Information Systems 36 (2011) 898–915

Contents lists available at ScienceDirect

Information Systems

journal homepage: www.elsevier.com/locate/infosys

Ranking uncertain sky: The probabilistic top-k skyline operator

Ying Zhang [a,*], Wenjie Zhang [a], Xuemin Lin [a], Bin Jiang [b], Jian Pei [b]

[a] The University of New South Wale & NICTA, Australia
[b] Simon Fraser University, Canada

ARTICLE INFO

Article history:
Received 16 October 2008
Received in revised form
2 September 2010
Accepted 23 March 2011
Recommended by: P. O'Neil
Available online 31 March 2011

Keywords:
Skyline
Uncertain
Top-k

ABSTRACT

Many recent applications involve processing and analyzing uncertain data. In this paper, we combine the feature of top-k objects with that of skyline to model the problem of top-k skyline objects against uncertain data. The problem of efficiently computing top-k skyline objects on large uncertain datasets is challenging in both discrete and continuous cases. In this paper, firstly an efficient exact algorithm for computing the top-k skyline objects is developed for discrete cases. To address applications where each object may have a massive set of instances or a continuous probability density function, we also develop an efficient randomized algorithm with an ϵ-approximation guarantee. Moreover, our algorithms can be immediately extended to efficiently compute p-skyline; that is, retrieving the uncertain objects with skyline probabilities above a given threshold. Our extensive experiments on synthetic and real data demonstrate the efficiency of both algorithms and the randomized algorithm is highly accurate. They also show that our techniques significantly outperform the existing techniques for computing p-skyline.

1. Introduction

Driven by many recent applications such as environmental surveillance, market analysis, quantitative economics research, WWW, and large sensor networks, a rapidly growing amount of research has been dedicated to managing uncertain data (see Section 8 for a brief review). Uncertainty is inherent in such applications because of various factors including data randomness and incompleteness, limitations of measuring equipments, delayed data updates, etc. As more and more uncertain data accumulated, it is highly desirable to conduct advanced data analysis over uncertain data.

Skyline analysis (e.g., [2,3,15,19,24,28,34,38]) has been demonstrated very useful in multi-criteria decision making applications. In a multi-dimensional space where a preference order is given in each dimension (e.g., low price and high quality are preferred in dimensions price and quality), a point u_1 *dominates* another point u_2 if u_1 is not worse than u_2 in every dimension, and is better than u_2 in at least one dimension according to the preference. Point u is a *skyline* point if there is no any other point u' such that u' dominates u. Given a set of certain points, the skyline consisting of all skyline points presents all best possible tradeoffs among different user preferences—modeled by monotonic scoring functions.

Skyline analysis is also meaningful on uncertain data. To motivate our study, let us consider an example. Suppose that the service performance of a realtor regarding a property sold by her/him is evaluated against the two aspects: the percentage of the actual price increments against the reserve price (dimension P)—the higher the better, and the service quality ranked by the property owner (dimension E) where scores 1–5 are given for ranking purposes—the lower the better. The performance of a realtor may fluctuate from one sale to another sale due to various reasons. Therefore, evaluating a realtor to

* Corresponding author.
E-mail addresses: yingz@cse.unsw.edu.au (Y. Zhang), zhangw@cse.unsw.edu.au (W. Zhang), lxue@cse.unsw.edu.au (X. Lin), bjiang@cs.sfu.ca (B. Jiang), jpei@cs.sfu.ca (J. Pei).

0306-4379/$ - see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.is.2011.03.008

reflect the statistic distribution of her sale records is an effective way to predict her performance. For this purpose, the performance of a realtor may be modeled as an uncertain object in the two-dimensional space (P, E), and each successful sale record can be viewed as an instance of the uncertain object.

There can be a large number of realtors doing business in an area. Therefore, a new customer (property owner) often likes to receive recommendations of realtors who are good in both aspects—high value of P and lower value of E. Unfortunately, in practice it is often impossible for one realtor to dominate all other realtors in both aspects against all sale records. Thus, skyline analysis against uncertain data makes sense here.

While skyline on certain data is well defined, finding skyline of realtors as uncertain objects is not straightforward. Consider two realtors A and B and their instances in Fig. 1 where we record multiplicative inverse of the price incremental percentage on dimension P and assume that coordinate values are all positive without loss of generality. Instances a_1 and a_2 of A dominate instance b_2 of B. Instance b_1 of B dominates instance a_3 of A. Moreover, a_1 neither dominates nor is dominated by b_1, b_3. Clearly, A neither dominates nor is dominated by B completely. In fact, A takes a probability of $\frac{2}{9}$ dominating B, and B takes a probability of $\frac{1}{9}$ dominating A, assuming each instance takes the same probability to appear.

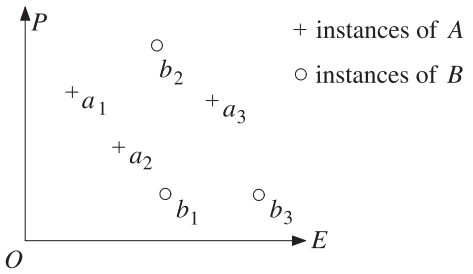

Fig. 1. Instances of two realtor uncertain objects.

Generally, a realtor as an uncertain object takes a probability not being dominated by any other realtors. We are particularly interested in the *skyline probability*—the probability that a realtor is not dominated by any other realtors. As a quality measure of the realtor's performance, the skyline probability quantifies the likelihood of a realtor is not worse than any other realtors. Due to a large number of realtors in the market, a new customer may often ask for only viewing a small portion of realtors who are less likely worse than other realtors, that is, higher skyline probabilities. While setting a skyline probability threshold p to retrieve realtors with skyline probability greater than p is also unable to control the size of realtors to be viewed, finding the top-k realtors who have the highest skyline probabilities is a natural way to control the size of realtors to be viewed. This is an example of computing top-k skyline objects from uncertain data.

The computation of top-k skyline objects is very useful in many other applications where ranking uncertain objects in a multi-dimensional space is involved. Fig. 2 shows a popular web site (http://www.restaurantratingz.com) in which each restaurant (uncertain object) may be evaluated by different customers with different knowledge levels (experiences) against food, ambience, and service, where knowledge level may be normalized to a probability value to represent the occurrence of each instance (food-rate, ambience-rate, service-rate) regarding each restaurant. Pei et al. [23] show that NBA players may be ranked using skyline against their game-by-game statistics, where each player is an uncertain object and the game-by-game statistics of each player are regarded

Fig. 2. Rating restaurants.

as the instances with the same probability to occur regarding each player (uncertain object). As demonstrated in [30], in many applications, the distances of an object from different facilities are important for the decision making. For example, in order to choose good observation points for forest fire management, we need to consider their distances to hydrology, roadways and fire points [37]. Due to the imprecision of the locations of the objects which is not uncommon in spatial database, systems often assume the locations of the objects follow particular distributions and hence the distance between a facility and an object is uncertain.

As demonstrated by the example (Table 2) in [23], computing the skyline probability of each uncertain object (NBA player) is different from computing the skyline of uncertain object using aggregates (e.g., the average scores) on the instances. Using aggregates like the average scores, the distribution information of instances is lost. On the contrary, the skyline probability of an uncertain object considers the relative distribution of the instances of the object against instances of other object. While the probabilistic model, p-skyline, formally define the probabilistic skyline against uncertain data, the algorithms to compute p-skyline are not applicable to the computation of top-k skyline objects. This is because that the algorithm for computing p-skyline outputs the objects with lower-bounds and upper-bounds of the corresponding skyline probabilities; nevertheless as shown in the experiment part in Section 7, higher lower- and upper-bounds do not necessaries imply higher skyline probability values. To the best of our knowledge, we are the first to investigate the problem of efficiently computing top-k skyline objects on uncertain data. In this paper, our investigation includes both discrete and continuous cases. Our principal contributions can be summarized as follows.

- We develop an efficient, threshold-based algorithm to compute the exact top-k skyline objects. The algorithm is based on a set of novel techniques to calculate skyline probabilities and prune objects.
- To address the applications with a large number of instances per object or a given continuous probability density function (PDF) per object, we develop an efficient randomized algorithm with an accuracy guarantee, ϵ-approximation. It follows the framework of our exact algorithm to effectively remove non-top-k skyline objects.

We evaluate the effectiveness and the efficiency of our techniques for computing top-k skyline objects on uncertain data. Our experiment results demonstrate the efficiency of both algorithms and also show that the randomized algorithm is highly accurate in practice. Moreover, our algorithms can be immediately applied to finding uncertain objects with skyline probabilities above a given threshold (p-skyline). While the randomized algorithm is the first technique to compute p-skyline regarding the continuous case with an ϵ-approximation guarantee, our exact algorithm significantly outperforms the existing techniques in [23].

The remainder of the paper is structured as follows. In Section 2, we model the problem and present preliminaries. Section 3 presents a framework to be adopted in the exact algorithm and randomized algorithm, as well as discrete and continuous cases. Section 4 applies the framework to our exact algorithm, while Section 5 presents novel techniques for the randomized algorithm. In Section 6, we extend our techniques to compute p-skyline. Section 7 reports the experiment results. This is followed by related work in Section 8. We conclude the paper in Section 9.

2. Background information

Points and/or instances referred in this paper, by default, are in a d-dimensional numeric space $D=\{D_1,\ldots,D_d\}$, where D_i denotes the i-th dimension. For two points u and v, u *dominates* v, denoted by $u \prec v$, if $u.D_i \le v.D_i$ for every $D_i \in D$, and there exists a dimension $D_j \in D$ where $u.D_j < v.D_j$. Given a set of points, the *skyline* consists of all points which are not dominated by any other point.

Example 1. Consider a set of points in Fig. 3. The skyline consists of a, b, and c, where b dominates d and e.

An uncertain object may be described by a probability density function (PDF) f_U such that $\int_{u\in U} f_U(u)\,du = 1$ where $f_U(u) \ge 0$; this is also referred as the continuous case. Nevertheless, in many applications PDFs are not always available. Instead, an uncertain object U is represented by a set of *instances* (points) such that each instance $u \in U$ has a probability p_u to appear. Such a representation, also referred as the *discrete* case, has the property that $0 < p_u \le 1$ and $\sum_{u\in U} p_u = 1$. In this paper, we investigate both discrete and continuous cases.

2.1. Problem definition

Discrete case. Given a set of uncertain objects $\mathcal{U}=\{U_1,\ldots,U_n\}$, a *possible world* $W=\{u_1,\ldots,u_n\}$ 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 $\sum_{W\in\Omega} Pr(W)=1$.

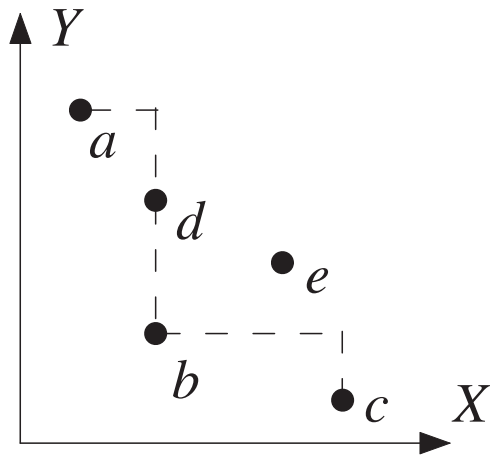

Fig. 3. certain data.

