_new arrives do
2     if |S| ≤ N then
3         insert(a_new);
4     else
5         remove(a_old);
6         insert(a_new);
```

3.1 Using $S_{N,q}$ only

Let $S_{N,q}$ denote the set of elements from DS_N with their P_{new} values not smaller than q; that is,

$$S_{N,q} = \{\alpha | \alpha \in DS_N \ \& \ P_{new}(\alpha) \geq q\} \quad (1)$$

A critical requirement in data stream computation is to have small memory space and fast computation. In our algorithms, instead of keeping all recent N elements DS_N, we propose to keep only $S_{N,q}$ which is shown to be logarithmic in size regarding N in the empirical study. Next, we show the correctness of using $S_{N,q}$ only in the computation.

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

The proofs of Theorem 1 is lengthy and we omit it here due to space limits. The correctness of the theorem is based on the following properties of $S_{N,q}$.

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

Size of $S_{N,q}$. Elements in the candidate set $S_{N,q}$ can be regarded as the k-skyband results in the $d+1$ dimensional space by including *time* as an additional dimension. Thus, with the assumption that all points follow the uniform distribution, the expected size of $S_{N,q}$ is $kln^d\frac{N}{k}$.

4 k-Skyband Probability Calculation

For a given uncertain element a in the sliding window, assume that a is dominated by n other elements $u_1, ..., u_n$. Note that $u_1, ..., u_n$ could be efficiently retrieved using any existing dominance reporting algorithms based on R-trees [9]. Clearly, the k-skyband probability of a is the sum of probabilities of all possible worlds in which a is dominated by at most $k-1$ other elements. Namely, at most $k-1$ elements from $u_1, ..., u_n$ appear since the occurrence of elements not dominating a does not affect the computation of $P_{skyband}(a)$.

Next we present a dynamic programming based algorithm for calculating $P_{skyband}(a)$. Assume that the elements $u_1, ..., u_n$ are sorted according to an arbitrary order, we build a matrix M with size $(n+1) \times (n+1)$ as follows. Let $m_{i,j}$ denote the probability that exactly i elements out of the first j elements happen. The following equation is immediate.

$$m_{i,j} = m_{i-1,j-1} \times P(u_j) + m_{i,j-1} \times (1 - P(u_j)) \tag{2}$$

where $P(u_j)$ is the occurrence probability of element u_j. In the initializing phase, $m_{0,j}$ (for $1 \leq j \leq n$) equals $(1 - P(u_1)) \times ... \times (1 - (P(u_j)))$, namely none of the first i elements occurs; $m_{i,i}$ (for $1 \leq i \leq n$) equals $P(u_1) \times ... \times P(u_i)$ meaning that all first i elements occur. The construction of the matrix is summarized in the following equation.

$$m_{i,j} = \begin{cases} (1 - P(u_1)) \times ... \times (1 - (P(u_i))) & j = 0 \\ P(u_1) \times ... \times P(u_i) & i = j \\ m_{i-1,j-1} \times P(u_j) + m_{i,j-1} \times (1 - P(u_j)) & otherwise \end{cases}$$

Algorithm 2. Skyband Probability Calculation

Input : $a, u_1, ..., u_n, k, q$
Output : $P_{skyband}(a)$

```
if n ≤ k − 1 then
    return 1;
m_{0,0} = 1;
for 1 ≤ i ≤ n do
    m_{0,i} = m_{0,i−1} × (1 − P(u_i));
    m_{i,i} = m_{i−1,i−1} × P(u_i);
for 1 ≤ i ≤ k − 1 do
    j = i + 1;
    while j ≤ n do
        m_{i,j} = m_{i−1,j−1} × P(j) + m_{i,j−1} × (1 − P(j));
return Σ_{i=0}^{k−1} m_{i,n};
```

The algorithm for computing $P_{skyband}(a)$ is reported in Algorithm 2.