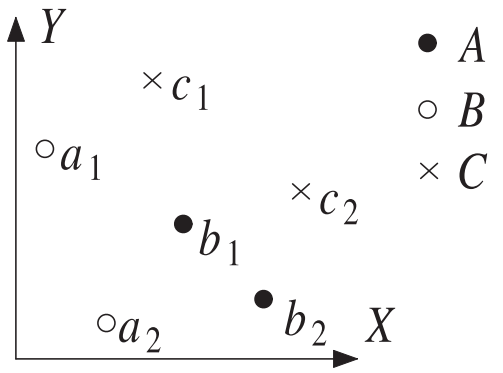

Fig. 4. uncertain data.

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.

Example 2. Fig. 4 plots a set of uncertain objects. Assume all instances take the probability 0.5 to appear. We have eight possible worlds in total. $P_{sky}(A)=1$ since a_1 and a_2 are in the skyline of every possible world.

Note that c_1 is dominated by every instance of A and c_2 is dominated by every instance of B; consequently, there is no possible world where C is in the skyline. Thus, $P_{sky}(C)=0$.

Note that B is in the skylines of four possible worlds $\{a_1, b_1, c_1\}$, $\{a_1, b_1, c_2\}$, $\{a_1, b_2, c_1\}$, and $\{a_1, b_2, c_2\}$. Therefore, $P_{sky}(B) = 4 \times (0.5 \times 0.5 \times 0.5) = 0.5$.

By the above definition, it can be immediately verified that for each instance u of $U \in \mathcal{U}$, the total probability of the possible worlds, in which instance u is in the skyline, is $p_u \times \prod_{V \in (\mathcal{U}-U)}(1-\sum_{v \in V, v \prec u} p_v)$. Let

$$P_{sky}(u) = \prod_{V \in (\mathcal{U}-U)} \left(1 - \sum_{v \in V, v \prec u} p_v\right).$$

$P_{sky}(u)$ is called the *skyline probability* of u, which is a conditional probability computed when we only consider the possible worlds containing instance u. It is also immediate that the skyline probability $P_{sky}(U)$ of U can be rewritten below:

$$P_{sky}(U) = \sum_{u \in U} p_u \times P_{sky}(u) = \sum_{u \in U} \left(p_u \times \prod_{V \in (\mathcal{U}-U)} \left(1 - \sum_{v \in V, v \prec u} p_v\right)\right). \quad (1)$$

Continuous case. Similarly, given a set of uncertain objects $\mathcal{U} = \{U_1, \ldots, U_n\}$ such that each U_i has a PDF f_{U_i} defined on U_i. The possible world semantic can be extended to cover the continuous case as follows. A *possible world* $W=\{u_1, u_2, \ldots, u_n\}$ is a point in the space $\Omega = \prod_{i=1}^{n} U_i$ such that $\int_{W \in \Omega} \prod_{i=1}^{n} f_{U_i}(u_i) du_1 du_2 \ldots du_n = 1$.

Similarly, we define $SKY(W)$ by including the objects with a point in the skyline of W. The skyline probability of U is

$$P_{sky}(U) = \int_{U \in SKY(W), W \in \Omega} \prod_{i=1}^{n} f_{U_i}(u_i) du_1 du_2 \ldots du_n. \quad (2)$$

This can be rewritten as

$$P_{sky}(U) = \int_{u \in U} f_U(u) \prod_{V \neq U} \left(1 - \int_{v \prec u, v \in V} f_V(v)\, dv\right) du. \quad (3)$$

Problem statement. In this paper, we investigate the problem of finding top-k skyline objects on uncertain data (top-k SOUND); it is formally defined below.

Definition 1 (*Top-k SOUND*). Given a set $\mathcal{U}$ of uncertain objects and an integer k, retrieve the k uncertain objects with the highest skyline probabilities.

Table 1 summarizes the notations used in this paper.

2.2. Preliminaries

Dominance relationships. A pair U, V of uncertain objects may have three kinds of dominance relationships as illustrated in Fig. 5.

Let *U.MBB* denote the minimum bounding box of the instances of an uncertain object U. U_{max} and U_{min} are the upper-right and lower-left corners of *U.MBB*, respectively. An object U is *fully dominated* by another object V if U_{min} is dominated by V_{max} or $U_{min}=V_{max}$ with the property that there is no instance from U allocated at U_{min} or there is no instance from V allocated at V_{max}. U is *partially dominated* by V if U_{max} is dominated by V_{min} but U is not fully dominated by V. Otherwise, U is *not dominated* by V. As

Table 1
The summary of notations.

Notation	Definition
U, V	Uncertain objects
u, v	Instances of uncertain objects
$f_U(u)$	Probability density function of U
p_u	The probability of u to appear
$M(U)$	The weighted centroid of U, i.e., $\sum_{u \in U} p_u \times u$
U.MBB	The minimum bounding box of U
U_{max} (U_{min})	The upper (lower) corner of *U.MBB*
$P_{sky}(U)$	Skyline probability of U
$P_{sky}(u)$	Skyline probability of u
$P_{sky}(U)\|_{\mathcal{U}}$	Skyline probability of U in $U \cup \mathcal{U}$
$Pr(U)$	The probability of U
$PD(U)$	The set of objects partially dominating U
n_U	The number of instances in U

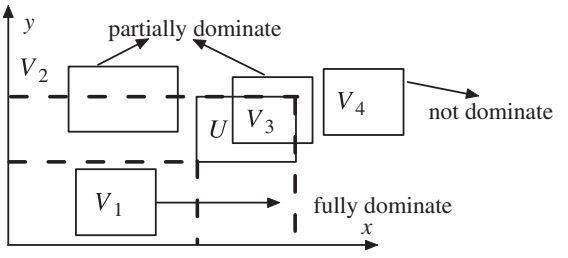

Fig. 5. Dominance relationships.

depicted in Fig. 5, U is fully dominated by V_1, partially dominated by V_2 and V_3, and is not dominated by V_4. Note that when U degenerates to one instance, the above concepts are immediately extendible if a point is treated as a special case of MBB.

An object is *redundant* if it is fully dominated by another object. Pei et al. [23] shows the following theorem.

Theorem 1. *Regarding the discrete case, a redundant object U has 0 skyline probability, and any instance from another object V dominated by an instance of U is fully dominated by a non-redundant object, and thus has 0 skyline probability.*

Theorem 1 immediately implies that any redundant object can be immediately removed. This is because that any instance dominated by an instance from a redundant object must be fully dominated by another non-redundant object.

Weighted centroids. Generally, the skyline probability of an object is determined by the distribution of its instances (*intra* relationships) and its relationships to the distributions of the instances of other objects (*inter* relationships). In our algorithms, we use the weighted (by probabilities) centroid $M(U)$ of instances to approximately represent the distribution of instances in object U to determine the processing order of objects. Formally, $M(U) = \sum_{u \in U} p_u \times u$.

R-tree. An initial computation in our algorithms is index-based, making use of the existing techniques. We assume that the minimum bounding boxes MBBs of objects' instances are indexed by an *R*-tree [11]. A node of an *R*-tree contains a set of entries. Each entry in a leaf node is in the form $\langle obj, obj.MBB \rangle$ where *obj* refers to the object ID and *obj*.MBB is the MBB of the object's instances. Each entry in a non-leaf node has form $\langle child, child.MBB \rangle$ where *child* refers to a child node, and *child*.MBB is the minimum bounding box of this child node. Fig. 6(a) illustrates an *R*-tree built on MBBs of nine uncertain objects. The root has three entries E_1, E_2, and E_3. Each child of the root encapsulate three objects, respectively.

BBS algorithm. BBS algorithm [21] will be used and modified in the initial computation phase of our algorithms. Given a set of points indexed by an *R*-tree, BBS algorithm traverses an *R*-tree (built on points) to compute the skyline. It maintains a *min-heap H* built against the *mindist* (minimum distance to the origin of the data space) of every entry (node). The algorithm goes iteratively. At the beginning, entries of the root are inserted into *H*. In each iteration, the top element *e* of *H* is processed. If *e* is fully dominated (i.e., the minimum corner of *e* is dominated) by an already computed skyline point, then *e* is discarded. Otherwise, if *e* is a data point, then it is output as a skyline point; if not, *e* is discarded and those entries of *e* which are not fully dominated by any already computed skyline point are inserted into *H*. An in-memory *R*-tree on already computed skyline points is maintained in order to facilitate examining the dominance relationship. The algorithm terminates when *H* is empty.

BBS ensures that each output point is in the skyline and it is I/O optimal.

3. Framework for top-*k* SOUND

Naively computing the skyline probability of each object is expensive and takes time $O(\sum_{\forall U,V} n_U \times n_V)$ for the discrete case, where n_U and n_V are the number of instances in objects U and V, respectively. The computation regarding the continuous case may be even more expensive due to integrating PDFs.

In the light of efficiently computing top-*k* queries (i.e., pruning away none top-*k* objects as soon as possible), below we present a framework to efficiently support both exact computation and randomized computation. It consists of three steps: *preprocessing–seeding–final computation*.

Step 1: Preprocessing. Remove redundant objects.

Step 2: Seeding. Select *k* objects and compute their skyline probabilities.

Step 3: Final computation. Finalize the computation of top-*k* SOUND.

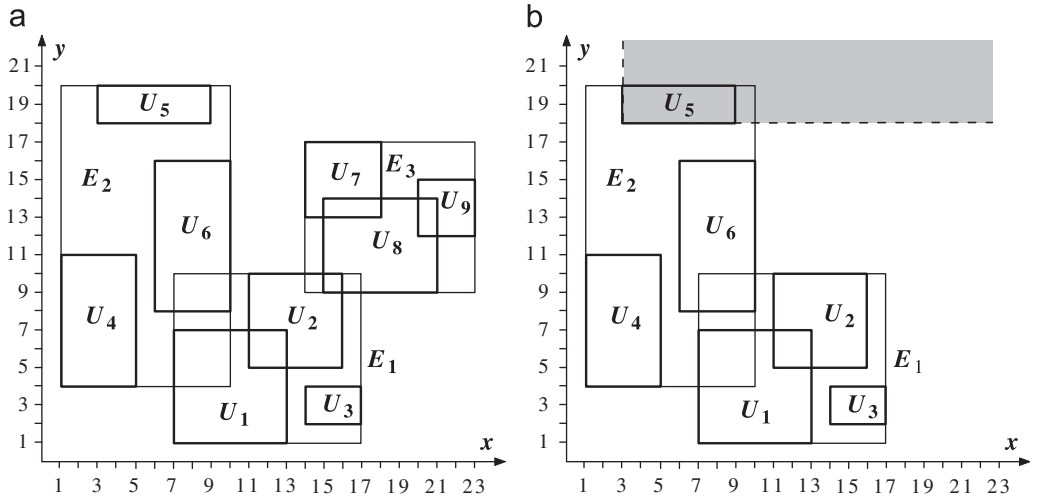

Fig. 6. Example. (a) Set of objects. (b) After removing E_3.

Note that after Step 1, if the number of objects left is not greater than k, then we can terminate the algorithm. Without loss of generality, we assume that there are more than k objects left after Step 1 in the rest of this section. Below, we present details in Steps 1–3.

3.1. Step 1: Preprocessing

As discussed in Section 2.2, Theorem 1 guarantees the correctness by removing redundant objects without affecting the skyline probability computation of remaining uncertain objects in discrete cases. Theorem 1 can be immediately extended to cover continuous cases. Note that if $u \prec v$, there must exist two small regions r_u and r_v surrounding u and v, respectively, such that r_u fully dominates r_v.