Time Complexity. In building the matrix, the sum of values $m_{i,n}$ ($0 \leq i \leq k-1$) captures the probability that at most $k-1$ elements occur among the n elements dominating a. Thus, we can stop once $m_{k,n}$ is retrieved, leading to the time complexity $O(kn)$.

Optimization. Since the elements dominating a are sorted in an arbitrary order, if a new element is inserted into the dominating set of a, we do not need to rebuild the whole matrix. Instead, the existing $n \times n$ matrix can be expanded to an $(n+1) \times (n+1)$ matrix and we retrieve the probability that at most k elements occur out of $n+1$ elements from the new matrix. The time complexity in this case is reduced to $O(k)$.

5 Continuously Processing Probabilistic k-Skyband Queries

A naive execution of Algorithm1 is with every update of the sliding window (i.e., arrival of a_{new} and expiration of a_{old}), for every element a in $S_{N,q}$, we recompute its k-skyband probability $P_{skyband}(a)$ following Algorithm 2 regarding the updates, and report elements with $P_{skyband}(a) \geq q$. In this section, we present novel aggregate information enhanced, R-tree based algorithms to efficiently handle the updates.

5.1 Aggregate R-Tree

We monitor the following two probability values in the continuous processing, P_{new} and $P_{skyband}$. Once $P_{new}(a)$ for an element a falls below the probability threshold q, as shown in Section 3, a could be safely removed. $P_{skyband}(a)$ for an element a decides its qualification as a k-skyband result. Specifically, we continuously keep two R-trees, R_1 indexes all candidate elements in $S_{N,q}$ which are not the k-skyband results in the sliding window, where R_2 indexes all current k-skyband results. Clearly, R_1 and R_2 together keep all candidate elements from $S_{N,q}$.

Furthermore, for each element a in R_1 and R_2. Denote the matrix to compute $P_{skyband}(a)$ in Section 4 as $M_{skyband}$. We maintain the last column of $M_{skyband}$, denoted as $\lambda_{M_{skyband}}(a)$ so that when a new element is added to the element list dominating a, $P_{skyband}$ can be updated by using $\lambda_{M_{skyband}}(a)$ only in $O(k)$ time. Similarly, denote the matrix for calculating $P_{new}(a)$ as M_{new}. Its last column $\lambda_{M_{new}}(a)$ is kept to update $P_{new}(a)$ in $O(k)$ time once a new element is added to the dominance list.

5.2 Handling Insertion of a_{new}

When a new tuple a_{new} arrives in the sliding window, it stays in $S_{N,q}$ because it is not dominated by any element newer than it and thus $P_{new}(a_{new}) = 1$. After computing the $P_{skyband}$ value of a_{new}, we may decide if it is a candidate element (to insert into R_1),or a k-skyband result (to insert into R_2). Besides, the following effects are invoked by the arrival of a_{new}.

Qualification as a Candidate Element. We check the qualification of an element to be in the candidate set for both R_1 and R_2. If an element a is dominated by a_{new} and the updated P_{new} value becomes below the probability threshold q, a is no longer a candidate element and will be deleted from R_1 or R_2.

Qualification as a Probabilistic k-skyband Result. We check the qualification of an element as a probabilistic k-skyband result for elements in R_2 only. If an element a in R_2 survives the qualification test and $P_{skyband}(a)$ falls below q, it will be moved from R_2 to R_1.

Insert a_{new}. $P_{skyband}$ value of a_{new} will be computed using the dynamic programming algorithm in Section 4. If $P_{skyband}(a_{new}) \leq q$, a_{new} is inserted into R_2; otherwise a_{new} is inserted into R_1.

Algorithm 3. Insertion of (a_{new})

```
1  retrieve the element set D_{R_1} in R_1 dominated by a_new;
2  for each element e in D_{R_1} do
3      update P_new(e) using λ_{M_new}(e);
4      if P_new(e) < q then
5          delete e from R_1;
6  retrieve the element set D_{R_2} in R_2 dominated by a_new;
7  for each element e in D_{R_2} do
8      update P_new(e) using λ_{M_new}(e);
9      if P_new(e) < q then
10         delete e from R_1;
11     else
12         update P_skyband(e) using λ_{M_skyband}(e);
13         if P_skyband(e) < q then
14             move e from R_2 to R_1;
```

Algorithm 3 reports the steps to handle the insertion of a_{new}. Lines 1 to 5 handles the updates on R_1 where we need to consider the qualification as a candidate element only, while lines 6 to 14 tackles updates on R_2 where we need to handle both candidate and k-skyband result qualification. Note that retrieving the element set which are dominated by a_{new} can be efficiently implemented using existing techniques [9].

5.3 Handling Removal of a_{old}

With the expiration of a_{old}, we check the elements in R_1 and R_2 which are dominated by a_{old} and update their $P_{skyband}$ values since the $P_{skyband}$ values for these element will increase. For every element in R_1 and R_2 which is dominated by a_{old}, its $P_{skyband}$ value will be recomputed using the dynamic programming algorithm in Section 4. If the updated $P_{skyband}$ value for an element in R_1 becomes larger than or equal to q, it will be moved from R_1 to R_2. Removing a_{old} is straightforward since we only need to delete it from R_1 or R_2. Algorithm 4 depicts the key steps in handling the removal of a_{old}. Lines 2 to 6 handles the updates on R_1 and Lines 7 to 9 handles the updates on R_2.

Algorithm 4. Removal of (a_{old})

```
1 delete a_old from R_1 or R_2;
2 retrieve the elements set D_{R_1} in R_1 dominated by a_old;
3 for each element e in D_{R_1} do
4     recompute P_skyband(e);
5     if P_skyband(e) ≥ q then
6         move e from R_1 to R_2;
7 retrieve the elements set D_{R_2} in R_2 dominated by a_old;
8 for each element e in D_{R_2} do
9     recompute P_skyband(e);
```

5.4 Continuously Processing Probabilistic k-Skyband Query

As introduced above, R-tree R_2 keeps the probabilistic k-skyband results for the current sliding window and is updated along the sliding of the window (i.e., new elements arrival and old elements expiration). So at every timestamp, scanning using breadth first search over R_2 and outputing all elements in R_2 provides the probabilistic k-skyband result for the current window.

6 Performance Evaluation

In this section, we only evaluate our techniques since this is the first paper studying the problem of probabilistic k-skyband computation over sliding windows.

All algorithms are implemented in C++ and compiled by GNU GCC. Experiments are conducted on PCs with Intel Xeon 2.4GHz dual CPU and 4G memory under Debian Linux. Our experiments are conducted on both real and synthetic datasets.

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 *Dec 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. We randomly assign a probability value to each transaction; that is, probability values follows *uniform* distribution. Elements' arrival order is based on their transaction time.
Synthetic datasets are generated as follows. We first use the methodologies in [2] to generate 2 million data elements with the dimensionality from 2 to 5 and the spatial location of data elements follow two kinds of distributions, *independent* and *anti-correlated*. Then, we use two models, *uniform* or *normal* distribution, to randomly assign occurrence probability of each element to make them be uncertain. In *uniform* distribution, the occurrences probability of each element takes a random value between 0 and 1, while in the *normal* distribution, the mean value P_μ varies from 0.1 to 0.9 and variance is set 10. We assign a random order for elements' arrival in a data stream.