Theorem 2. *In continuous cases, a redundant object U also has* 0 *skyline probability. Moreover, any region in an uncertain object V fully dominated by a region in U is fully dominated by a non-redundant object, and hence has* 0 *skyline probability.*

Theorem 2 can be immediately verified according to the definitions. It implies that we can also remove the redundant objects regarding continuous cases without affecting the skyline probability computation for remaining objects.

We modify the original BBS algorithm to conduct the preprocessing. Below are some details of our modified BBS algorithm.

- An R-tree is built on MBBs; that is the unit data are MBBs of objects instead of points in the original BBS.
- Replace the dominance relationships among points by the *fully dominance relationship* among MBBs of objects in the modified BBS.
- In modified BBS, for every data entry in the R-tree we adopt the distance $d_M(U)$ between the centroid $M(U)$ and the origin as *mindist* to serve as the key of the heap, while for an internal node (entry) in the R-tree, the minimum of such distances among the objects (data entries) contained is used as *mindist*; we assume that they are recorded in the R-tree.

It can be immediately verified that for two objects U and V, if $d_M(U) < d_M(V)$ then U will not be fully dominated by V. This guarantees no false positive and hence the modified BBS algorithm can remove all redundant objects.

The Algorithm 1 illustrates the details of the modified BBS. Regarding the objects in Fig. 6(a), after Step 1, only the objects in Fig. 6(b) remain.

Algorithm 1. *Modified BBS*

Input:
R: R-tree on MMBs of the objects;
Output:
R_{nr}: an in-memory R-tree index of non-redundant objects;
R_s: an in-memory R-tree index of skyline of the weighted centroid of non-redundant objects;
Description:
1: $H := \emptyset$;$R_s := \emptyset$; $R_{nr} := \emptyset$;
2: push root of R into heap$H$;
3: **while** $H \neq \emptyset$ **do**
4: $e := H.deheap()$;
5: **if** $e.MBB$ is not *fully dominated* by MMB of any object in R_{nr} **then**
6: **if** e is MBB of an object U **then**
7: Put e to R_{nr};
8: Put $M(U)$to R_s if it is not dominated by any points in R_s;
9: **else**
10: Push all child entries of e into H;
11: **return** R_{nr}, R_s

3.2. Step 2: seeding

The aim is to initially choose k objects with large skyline probabilities as a threshold to quickly prune away objects with small skyline probabilities without conducting the entire computation of their skyline probabilities. Intuitively, an object U with $M(U)$ as a skyline point, of all weighted centroids, may have more instances not being dominated by other objects' instances; thus, it has a good chance to have a high skyline probability value. Moreover, we also give the preference to $M(U)$ with the smallest $d_M(U)$ since intuitively, smaller $d_M(U)$, less chance U being dominated by others. Note that the selection of k objects does not affect the correctness of the Algorithm 2. Nevertheless, a good selection might help to avoid computing skyline probabilities of many objects.

Algorithm 2. Seeding

Choose the k objects U such that:
- $M(U)$ are skyline points with smallest $d_M(U)$, and
- if there are not enough skyline points $M(U)$, then choose the remaining objects V with the smallest $d_M(V)$ to make total k objects.

Algorithm 2 involves the computation of the skyline of the weighted centroids of non-redundant objects. This could be separately conducted by first computing all non-redundant objects and then retrieving skylines of their weighted centroids. In our algorithm, we conduct this simultaneously by executing the original BBS on centroids while computing non-redundant objects at the same time. This can share the computational costs since in our modified BBS $d_M(U)$ $(\forall U)$ is already used as *mindist*; that is, we only need to maintain one heap.

Note that BBS, so does the modified BBS, always generates objects sorted increasingly on the search key *mindist* ($d_M(U)$) as a by-product. Thus, after running the modified BBS on objects and the original BBS on weighted centroids, the non-redundant objects $\{U\}$ are sorted as a sorted list L_2 on $d_M(U)$ (non-decreasingly) and the objects with $M(U)$ as skyline points are also sorted as a sorted list L_1 against $d_M(U)$ (non-decreasingly). Clearly, L_2 contains all objects in L_1. Nevertheless, we can remove from L_2 the objects in L_1, while running BBS on the centroids and modified BBS simultaneously, as the objects are processed in the same order in these two algorithms. Algorithm 2 is thus executed in linear time $O(k)$.

Computing the skyline probability. The last phase of Step 2 (seeding) is to compute skyline probability for each seeded object, totally k objects. The computation is different for the exact algorithm and the randomized

algorithm. Our computation techniques will be introduced in Sections 4 and 5, respectively.

3.3. Step 3: final computation

The framework regarding this step is outlined below. We iteratively process each object U as follows.

Filtering. If $P_{sky}(U) \le \mathcal{P}_k$, then U is not a candidate of top-k SOUND. Here, $\mathcal{P}_k$ is the smallest skyline probability of the current top-k objects. Note that current objects refer to the objects processed for their skyline probability computation so far, including the objects chosen as seeds.

Refinement. Otherwise replace by U the object V, with the skyline probability $\mathcal{P}_k$, in the current top-k objects. Update $\mathcal{P}_k$.

Suppose that for each object U, $PD(U)$ denotes the set of objects each of which partially dominates U. Clearly, computing $P_{sky}(U)$ takes time $(n_U \times (\sum_{V \in PD(U)} |n_V|))$ if $PD(U)$ is already obtained.

An object is called a *master object* when we compute its skyline probability in the final-computation phase. Then objects in $PD(U)$ are called *associated objects* to the master object U since they will be employed to compute $P_{sky}(U)$. There are two key issues.

1. By which order are the associated objects (i.e., objects in $PD(U)$) accessed against a master object?
2. By which order are objects accessed as master objects?

Our experiments demonstrate that a random selection towards these two issues leads to the computation time an order of magnitude slower than the techniques developed below.

Order of associated objects. The main goal of this step is to develop efficient and effective techniques to prune away a master object U as earlier as possible; that is, access as less as possible the objects in $PD(U)$. The following theorem is immediate from the definitions.

Theorem 3. *Suppose that $\mathcal{U}'$ is a subset of the set $\mathcal{U}$ of objects. Then, $P_{sky}(U)|_{\mathcal{U}'} \ge P_{sky}(U)|_{\mathcal{U}}$, where $P_{sky}(U)|_{\mathcal{U}}$ ($P_{sky}(U)|_{\mathcal{U}'}$) denotes the skyline probability of U regarding the set $\{U\} \cup \mathcal{U}$ ($\{U\} \cup \mathcal{U}'$) of objects.*

The monotonic property in Theorem 3 implies that for the skyline probability computation of object U, we only need to access objects $\mathcal{U}'$ if $P_{sky}(U)|_{\mathcal{U}'} \le \mathcal{P}_k$ since U can be safely excluded from top-k SOUND by only considering $\mathcal{U}'$.

Pruning Rule 1. *For an object U, let $\mathcal{U}$ be a subset of $PD(U)$. U can be excluded from the candidates of Top-k SOUND if $P_{sky}(U)|_{\mathcal{U}} \le \mathcal{P}_k$.*

Example 3. Regarding the example in Fig. 6(b), suppose that $k=2$ and objects U_1 and U_4 are initially chosen. When computing the probability value of U_5, we may find that the $P_{sky}(U_5)|_{U_4}$ is already smaller than $\mathcal{P}_k$. Thus, we no longer need to do a further computation between U_5 and U_6, nor U_5 and U_1.

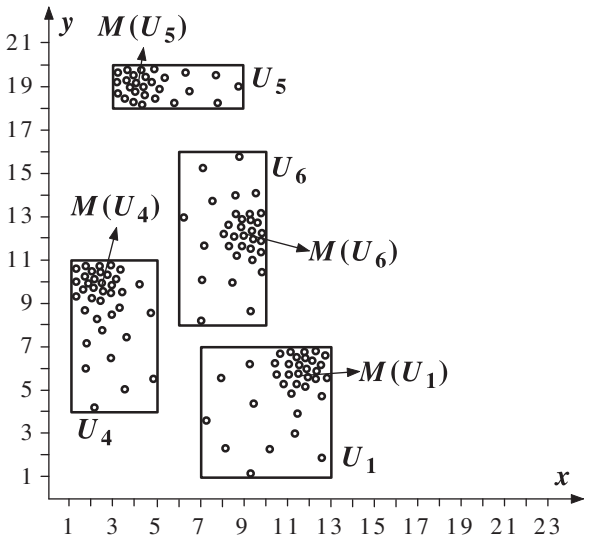

Fig. 7. Data distributions.

Continuing this example, suppose that the distributions of instances in U_1, U_5, U_4, and U_6, respectively, are as what illustrated in Fig. 7. Clearly, it is better to start with the pair of U_4 and U_5 as the $P_{sky}(U_5)|_{U_4}$ is intuitively smaller than $P_{sky}(U_5)|_{U_1}$ or $P_{sky}(U_5)|_{U_6}$. Thus, there is a great chance to eliminate U_5 by computing $P_{sky}(U_5)|_{U_4}$ only.

Ideally we would like to find a *perfect* permutation, $\{U_i : 1 \le i \le |PD(U)|\}$, of the objects in $PD(U)$ such that, for each $1 \le i \le |PD(U)|$, $P_{sky}(U)|_{\{U_1,\ldots,U_i\}}$ is minimum among any subset of $PD(U)$ with i objects. That is, when U is pruned away from the candidates, the number of objects accessed from $PD(U)$ is always minimized. Nevertheless, such a permutation does not always exist.

Example 4. For example, suppose that there are four objects U, U_1, U_2, and U_3, where U is the master object and has only two instances u_1 and u_2 with the equal probability 0.5 such that:

- the probabilities that u_1 is not dominated by U_1, U_2, and U_3 are 0.15, 0.3, and 0.4, respectively;
- the probabilities that u_2 is not dominated by U_1, U_2, and U_3 are 0.9, 0.8, and 0.7, respectively.

It can be immediately verified that to follow the property specified above for each i in a perfect permutation, the first associated object has to be accessed is U_1. Nevertheless when $i=2$, the skyline probability of U against U_2 and U_3 is the minimum. Therefore, such a perfect permutation does not exist.

Below we develop a ranking function based on Eq. (4) to order associated objects so that a good permutation may be obtained. It is based on the following observation. Let U and V be two uncertain objects where U is a master object. Consider that $M(U)$ approximately represents the instance distribution of object U. Intuitively, when $M(U)$ is dominated by an $M(V)$, we may expect that U has more chances to be dominated by V; in this case, the further the

distance between $M(U)$ and $M(V)$ is, the larger the chance that the probability of U dominated by V has a larger value (i.e., the smaller the skyline probability of U).

$$\Delta(U,V) = \delta(U,V)d(M(U),M(V)). \quad (4)$$

Here, d is a distance metric; Manhattan distance is used in our implementation; $\delta(U,V)=1$ if $M(V) \prec M(U)$, otherwise -1. Based on the above observations, in our algorithm we choose associated objects one-by-one increasingly according to Δ values.

Order of master objects. The goal is to make $\mathcal{P}_k$ increase as quickly as possible to reach the k-th largest skyline probability. Sharing with the same intuition in seeding algorithm (Algorithm 2), we choose a master object according to the priority described in Algorithm 2.

Algorithm. We present our algorithm to determine the final top-k according to the above discussions.

Algorithm 3. Final computation

Input:
L_1: a sorted list of remaining objects (with the weighted centroids as skyline points) on $d_M(U)$;
L_2: a sorted list of remaining objects (with the weighted centroids as non-skyline points) on $d_M(U)$;
Output: TOP_k: min-heap on the skyline probabilities of k objects, together with the corresponding object IDs;
Description:
1: **for** each U of initial k objects **do** TOP_k.push (U);
2: $\mathcal{P}_k := TOP_k.top.key$;
3: **for** each $U \in L_1 \cup L_2$ **do**
4: **if** $ProbB(U,\mathcal{U}_{NR}) > \mathcal{P}_k$ **then**
5: $TOP_k.pop()$; $TOP_k.push(U)$;
6: $\mathcal{P}_k := TOP_k.top.key$;
7: **return** TOP_k

As described in the seeding phase (Section 3.2), L_1 is a by-product of BBS algorithm, while L_2 is a by-product of the modified BBS. In Algorithm 3, $\mathcal{U}_{NR}$ is the set of non-redundant objects from a given set $\mathcal{U}$ of objects. We accessing $L_1 \cup L_2$ by firstly accessing L_1 and then accessing L_2 according to their sorted order, respectively.

To save the storage space, in TOP_k we only keep object IDs and their corresponding skyline probabilities. The method $ProbB(U,\mathcal{U}_{NR})$ is employed to calculate the skyline probability of the object U. Note that U might be pruned based on probabilistic threshold $\mathcal{P}_k$ during the computation. This procedure is different for the exact algorithm and the randomized algorithm since different pruning techniques will be employed. Details of method $ProbB(U,\mathcal{U}_{NR})$ will be presented in the next two sections.

4. Exactly computing top-K SOUND

The exact algorithm for discrete cases follows the framework of three steps in Section 3. In this section, we present the details of computing the skyline probability in Step 2—Section 3.2, and our pruning strategies in Step 3—Section 3.3.

Computing the skyline probability. The last part of Step 2 computes the skyline probabilities of the initially chosen k objects.

Note that an in-memory R-tree on MBBs of non-redundant objects has been built as a by-product of our modified BBS, corresponding to the in-memory R-tree on skyline points in the original BBS.

For each U of these initially chosen k objects, our algorithm to compute its skyline probability is conducted in two stages. At stage 1, it iteratively traverses the in-memory R-tree in a depth-first manner to search for objects with MBBs partially dominating U; that is, search for objects in $PD(U)$. Once such an object V is found, it performs an update of the skyline probability of each instance of U. In our implementation, the *synchronous traversal* (ST) join paradigm $ST(U,V)$ [14] is adopted instead of trivially comparing each pair of instances from U and V. Let CH_R denote the set of children of the root of R. Algorithm 4 presents our algorithm to update the skyline probability. Note that there are only two relationships among non-redundant objects: partially dominating or not dominating. If R' does not (partially) dominate U, then R' can be simply passed-over since each $u \in U$ has the probability 1 not being dominated by any object in R'.

Algorithm 4. *Prob* (U, R)

Input:
R: an in-memory R-tree index of non-redundant objects;
U: an object;
Output: $P_{sky}(U)$
Description:
1: $Q := CH_R$;
2: remove entries from Q that do not (partially) dominate U;
3: **while** $Q \neq \emptyset$ **do**
4: $R' := Q.pop()$;
5: **if** $R'.MBB$ does not (partially) dominate U **then**
6: Pass-over R';
7: **else**
8: **if** R' is an object $V \neq U$ **then**
9: $ST(U,V)$;
10: $U := REMOVE\_ZERO(U)$;
11: **else**
12: $Q := Q \cup CH_{R'} - \{R'\}$;
13: $P_{sky}(U) := \sum_{u \in U} p_u \times P_{sky}(u)$;
14: **return** $P_{sky}(U)$

In Algorithm 4, Q is maintained as a queue. To facilitate the synchronous traversal join paradigm, the instances in each object are pre-organized by an in-memory R-tree data structure such that at each node E, we also record p_E—the summation of the probabilities of the instances which are descendant of E in the R-tree. $ST(U,V)$ is a simple modification of the synchronous traversal join algorithm to conduct an in-memory update of the skyline probability of each instance in U due to an addition of object V. We only need to modify the join condition to "one rectangle (point) fully dominates another rectangle (point)". In $ST(U, V)$, for a pair of node $E \in U$ and node $E' \in V$, there are three cases below.

Case 1: If E' does not dominate E, pass-over E'.

Case 2: If E' fully dominates E, then $Pr(E) := Pr(E) + p_{E'}$. Note that $Pr(E)$ is initiated to 0 when a new object is added, and is the summation of the occurrence probabilities of instances in an E' which is captured in $ST(U,V)$ to fully dominate E.

Case 3: Otherwise (E' partially dominates E), put (not Case 1) pairs of children of E' and E in a queue for further traversal.

To compute the skyline probability correctly, after performing $ST(U, V)$ for an object V, we push down $Pr(E)$ from each internal node E to the leaf nodes (instances) along the tree path. That is, $Pr(u) = \sum_{E \in l_u} Pr(E)$, where l_u is the path from the root to the leaf u. Note that the whole push-down computation can be performed in linear time if it is executed in a top-down fashion. Moreover, after push-down, we update $P_{sky}(u)$ to $P_{sky}(u) := P_{sky}(u)(1-Pr(u))$[1], and then reset $Pr(E)=0$ for each entry E in the R-tree, including leafs.

Remove instances with skyline probability 0 *by REMOVE_ZERO* (U). The following Theorem is fundamental.

Theorem 4. *Suppose that an instance u has* 0 *skyline probability. Then, there must be a non-redundant object V such that u is fully dominated by V; that is, each instance of V fully dominates u. Moreover, no instance from V has skyline probability* 0.

Proof 1. Suppose that PS is the set of objects such that the right-upper corners of all objects in $U \in PS$ form the skyline against all those of the non-redundant objects. It can be immediately verified that one object from PS must fully dominate u. Moreover, it is also immediate that none of instances in an object U in PS has 0 skyline probability. □

Instances with skyline probability 0 can be removed from U from further considerations. Firstly, removing them from U implies that while computing the skyline probability of U, these instances will not be counted. This is equivalent to counting their probabilities as 0. Secondly, removing them from U will not affect the computation of skyline probabilities of other objects. This is because any instance v fully dominated by an instance u with skyline probability 0 must be fully dominated by a non-redundant object V' without any instance removed according to Theorem 4. Consequently, the 0 skyline probability can be discovered from the relationship between V' and u. Thirdly, removing these instances not only saves the memory space for the scalability but also reduces the computation costs when U is used in computing the skyline probabilities of other objects.

Example 5. Regarding the example in Fig. 7, once the instances in U_6 with skyline probability 0 are removed, we no longer need to use them when update the skyline probability of U_5 by adding U_6.

In *REMOVE_ZERO*(U), we remove the instances with 0 skyline probability; if an entry in the R-tree on U's instances only contains the instances with 0 skyline probability, then the entry is removed as well. We do not re-balance the R-tree of U as our initial experiment demonstrates that such re-balance costs cannot be paid off. Note that we do not physically remove instances or entries from a pre-built in-memory R-tree; instead, we mark out those "removed" instances and entries to prevent them from being involved in further computation.

[1] Note that $P_{sky}(u)$ is initialized to 1.

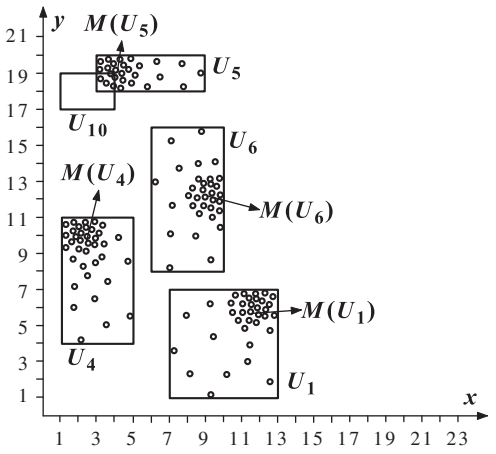

Fig. 8. Dealing I_U.

Processing ProbB(U,$\mathcal{U}_{NR}$). It can be done exactly in the same way as Algorithm 4. However, as implied by Theorem 3, we do not have to always conduct the entire computation of $P_{sky}(U)|_{\mathcal{U}_{NR}}$. To facilitate this, we always choose an entry with the largest Δ value. As discussed above, $\mathcal{U}_{NR}$ is the set of non-redundant objects that are indexed by an in-memory R-tree as a by-product of our modified algorithm.

It can be immediately verified that Theorem 3 also holds for the situation where instances with skyline probability 0 are removed; this together with Theorem 3 yields that we can add associated objects one-by-one to calculate skyline probability and prune away a master object U against a subset of PD (U) based on Pruning Rule 1.

Prob B (U,$\mathcal{U}_{NR}$) modifies *Prob*(U,R) (Algorithm 4) as follows.

- In *Prob B* (U,$\mathcal{U}_{NR}$), we maintain a max-heap Q based on Δ values of the R-tree entries instead of a queue where Δ values are calculated on the fly. To retain the monotonic property that for each internal entry E, its Δ value is the maximum of Δ values of the entries/objects contained,

  for each entry E, we record $u_{E,M}$ which is the lower-left corner of the minimum bounding box of the weighted centroids of objects contained in E; then compute the Δ value using $M(U)$ and $u_{E,M}$.
- Between lines 10 and 11 in Algorithm 4, we calculate the current skyline probability after adding one associated object; that is, add $\sum_{u \in U} p_u P_{sky}(u)$. If it is already not larger than $\mathcal{P}_k$, then we terminate *Prob B* (U, R); consequently U is excluded from the candidates of top-k SOUND; that is, the condition in line 4 of Algorithm 3 does not hold.

Remark. A non-redundant object may have 0 skyline probability for every instance. As illustrated in Fig. 9, U is a non-redundant object. Nevertheless, the skyline

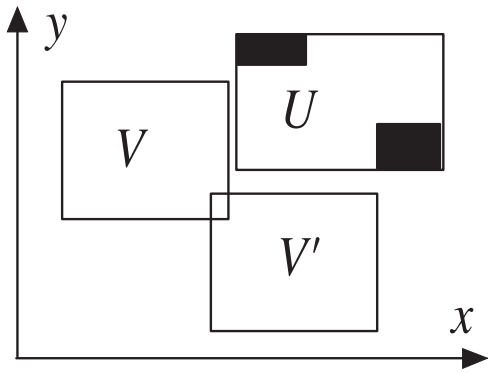

Fig. 9. Multiple dominance relationships.

probability of U is zero if its instances are located in the two black-colored boxes only.

Generally, an object U may have a subset I_U of instances such that each instance in I_U has 0 skyline probability. I_U can be removed from U without further consideration, as explained before. This can be done by using window query techniques to detect the instances dominated by the skyline points on the upper-right corners of MBBs of non-redundant objects. However, our initial experiments suggest that such pre-computation costs cannot be paid off.

In our algorithm, we only remove a subset of instances, captured with 0 skyline probability on the fly, in examining its top-k candidature. For instance, regarding the example in Fig. 8, U_5 may be excluded from a top-k candidate after computing $Psky(U_5)|U_4$ without examining U_{10}. Consequently, before U_5 is excluded from a further consideration (i.e., examining U_{10}, etc.) as a master object we are only able to capture the set of instances in U_5 fully dominated by U_4 but not those fully dominated by U_{10} only.