Table 2 summarizes parameters and corresponding default values. **In our experiments, all parameters take default values unless otherwise specified.**

Table 2. System Parameters

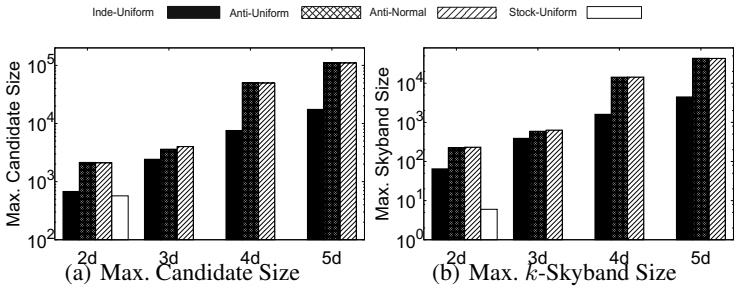
Notation	Definition (Default Values)
n	Number of points in the dataset (2M)
N	Sliding Window size (400K)
d	Dimensionality of the of the dataset (3)
D	Dataset (Anti)
D_P	Probabilistic distribution of appearance (*uniform*)
P_μ	expected appearance probability (0.5)
q	probabilistic threshold (0.3)
k	k-skyband width (4)

6.1 Evaluate Space Efficiency

We evaluate the space usage in terms of the number of uncertain elements kept in $S_{N,q}$ against different settings. As this number may change as the window slides, we record the maximal value over the whole stream. Meanwhile, we also keep the maximal number of k-skyband.

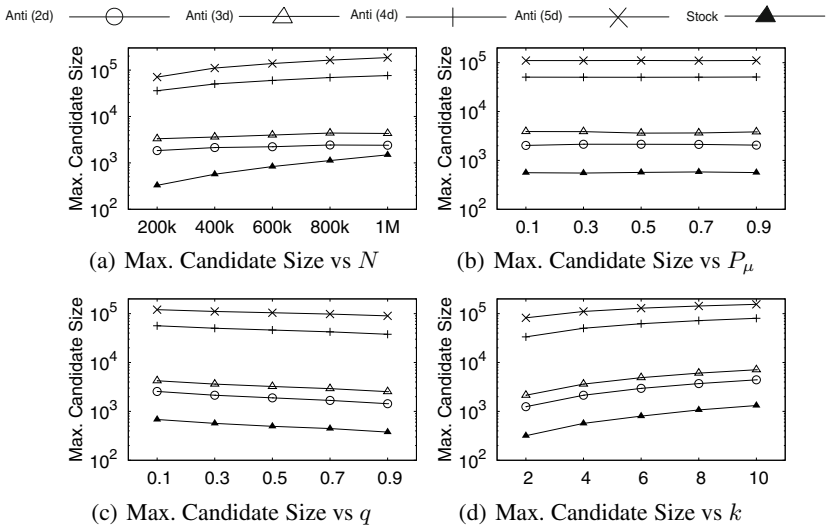
The first set of experiments is reported in Figure 2 where 4 datasets are used: Inde-Uniform (Independent distribution for spatial locations and Uniform distribution for occurrence probability values), Anti-Uniform, Anti-Normal, and Stock-Uniform. We record the maximum sizes of $S_{N,q}$ and k-skyband. It is shown that very small portion of the 2-dimensional dataset needs to be kept. Although this proportion increases with the dimensionality rapidly, our algorithm can still achieve a 72% space saving even in the worst case, 5 dimensional *anti-correlated* data. Size of k-skyband is much smaller than that of candidates. Since the *anti-correlated* dataset is the most challenging, it will be employed as the default dataset in the following.

The second set of experiment evaluates the impact of sliding window size N on the space efficiency. As depicted in Figure 3(a), the space usage is sensitive towards the increment of window size.