5. Randomized algorithm

In this section, we present a randomized algorithm to deal with both continuous (with the assumption that PDFs are *continuous* functions) and discrete cases. Let $\mathcal{U}=\{U_j|1\le j\le n\}$ be the set of non-redundant objects. The basic idea is to sample all possible worlds, $\prod_{i=1}^{n} U_j$, by m possible worlds S_i; that is, each sample S_i $(1\le i\le m)$ consists of n randomly chosen points for the continuous case (or instances for the discrete case)—one per each object. Then, we use m_U/m as the approximation of the skyline probability of an object U to determine the solution for top-k SOUND. Here, m_U is the number of times that object U is involved as the skyline points in these m samples (worlds).

Example 6. Consider the example in Fig. 4. Regarding two samples (worlds) (i.e., $m=2$) (a_1, b_1, c_2) and (a_2, b_1, c_1), $m_B=1$).

Algorithm 5. Randomized algorithm

Input: $\mathcal{U}=\{U_i|1\le i\le n\}$; m: an integer.
Output: T_k: k objects.

Description:
1: **for** $i:=1$ to m **do**
2: **for** $j:=1$ to n **do**
3: $u_{i,j}:=random(U_j)$;
4: Sky-COUNT $(\{u_{i,j}|1\le i\le m,1\le j\le n\})$;
5: $T_K:=$ the k objects U with the largest m_U/m;
6: **return** T_k

In Algorithm 5, regarding the discrete case we use *random* (U_j) to randomly select an instance from U_j such that each instance $u\in U_j$ has the probability p_u to be selected. Regarding a continuous case, we first divide the whole data space, the MBB of U, into very small regions such that in each region, the difference of values of a PDF is bounded by a very small value ξ. Then, *random* (U_j) randomly selects a point from a region r with probability $Pr(r)$. Sky-COUNT $(\{u_{i,j}|1\le i\le m,1\le j\le n\})$ computes m_U for each object U. Computing the skyline of each sample $S_i=\{u_{i,j}|1\le j\le n\}$ (for $1\le i\le m$) to get each m_U is expensive, even more expensive than the exact algorithm, when m is reasonably large. Below, we will present an efficient counting technique. First, we present the accuracy guarantee of Algorithm 5.

5.1. Accuracy guarantee

For each object U_j, in Algorithm 5 the events whether $u_{i,j}=random(U_j)$ is a skyline point of S_i is described by the following totally independent random variable:

$$X_{i,j}=\begin{cases}1 & \text{if } u_{i,j} \text{ is a skyline in sample i}\\ 0 & \text{otherwise}\end{cases} \tag{5}$$

It is immediate that $E(X_{i,j})=\sum_{u\in U_j} p_u P_{sky}(u)$ in a discrete case for each i and j. For a continuous case, without loss of generality we may assume that for each object U its PDF domain is a finite region and S is the maximum of the region volumes of these n uncertain objects.[2] We choose ξ such that $\xi<\epsilon/(2^n\times S\times 4)$. Consequently, it is immediate that $E(X_{i,j})=\int_{u\in U_j} f_{U_j}(u)\prod_{V\ne U_j}(1-\int_{v\prec u, v\in V} f_V(v)\,dv\,du)+\xi_{U_j}$ where for each U, $0\le|\xi_U|<\epsilon/4$; that is, $E(X_{i,j})=P_{sky}(U_j)+\xi_{U_j}$. Let

$$X_j=\frac{\sum_{i=1}^{m} X_{i,j}}{m} \tag{6}$$

We have $m_{U_j}/m=X_j$ and $E(X_j)=P_{sky}(U_j)+\xi''_{U_j}$, where $|\xi''_{U_j}|\le\epsilon/8$. Given a set of objects, for $1\le l\le k$ let P_l denote the skyline probability of the object with the l-th largest skyline probability. Suppose that for $1\le l\le k$, U_{j_l} is ranked as the top l-th object by Algorithm 5. Note that the object U_{j_l} could be different than the real top l-th object (with skyline probability P_l). Nevertheless, the following theorem states that when the sample size m is sufficiently large, X_{j_l} $(=m_{U_{j_l}}/m)$ will be an ϵ-approximation of P_l with confidence $1-\delta$.

[2] For a PDF of an uncertain object U with an infinite domain, we can simply cut the infinite part of the domain with a very small probability ξ'_U. Then, the following analysis still holds.

Theorem 5. *Given an* ϵ $(0<\epsilon<1)$, *a* δ $(0<\delta<1)$, *and n non-redundant objects, if* $m=O((1/\epsilon^2)\log n/\delta)$ *and* ξ *is chosen with* $\xi<\epsilon/(8\times 2^n\times S)$, *then*

$$Pr\left(\bigwedge_{l=1}^{k}\{|X_{j_l}-P_l|\le\epsilon\}\right)\ge 1-\delta.$$

To prove Theorem 5, we need the following Lemmas. By the *Chernoff/Hoeffding* bound (Theorem 2.7 in [10]), the following Lemma is immediate. It implies that when the sample size is sufficiently large, X_j is very close to the skyline probability of U_j with a high confidence.

Lemma 1. $Pr(|X_j-P_{sky}(U_j)|\ge\lambda)\le 2\exp^{-2m(\lambda-\epsilon/8)^2}$ $(0<\lambda\le 1)$ $\forall j\in[1,n]$.

The following lemma states that if the sample size is sufficiently large, then with a high confidence, Algorithm 5 can only reverse the order, against their skyline probabilities, of two objects with a small difference between their skyline probabilities.

Lemma 2. *For* j *and* j', *suppose that* $P_{sky}(U_j)<P_{sky}(U_{j'})$. *Then*, $Pr(X_j\ge X_{j'})\le\exp^{-m(P_{sky}(U_{j'})-P_{sky}(U_j)-\epsilon/4)^2/2}$.

Proof 2. Let $Z=X_j-X_{j'}$. We have $Pr(X_j\ge X_{j'})\le Pr(Z-E(Z)>P_{sky}(U_{j'})-P_{sky}(U_j)-\epsilon/4)$. By Hoeffding inequality (Theorem 2 in [13]), the lemma is immediate. □

Proof of Theorem 5. Without loss of generality, suppose that for an $l\in[1,k]$, the object U_l has the l-th largest skyline probability P_l (i.e., $P_{sky}(U_l)=P_l$). For each object U_{j_l} (ranked the l-th by Algorithm 5), we prove the probability of the following three events to appear is small when m is chosen appropriately.

Event 1: $P_{sky}(U_l)-P_{sky}(U_{j_l})>\epsilon/2$.
Event 2: $P_{sky}(U_{j_l})-P_{sky}(U_l)>\epsilon/2$.
Event 3: $|X_{j_l}-P_{sky}(U_{j_l})|>\epsilon/2$.

Let $m=(8/\epsilon^2)\log 2(n+1)k^2/\delta$.

If Event 1 occurs, then $\exists U_i$ such that $P_{sky}(U_i)\ge P_l$ (i.e., $i\le l$) and $X_i\le X_{j_l}$. This implies that U_i and U_{j_l} change their order by Algorithm 5. Consider that there are l such objects. Consequently, from Lemma 2, the total probabilities that U_{j_l} change the order with these l objects is bounded by $l\delta/4k^2(n+1)$. Therefore, $Pr(Event1)\le l\delta/4k^2(n+1)$.

If Event 2 occurs, then $\exists U_i$ such that $P_{sky}(U_i)\le P_l$ (i.e., $i\ge l$) and $X_i\ge X_{j_l}$. Note that there are $(n-l+1)$ such objects. Similarly, from Lemma 2, we have $Pr(Event2)\le(n-l+1)\delta/4k^2(n+1)$.

Let $\lambda=\epsilon/2$. By Lemma 1, $Pr(Event\ 3)<\delta/2k$.

Therefore, for all l $(1\le l\le k)$ the probability of that one of these $3k$ events occurs is not greater than δ. Consequently, the theorem holds since $(16/\epsilon^2)\log 2(n+1)k^2/\delta=O((1/\epsilon^2)\log n/\delta)$.

Discussions. Note that the sample size in Theorem 5 is irrelevant to the number of instances in an object. If we run the seeding phase, and choose ϵ as $\epsilon_1\mathcal{P}_k$ if $\mathcal{P}_k\neq 0$, then we can guarantee a relative ϵ-approximation theoretically. Theoretically, to guarantee ϵ-approximation, we need a sample size as stated in Theorem 5; nevertheless, our experiment demonstrate that $m=1000$ can provide a quite accurate solution to top-k SOUND.

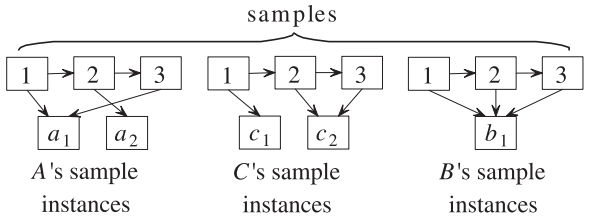

Fig. 10. Example of samples.

5.2. Efficient algorithm

The Sky-COUNT in Algorithm 5 follows the framework in Section 3; that is, three steps: preprocessing, seeding, and final computation. While others are exactly the same as those in the exact algorithm, we present efficient techniques to compute m_U/m (corresponding to Algorithm 4—*Prob*( )) for the initially chosen k objects in the seeding phase and execute *Prob B* ( ) in Algorithm 3 in Step 3. We aim to directly compute m_U for each object U by avoiding computing the skyline for each sample.

In our techniques, we organize sampled instances (points) of each object U as a list *U.list* to save the storage space by removing duplicates.[3] Initially, the i-th node of this linked list contains the integer i, and the reference referring to the instance of U in the i-th sample. For instance, regarding the example in Fig. 4, three samples are dropped, $S_1=\{a_1, c_1, b_1\}$, $S_2=\{a_2, c_2, b_1\}$, and $S_3=\{a_1, c_2, b_1\}$. Their linked lists are illustrated in Fig. 10.

The basic idea of our counting algorithm is as follows. If a sampled instance (point) u of an object U is dominated by a sampled instance (point) of another object V from the same sample, we simply remove the sampled instance u from the linked list. In the end, the number of sampled instances remained in each object U is m_U. Regarding the above example in Figs. 4 and 10, in *A.list* three sampled instances $\{a_1, a_2, a_1\}$ are left after our algorithm; thus $m_A=3$. In *C.list* and *B.list*, 0 and 2 sampled instances ($\{b_1, b_1\}$) are left, respectively; thus $m_C=0$ and $m_B=2$.

While our counting techniques follow the framework of EXACT algorithm except that *Prob* (U, V) and *Prob B* (U, V) are executed differently. We do not use a tree-like data structure to organize the sampled instances (points). Consequently, we replace the $ST(U,V)$ in both $Prob(U,V)$ and *Prob B*(U, V) by *dominance check* $DC(U,V)$ which checks the dominance relationship between the instances of U and V within the same sample.

In $DC(U,V)$ where U is a master object, we adopt the same traversal strategy as the sort-merge join between *U.list* and *V.list* since they are sorted on sample subindexes. Once u and v are found in the same sample (by their corresponding sample subindexes), we remove the sampled instance (point) from *U.list* if $v\prec u$, or remove

[3] An instance may appear in several samples especially in the discrete case.

the sampled instance (point) from *V.list* if $u \prec v$. An instance (point) is removed if there is no sampled instance (point) referring to it any more.

Example 7. In the example in Fig. 10, in $DC(C,A)$ after checking against the first sample index 1, element 1 is removed from *C.list*. Since instance c_1 only has element 1 referring to it, instance c_1 is also removed.

Clearly, the complexity of $DC(U,V)$ is $O(dm)$ where d is the dimensionality. Moreover, Sky-COUNT runs in time $O(d \times n \times m \times a + dn\mathcal{F}(n))$. Here, a is the average number of associated objects to a master object and $\mathcal{F}(n)$ represents the average costs for one master object to obtain associated objects. Consequently, if a is a constant and m is a constant (say, 1000), then the time complexity of our Sky-COUNT is $O(dn\mathcal{F}(n))$. While there is no theoretical guarantee on $\mathcal{F}(n)$ regarding an R-tree, we could expect that $\mathcal{F}(n)$ is poly-log (n) in practice in a low dimensional space when a is a constant.

6. Computing *p*-skyline

Our exact and randomized algorithms can be immediately extended to compute p-skyline proposed in [23]; that is, for a given threshold p $(0 \le p \le 1)$, compute all uncertain objects U such that $P_{sky}(U) \ge p$. Below are the modifications.

Regarding the framework in Section 3, in our exact and randomized algorithms for computing p-skyline we keep Step 1 but do not use Step 2. In Step 3 of both algorithms, we prune away the objects U with skyline probabilities (or m_U/m) below a pre-given threshold p, the Step 3 of both exact and randomized algorithms can be immediately applied to compute the p-skyline. The modified algorithms are named p-EXACT and p-RAND, respectively.

It can be immediately verified that the p-EXACT is correct. Moreover, by the Chernoff/Hoeffding bound (Theorem 2.7 in [10]) together with the fact that for each output object U $(m_U/m) \ge p$, the following theorem regarding accuracy immediately holds.

Theorem 6. *Given an* ϵ $(0 < \epsilon < 1)$ *and a* δ $(0 < \delta < 1)$, *let* $m = O(1/\epsilon^2 \log(1/\delta))$ (*the sample size in p-RAND*). *For each object U output by p-RAND*, $Pr(P_{sky}(U) - p < -\epsilon) \ge 1 - \delta$, *where p is a given probability threshold in the problem of p-skyline*.

Theorem 6 states that with confidence $1-\delta$, the objects output by the algorithm p-RAND with the skyline probability not less than $p-\epsilon$. Note that Theorem 6 immediately implies that if we replace $m = O(1/\epsilon^2 \log(1/\delta))$ by $O(1/(p\epsilon)^2 \log(1/\delta))$, then we will have a relative ϵ-approximation guarantee, that is, with confidence $1-\delta$, the objects output by the algorithm p-RAND with the skyline probability not less than $(1-\epsilon)p$.

Following similar arguments to those in TOP-k SOUND, the algorithm p-RAND is immediately applicable to the continuous case with the above accuracy guarantee.

Beside the theoretical guarantee of accuracy as above, our experiment demonstrated that p-RAND has already been very accurate when m reaches 1000.

7. Performance evaluation

We report an extensive empirical study to evaluate the effectiveness and the efficiency of our algorithms. All algorithms are implemented in C++ and compiled by GNU GCC. Experiments are conducted on PCs with Intel P4 2.8 GHz CPU and 2G memory under Debian Linux.

The following algorithms are implemented and evaluated.

1. *EXACT*: the exact algorithm proposed in Sections 3 and 4.
2. *TEXACT*: the trivial version of the exact algorithm in which the order of accessing master objects and associated objects of a given master object is randomly conducted instead of being arranged as described in Sections 3 and 4.
3. *RAND*: the randomized algorithm proposed in Section 5.
4. *TRAND*: the trivial randomized algorithm using SFS algorithm [6] to compute the skyline of each sample.

Our experiments are conducted on the real dataset and synthetic datasets.

Real datasets. We use the NBA game-by-game technique statistics from 1991 to 2005. The NBA dataset is downloaded from www.nba.com and consists of 339,721 records (instances) of 1313 players. We treat each player as an uncertain object and the records of a player as the instances of the corresponding object. Instances of an object take equal probability to appear. Three attributes are selected in our experiments: the number of points, the number of assistants and the number of rebounds. The larger the attribute values, the better.

In order to evaluate the efficiency of the random algorithm against the continuous case. We also create dataset *FC* based on real dataset[4] which records 581,012 forest land cells and their distances to nearby hydrology, roadway and fire points. We assume the distances recorded are inaccurate and follow the Uniform distribution within the range of 4 m. Clearly, the smaller distance value, the better.

Synthetic datasets. We generate discrete synthetic datasets and continuous synthetic datasets, where objects are represented by instances and PDFs, respectively.

For both kinds of datasets, the domain of each dimension is [0,1] and the dimensionality d varies from 2 to 5 with the default value 3. We first generate the centers of n uncertain objects using the benchmark data generator in [2], where n varies from 10,000 to 100,000 with the default value 10,000. Anti-correlated and independent distributions of n object centers are used in our experiments. By default, we use Anti-correlated distribution. Then, for each uncertain object we create a hyper-rectangle region where the instances or the PDF of this object appear. The edge size of the hyper-rectangle region follows a normal distribution in range [0,0.2] with expectation 0.1 and standard deviation 0.025.

For discrete synthetic datasets, the number of instances of an uncertain object follows a Uniform distribution in range [1, h] where h varies from 400 to 5000

[4] Forest cover dataset, UCI KDD Archive. http://kdd.ics.uci.edu

with the default value 600. In expectation, each object has $h/2$ instances and the total number of instances in a dataset is $hn/2$; by default, it is 3,000,000. In our experiments, two largest datasets have total instances of $\frac{5000}{2} \times 10{,}000 = 25{,}000{,}000$ and $\frac{600}{2} \times 100{,}000 = 30{,}000{,}000$, respectively.

Instances of an uncertain object follow a *Uniform* or *Zipf* distribution in our experiments. In Uniform distribution, the instances of an object are distributed uniformly in the region and have the same probability. In Zipf distribution, for an object firstly an instance u is randomly generated, the distances of other instances to u follow a Zipf distribution with $z=0.5$; the probabilities of each generated instances also follow a Zipf distribution with $z=0.2$.

Considering distributions of the object centers and the instances within an object, we have Anti-Uniform datasets where the centers of object MBBs follow Anti-correlated distribution, while instances follow Uniform distribution. Similarly, we have Inde-Uniform, Anti-Uniform, Anti-Zipf, and Inde-Zipf datasets.

For continuous synthetic datasets, the PDF of each object is *Uniform* or *Constrained-Normal* (Con-Nor for short), while other settings are generated in the same way as the discrete case. For a Uniform distribution, the PDF of an object U is a constant. The Con-Nor distribution is a Normal distribution constrained within the region (i.e., MBB) of an object U given by the formula below:

$$pdf_{CN}(x) = \begin{cases} pdf_N(x)/\lambda & \text{if } x \in U \\ 0 & \text{otherwise} \end{cases} \quad (7)$$

Here, $\lambda = \int_{x \in U} pdf_N(x)\,dx$ and the d dimensional pdf of Normal distribution $pdf_N(x) = (1/(\sqrt{2\pi\sigma^2})^d) e^{-(1/2\sigma^2)\sum_{i=1}^{d}(x.D_i - \mu.D_i)^2}$, where $\mu.D_i$ $(1 \le i \le d)$ is the coordinate of the center of U on dimension D_i and the standard deviation $\sigma = 0.025$.

We use Anti-Uniform-PDF to denote continuous synthetic datasets with Anti-correlated distribution for the centers of object MBBs and Uniform PDF for each object. Similarly, we have Anti-Con-Nor-PDF datasets.

In our experiments, k varies from 10 to 100 with the default value 30. The sample size m varies from 1000 to 2500 with the default value 1000.

Note that *in our experiments all parameters use default values unless otherwise specified*. Table 2 summaries the experiment settings.

7.1. Significance of top-k probabilistic skyline

Table 2 in [23] demonstrates that the probabilistic skyline model may capture the useful semantics which cannot be provided by the aggregation-based conventional skyline model. Nevertheless, the algorithms in [23] for computing probabilistic threshold-based skyline objects is not applicable to the computation of top-k skyline because (1) the output skyline objects by the algorithms in [23] may only have the information of lower/upper-bound probabilities and (2) there is no way to predicate the ranks of skyline probability based on the obtained lower/upper-bounds. For instance, when probabilistic threshold is set to 0.05, the lower and upper probabilistic bounds for Grant Hill are 0.0503 and 1, respectively. Although he is only ranked 17 based on his exact skyline probability (0.191164), the *11 out of the top 16* players have the output lower probabilistic bounds smaller than 0.0503 and the upper probabilistic bounds smaller than 1. This implies the techniques in [23] cannot be used to rank probabilistic skyline objects.

We further use the NBA dataset to evaluate the significance of top-k probabilistic skyline objects. The first and second columns in Table 3 display the top-25 NBA players based on the no-decreasing order of skyline probabilities. We examine another two aggregation based methods to rank NBA players. The third column ranks the NBA players based on the "skyline layers" using the average score on each dimension per player. That is, for each player, firstly calculate (average-score, average-rebound, average-assistance) and then compute skyline against (average-score, average-rebound, average-assistance) for each player. The players are on the skyline are ranked number 1. After removing the skyline players, the skyline players in the remaining players are ranked next; we continue this till obtain the top-k players. Note that in the last round, if more than k players are selected, then in the last layer, a random selection is made. Such ranks are presented in the column 3.

Table 2
Experiment settings.

Notation	Definition (default values)
n	Number of uncertain objects in the dataset (10 K)
k	Number of uncertain objects in SOUND (30)
d	Dimensionality of the dataset (3)
h	Largest number of instances in an objects (600)
m	Number of samples in randomized algorithms (1000)
$\mathcal{D}$	Dataset types (Anti-Uniform)

Table 3
Top-25 players with different ranking criteria.

Players probability	Skyline prob.	Agg. skyline	Aggregate-score
LeBron James	0.350699	1	2
Dennis Rodman	0.327592	1	79
Shaquille O'Neal	0.323401	1	1
Charles Barkley	0.309311	1	7
Kevin Garnett	0.302531	1	8
Jason Kidd	0.293569	1	30
Allen Iverson	0.269871	1	5
Michael Jordan	0.250633	1	3
Tim Duncan	0.241252	1	6
Karl Malone	0.239737	1	4
Chris Webber	0.22153	1	9
Kevin Johnson	0.208991	1	36
Hakeem Olajuwon	0.203641	21 (layer 2)	13
Kobe Bryant	0.200272	21 (layer 2)	14
Dwyane Wade	0.199065	21 (layer 2)	10
Tracy Mcgrady	0.198185	21 (layer 2)	20
Grant Hill	0.191164	1	15
John Stockton	0.183591	1	59
David Robinson	0.177437	45 (layer 3)	19
Stephon Marbury	0.16683	1	23
Tim Hardaway	0.166206	1	41
Magic Johnson	0.151813	1	52
Chris Paul	0.149264	1	37
Gilbert Arenas	0.142883	45 (layer 3)	22
Clyde Drexler	0.138993	21 (layer 2)	21

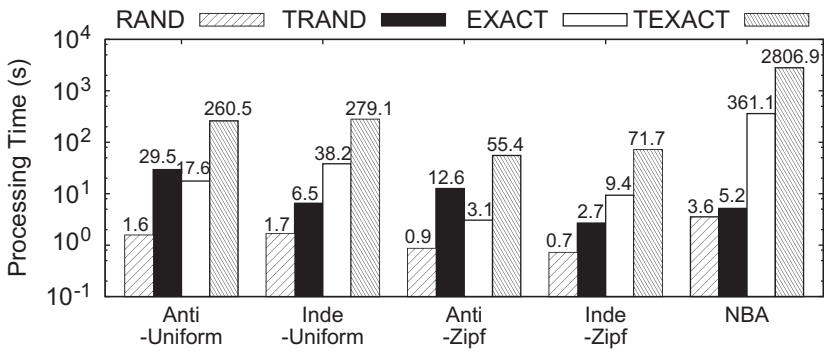

Fig. 11. Different datasets.