(a) Max. Candidate Size (b) Max. k-Skyband Size

Fig. 2. Space Usage vs Diff. Data set



(a) Max. Candidate Size vs N (b) Max. Candidate Size vs P_μ

(c) Max. Candidate Size vs q (d) Max. Candidate Size vs k

Fig. 3. Space Usage

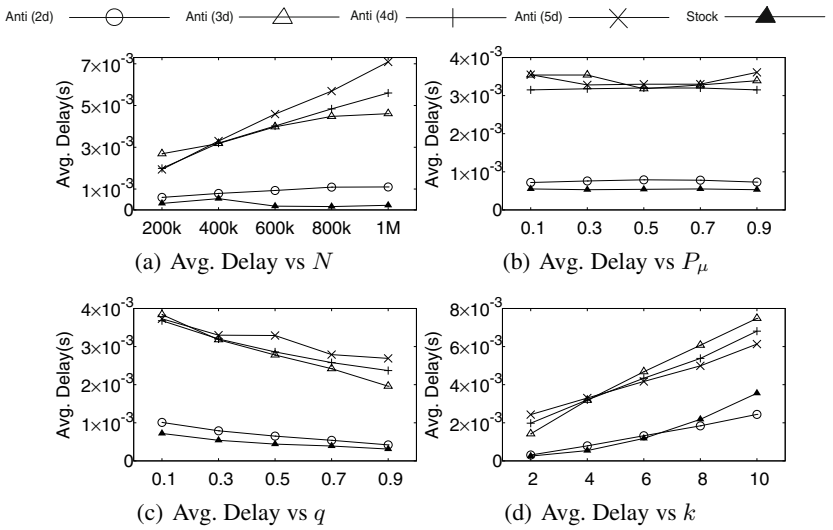
Figure 3(b) reports the impact of occurrence probability distribution against the space usage on different datasets. It demonstrates that there's no obvious relationship between candidate size and expected appearance probability.

Figure 3(c) reports the effect of probabilistic threshold q on space efficiency. The threshold probability q varies from 0.1 to 0.9. As expected, candidate set size drops as q increases.

Figure 3(d) reports the effect of k-skyband width k on space efficiency. As expected, candidate set size increases as k increases.

6.2 Evaluation Time Efficiency

We evaluate the time efficiency of our continuous query processing techniques. We first compare our algorithm with a trivial algorithm based on dynamic programming to



(a) Avg. Delay vs N (b) Avg. Delay vs P_μ

(c) Avg. Delay vs q (d) Avg. Delay vs k

Fig. 4. Time Efficiency

compute $P_{skyband}$. We find it is too late since it recalculates $P_{skyband}$ value of every element in every sliding window. Thus, we exclude the trivial algorithm from further evaluation.

Since the processing time of one element is too short to capture precisely, we record the average time for each batch of 1K elements to estimate the delay per element.

Figure 4(a) evaluates the system scalability towards the size of the sliding window. The performance of our algorithm decreases as the size of sliding window increases. This is because the candidate size increases with N, as reported in Figure 3(a).

Figure 4(b) evaluates the impact of occurrence probability distribution on time efficiency of our algorithm where normal distribution is used for probability values. Similar to Figure 3(b), there is no obvious relationship.

Figure 4(c) evaluates the effect of probability threshold q. Since both size of candidate set and k-skyband objects set are small when q is large as depicted in Figure 3(c), our algorithm is more efficient when q increases.

The last experiment evaluates the impact of k-skyband width k on time efficiency. Results are reported in Figures 4(d). As expected, Figure 4(d) shows that processing cost increases when k increases.

6.3 Summary

As a short summary, our performance evaluation indicates that we only need to keep a small portion of stream objects in order to compute the probabilistic k-skyband over sliding windows. Moreover, our continuous query processing algorithms are very efficient and can support data streams with high speed for 2d and 3d datasets. Even for the most challenging data distribution, *anti-correlated*, we can still support the data stream with medium speed of more than 140 elements per second when dimensionality is 5.

7 Related Work

We review related work in two aspects, skyband and uncertain data streams. To the best of our knowledge, this paper is the first one to address the problem of k-skyband queries on uncertain data streams.

The concept of k-skyband is first proposed in [12]. [11] shows that it suffices to keep the k-Skyband of the data set to provide the top-k ranked queries for the monotone preference function f. Reverse k-skyband query is recently studied in [10].