In column 4, we rank the NBA players according to the overall scores, average-score+average-rebound+average-assistant; the ranks are presented in column 4.

It is observed that there are too many players with the same rank for the ranking method by aggregation-based skyline layers, while our model can distinguish players well. A comparison with the aggregate-score based ranking method shows that the top-k skyline model captures well the distributions of rebounds, scores, and assists. For instance, Jason Kidd, a well-known triple double machine, is only ranked 30th according to aggregate-based model, while he is ranked 6th in the skyline probability model. The comparison with the aggregate-score-based ranking method also shows that the top-k skyline model captures well of "being not worse than any other players" regarding the rebounds, scores, and assists. For instance, Dennis Rodman, one of the best NBA rebounders, is ranked among top-2 according to our skyline probability model simply because of his consistent rebounding performance, while he is only ranked 79th in the aggregate-score-based model. These demonstrates the significance of the top-k skyline object model.

7.2. Evaluating efficiency

In this subsection, we evaluate the efficiency of our algorithms on the NBA dataset and discrete synthetic datasets. Note that we only report the performance of algorithm RAND against discrete synthetic datasets because for the same type of probability distribution of instances (e.g., instances follow Uniform distributions), RAND has almost the same performance for the discrete and continuous cases; consequently we focus on the report the performance of the *discrete* case.

We also implement the naive algorithm mentioned at the beginning of Section 3 which will compute the skyline probability of each uncertain object.[5] The naive algorithm is very computational expensive. Under default setting, it takes around 180,857 s for the Anti-Uniform dataset which is 10,275 and 113,046 times slower than that of EXACT and RAND Algorithms, respectively. So we exclude the naive algorithm in the following experiments.

[5] For comparison fairness, we only compute the skyline probability for the non-redundant uncertain objects in the implementation.

Fig. 11 shows the running time of the four algorithms on various datasets. While both EXACT and RAND are quite efficient, RAND is more efficient than EXACT on both synthetic and real datasets. EXACT performs poorly on the NBA dataset. This is because in the NBA dataset most objects are partially dominated by many others; thus many join-like operations (i.e., $ST(U,V)$) have been performed. In this case, the linear time complexity of $DC(U,V)$ of RAND shows a great advantage.

Fig. 11 also demonstrates that TEXACT is up to 9 times slower than EXACT; this shows the great advantage of developed accessing order techniques regarding associated and master objects, respectively. RAND is also significantly more efficient than the trivial randomized algorithm TRAND. Therefore, in the remaining experiment part we can exclude the performance evaluation of TEXACT and TRAND since they provide the same results as EXACT and RAND, respectively, but are much more slower.

The results of the second experiment reported in Fig. 12 demonstrates that the performance of both EXACT and RAND are quite scalable when the number of uncertain objects increases.

The third experiment reported in Fig. 13 shows that RAND is not sensitive to the number of instances (regarding the discrete case) since only a fixed number of samples are generated in randomized computation. On the other hand, the performance of EXACT drops when the number of instances grows.

Fig. 14 demonstrates the impact of k on performance. RAND is more efficient and the processing time grows much slower than EXACT because the cost for seeding the first k objects in RAND is not as dominating as that in EXACT.

Fig. 15 evaluates the impact of dimensionality. It shows that the running time of EXACT significantly decreases when d increases. This is because when d is large, the dominating relationships among objects are weak.

Fig. 16 shows that the running time of RAND increases linearly against the increment of the sample size m. This is because our counting technique runs in linear time with respect to m.

We also evaluate the performance of the RAND Algorithm against the real dataset *FC* in which k various from 20 to 100. As shown in Fig. 17, the performance of the RAND Algorithm degrades slowly against the growth of k.

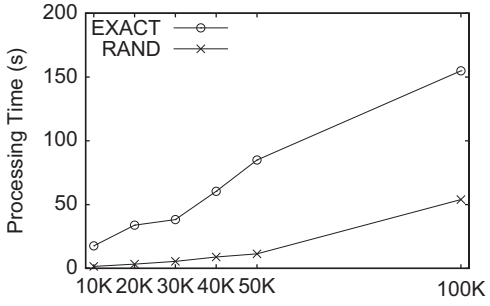

Fig. 12. Varying n.

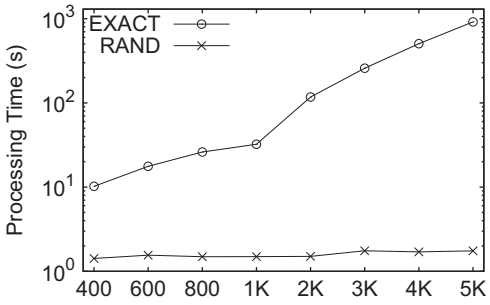

Fig. 13. Varying h.

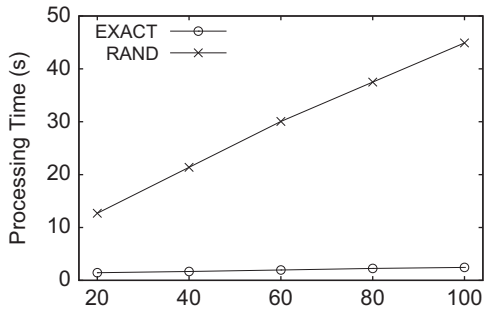

Fig. 14. Varying k.

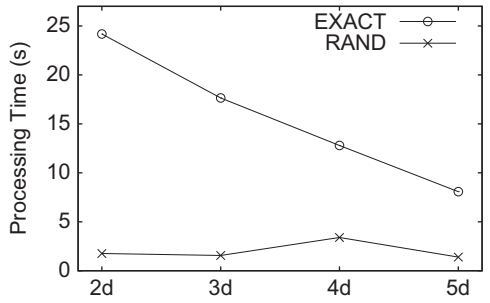

Fig. 15. Varying d.

7.3. Evaluating accuracy

We evaluate the accuracy of RAND regarding both discrete and continuous cases. We use the average *relative errors*, the average of $|P'_i - P_i|/P_i$ (for $1 \le i \le k$), to measure the accuracy, where P_i is the i-th largest skyline probability and P'_i is the estimated skyline probability of the i-th element returned by RAND.

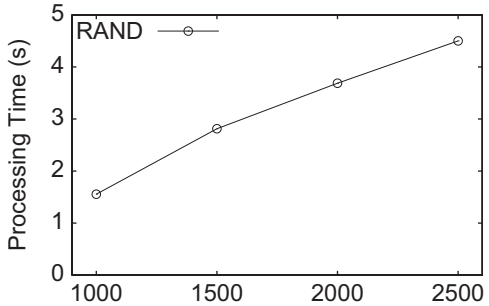

Fig. 16. Varying m.

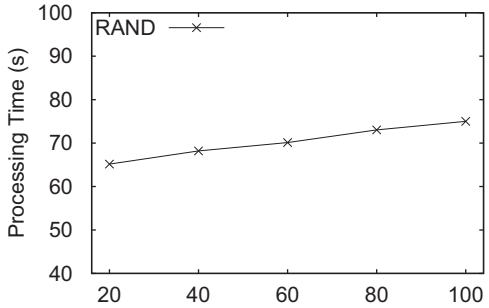

Fig. 17. Varying k for *FC* data.

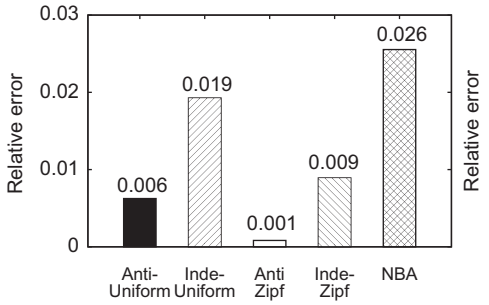

Fig. 18. Different datasets.

7.3.1. Discrete cases

We first evaluate RAND on the NBA dataset and discrete synthetic datasets.

Fig. 18 illustrates the accuracy of RAND on various datasets. It shows that $m=1000$ already gives very accurate results (average relative error <0.03). The NBA dataset has the worst performance; this is because that skyline probabilities are small due to a high overlapping degrees among MBBs of uncertain objects.

Fig. 19 reports the impact of k on accuracy. It shows that when k is not large (typical situation for a top-k problem), the accuracy is not very sensitive to k.

Fig. 20 shows that the accuracy is not quite related to the number of objects n.

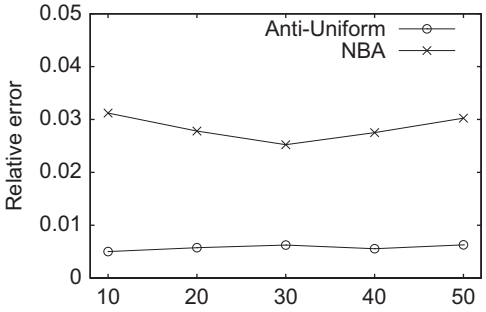

Fig. 19. Varying k.

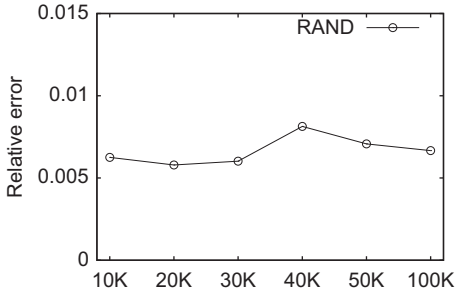

Fig. 20. Varying n.

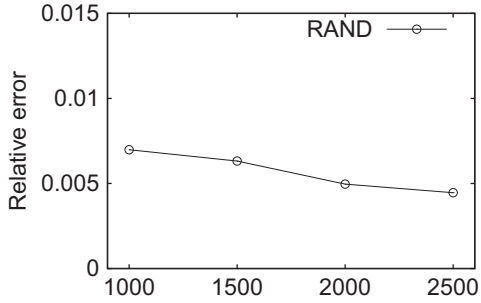

Fig. 21. Varying m.

Fig. 21 shows that the accuracy increases (i.e., relative errors decrease) when the sample size grows, as what we expected.

Fig. 22 shows that the average relative error drops (i.e., accuracy increases) quickly as the dimensionality increases. This is because the average skyline probability of objects increases when the dimensionality increases.

7.3.2. Continuous cases

RAND is also very accurate in the continuous case. We run RAND on continuous synthetic datasets with Anti-Uniform-PDF and Con-Nor-PDF, while other parameters adopt default values.

For datasets of Anti-Uniform-PDF, the skyline probability of an object is computed following Eq. (3), where all integrals can be computed as the volumes of the corresponding regions. To generate samples in RAND, the coordinates of sampled points are generated uniformly within the MBB of an object.