Aggregates over uncertain data streams have been studied in [3,5,6]. Problems such as clustering uncertain data stream [1], frequent items retrieval in probabilistic data streams [15], and sliding window top-k queries on uncertain streams [7] are also investigated. The skyline query processing on uncertain data is firstly approached by Pei *et al* [13] where *Bounding-pruning-refining* techniques are developed for efficient computation. Lian *et al* [8] combine reverse skylines [4] with uncertain semantics and model the *probabilistic reverse skyline* query in both monochromatic and bichromatic fashion. Efficient pruning techniques are developed to reduce the search space for query processing. Probabilistic skyline processing over uncertain sliding windows are investigated in [16].

8 Conclusion

In this paper, we investigate the problem of probabilistic k-skyband query processing over sliding windows in a probability threshold fashion. We firstly characterize the minimum information to be kept for correct probabilistic k-skyband computation. Then a novel dynamic programming based algorithm is proposed to compute the k-skyband probabilities for a given uncertain element. Efficient, aggregate information enhanced, incremental updating techniques are developed to handle frequent insertions and deletions in the streaming environment. Time and space efficiency of our proposed techniques are demonstrated in a comprehensive empirical study over both real and synthetic datasets.

References

1. Aggarwal, C.C., Yu, P.S.: A framework for clustering uncertain data streams. In: ICDE 2008 (2008)
2. Borzsonyi, S., Kossmann, D., Stocker, K.: The skyline operator. In: ICDE 2001 (2001)
3. Cormode, G., Garofalakis, M.: Sketching probabilistic data streams. In: SIGMOD 2007 (2007)
4. Dellis, E., Seeger, B.: Efficient computation of reverse skyline queries. In: VLDB 2007 (2007)
5. Jayram, T., Kale, S., Vee, E.: Efficient aggregation algorithms for probabilistic data. In: SODA 2007 (2007)
6. Jayram, T.S., McGregor, A., Muthukrishan, S., Vee, E.: Estimating statistical aggregrates on probabilistic data streams. In: PODS 2007 (2007)
7. Jin, C., Yi, K., Chen, L., Yu, J.X., Lin, X.: Sliding-window top-k queries on uncertain streams. In: VLDB 2008 (2008)

8. Lian, X., Chen, L.: Monochromatic and bichromatic reverse skyline search over uncertain databases. In: SIGMOD 2008 (2008)
9. Lin, X., Yuan, Y., Wang, W., Lu, H.: Stabbing the sky: Efficient skyline computation over sliding windows. In: ICDE 2005 (2005)
10. Liu, Q., Gao, Y., Chen, G., Li, Q., Jiang, T.: On efficient reverse *k*-skyband query processing. In: Lee, S.-G., Peng, Z., Zhou, X., Moon, Y.-S., Unland, R., Yoo, J. (eds.) DASFAA 2012, Part I. LNCS, vol. 7238, pp. 544–559. Springer, Heidelberg (2012)
11. Mouratidis, K., Bakiras, S., Papadias, D.: Continuous monitoring of top-k queries over sliding windows. In: SIGMOD 2006 (2006)
12. Papadias, D., Tao, Y., Fu, G., Seeger, B.: An optimal progressive algorithm for skyline queries. In: SIGMOD 2003 (2003)
13. Pei, J., Jiang, B., Lin, X., Yuan, Y.: Probabilistic skylines on uncertain data. In: VLDB 2007 (2007)
14. Tao, Y., Papadias, D.: Maintaining sliding window skylines on data streams. In: TKDE 2006 (2006)
15. Zhang, Q., Li, F., Yi, K.: Finding frequent items in probabilistic data. In: SIGMOD 2008 (2008)
16. Zhang, W., Lin, X., Zhang, Y., Wang, W., Yu, J.X.: Probabilistic skyline operator over sliding windows. In: Information Systems 2012 (2012)