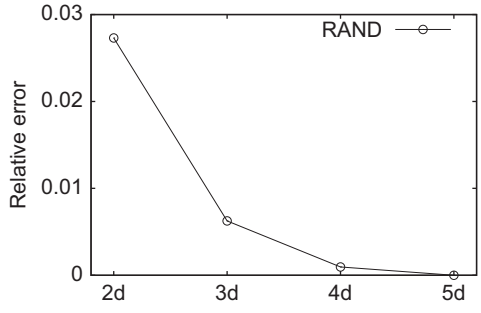

Fig. 22. Various d.

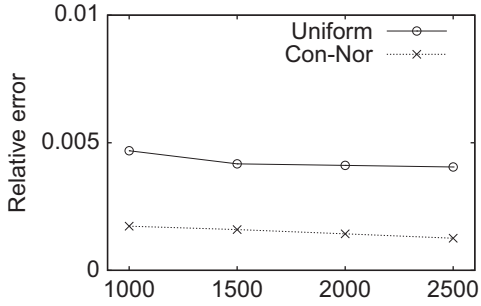

Fig. 23. Various m.

For datasets of Anti-Con-Nor-PDF, since the integral of Gaussian function cannot be evaluated exactly, each object is discretized by drawing 10,000 samples and then run the algorithm EXACT against the discretized objects to get the skyline probability of each object. To generate samples, the coordinates of sampled points are generated according to the Con-Nor distribution of each object. We use GNU Scientific Library to generate the Normal distribution and transform it into Con-Nor distribution.

The experiments reported in Fig. 23 show that the accuracy of RAND is already very high when the sample size reaches 1000; that is, the relative skyline probability error as defined in the last subsection is very low.

The experiment reported in Fig. 24 evaluates the impact of dimensionality on accuracy. The trends for both Anti-Uniform and Con-Nor-PDF are similar to that of discrete case as depicted in Fig. 22.

7.4. p-skyline computation

We run an experiment to compare the efficiency of p-EXACT and p-RAND in Section 6 with BU and TD algorithms in [23]. $p=0.9$ and the 3D Anti-Uniform datasets are employed in the experiment with the number of objects varies from 10 to 50 K and $h=600$. Fig. 25 shows that p-RAND and p-EXACT significantly outperform BU and TD. Moreover, as the number of objects grows, p-RAND and p-EXACT are quite scalable.

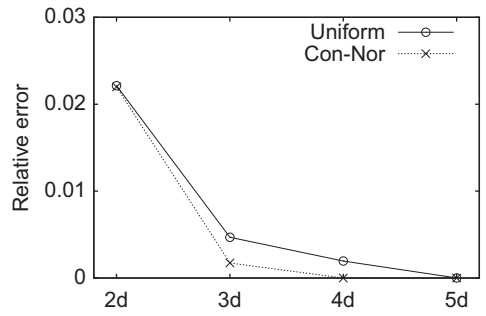

Fig. 24. Various *d*.

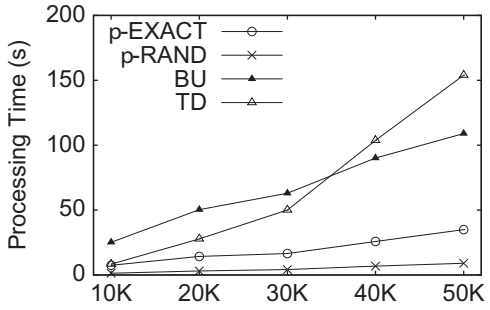

Fig. 25. Various *n*.

7.5. Summary

As a short summary, our experimental results indicate that EXACT is efficient for datasets with a medium number of instances per object. RAND is much more efficient and scalable than EXACT. Meanwhile, it provides high accuracy in both discrete case and continuous case. Moreover, applying these two algorithms to *p*-skyline problem significantly improves the efficiency of the existing *p*-skyline techniques.

8. Related work

Studies on skyline computation have a long history. Börzsönyi et al. [2] first investigate the skyline computation problem in the context of databases and propose a SQL syntax for the skyline query. They also develop the skyline computation techniques based on *block-nested-loop* and *divide-conquer* paradigms, respectively. Chomicki et al. [6] propose another block-nested-loop based computation technique, SFS (*sort-filter-skyline*), to take the advantages of a pre-sorting. The SFS paradigm is significantly improved by Godfrey et al. [9]. Tan et al. [31] propose the first *progressive* technique that can output skyline points without having to scan the whole dataset. Two auxiliary data structures are proposed, *bitmap* and *search tree*. Kossmann et al. [16] present another *progressive* technique based on the nearest neighbor search technique on *R*-tree [25,12], which adopts a divide-and-conquer paradigm on the dataset indexed by *R*-tree. Papadias et al. [21] propose a branch and bound search technique (BBS) to progressively output skyline points on datasets indexed by *R*-tree. One of the most important properties of BBS in [21] is that it guarantees the minimum I/O costs.

Recently, many variants of the skyline problem have been proposed including subspace skyline analysis [34], skyline cube [22,24,38,36], and skyline computation in data streams [19,33,26], in distributed environments [1,15,35] and in the categorical domain [26]. Moreover the skyline operator is employed as fundamental part in various applications [29,7].

Managing large volume of uncertain data has recently attracted a great deal of attention in the database community (see [8] and [27] for a survey). The uncertainty is often represented as probabilities. In spatial-temporal databases, Cheng et al. [4] study the problem of augmenting probability information to queries over uncertain data. In [5], they also develop two indexes to support answering probabilistic threshold queries, which is a range query on uncertain dataset returning objects whose probabilities of satisfying this query exceed a given threshold. Tao et al. [32] propose a U-tree to index multi-dimensional uncertain data and evaluate probabilistic range queries with arbitrary PDFs. Kriegel et al. [17] define probabilistic distance functions to measure the similarity between uncertain objects. Recently, Kriegel et al. [18] and Ngai et al. [20] tackle the problem of clustering uncertain data.

Pei et al. [23] is the work most related to our Top-*k* SOUND problem. It proposes to retrieve the *p*-skyline that consists of objects with skyline probabilities above a given threshold *p*. Two efficient algorithms, bottom-up and top-down, have been proposed. However, these two algorithms can only guarantee the objects output with the skyline probabilities not smaller than *p*. They often do not provide the exact skyline probabilities nor the ranks of objects according to skyline probabilities. Therefore, the techniques in [23] are not applicable to Top-*k* SOUND. On the other hand, a simple modification of our two algorithms can support efficient *p*-skyline computation and they significantly outperform the existing techniques in [23].

9. Conclusion

In this paper, we tackle the problem of computing the top-*k* probabilistic skyline objects on uncertain data. We employ an *R*-tree index to efficiently conduct the initial computation. Two efficient algorithms are proposed. The exact algorithm aims to precisely rank the top-*k* skyline objects against skyline probabilities. To deal with the applications where each object has a large set of instances, we develop a randomized algorithm with ϵ-approximation accuracy guarantee, together with a novel, efficient counting algorithm. These two algorithms can be immediately extended to compute *p*-skyline [23] to improve the efficiency of the existing techniques and to leading to the first work for computing *p*-skyline in the continuous case. The extensive experiments demonstrate the efficiency, scalability, and accuracy of our algorithms.

Acknowledgments

The work was supported by ARC Grants (DP0987557 and DP0881035) and Google Research Award. The research was also supported in part by a NSERC Discovery Grant and a NSERC Discovery Accelerator Supplement Grant. All opinions, finding, conclusions and recommendations in this paper are those of the authors and do not necessarily reflect the views of the funding agencies.

References

[1] W.-T. Balke, U. Güntzer, J.X. Zheng, Efficient distributed skylining for web information systems, in: EDBT, 2004.
[2] S. Börzsönyi, D. Kossmann, K. Stocker, The skyline operator, in: ICDE, 2001.
[3] C.Y. Chan, H.V. Jagadish, K.-L. Tan, A.K.H. Tung, Z. Zhang, Finding k-dominant skylines in high dimensional space, in: SIGMOD, 2006.
[4] R. Cheng, D.V. Kalashnikov, S. Prabhakar, Evaluating probabilistic queries over imprecise data, in: SIGMOD, 2003.
[5] R. Cheng, Y. Xia, S. Prabhakar, R. Shah, J.S. Vitter, Efficient indexing methods for probabilistic threshold queries over uncertain data, in: VLDB, 2004.
[6] J. Chomicki, P. Godfrey, J. Gryz, D. Liang, Skyline with presorting, in: ICDE, 2003.
[7] G. Cormode, F. Korn, S. Muthukrishnan, D. Srivastava, Summarizing two-dimensional data with skyline-based statistical descriptors, in: SSDBM, 2008.
[8] N.N. Dalvi, D. Suciu, Management of probabilistic data: foundations and challenges, in: PODS, 2007.
[9] P. Godfrey, R. Shipley, J. Gryz, Maximal vector computation in large data sets, in: VLDB, 2005.
[10] O. Goldreich, Randomized methods in computation. 〈http://www.wisdom.weizmann.ac.il/~oded/rnd.html〉.
[11] A. Guttman, R-trees: a dynamic index structure for spatial searching, in: SIGMOD Conference, 1984.
[12] G.R. Hjaltason, H. Samet, Distance browsing in spatial databases, ACM Transactions on Database Systems (1999).
[13] W. Hoeffding, Probability inequalities for sums of bounded random variables, Journal of the American Statistical Association (1963).
[14] Y.-W. Huang, N. Jing, E.A. Rundensteiner, Spatial joins using r-trees: breadth-first traversal with global optimizations, in: VLDB, 1997.
[15] Z. Huang, C.S. Jensen, H. Li, B.C. Ooi, Skyline queries against mobile lightweight devices in MANETs, in: ICDE, 2006.
[16] D. Kossmann, F. Ramsak, S. Rost, Shooting stars in the sky: an online algorithm for skyline queries, in: VLDB, 2002.
[17] H.-P. Kriegel, P. Kunath, M. Pfeifle, M. Renz, Probabilistic similarity join on uncertain data, in: DASFAA, 2006.
[18] H.-P. Kriegel, M. Pfeifle, Density-based clustering of uncertain data, in: KDD, 2005.
[19] X. Lin, Y. Yuan, W. Wang, H. Lu, Stabbing the sky: efficient skyline computation over sliding windows, in: ICDE, 2005.
[20] W.K. Ngai, B. Kao, C.K. Chui, R. Cheng, M. Chau, K.Y. Yip, Efficient clustering of uncertain data, in: ICDM, 2006.
[21] D. Papadias, Y. Tao, G. Fu, B. Seeger, An optimal and progressive algorithm for skyline queries, in: SIGMOD, 2003.
[22] J. Pei, A.W.-C. Fu, X. Lin, H. Wang, Computing compressed multidimensional skyline cubes efficiently, in: ICDE, 2007.
[23] J. Pei, B. Jiang, X. Lin, Y. Yuan, Probabilistic skylines on uncertain data, in: VLDB, 2007.
[24] J. Pei, W. Jin, M. Ester, Y. Tao, Catching the best views of skyline: a semantic approach based on decisive subspaces, in: VLDB, 2005.
[25] N. Roussopoulos, S. Kelley, F. Vincent, Nearest neighbor queries, in: SIGMOD Conference, 1995.
[26] N. Sarkas, G. Das, N. Koudas, A.K.H. Tung, Categorical skylines for streaming data, in: SIGMOD Conference, 2008.
[27] A.D. Sarma, O. Benjelloun, A.Y. Halevy, J. Widom, Working models for uncertain data, in: ICDE, 2006.
[28] M. Sharifzadeh, C. Shahabi, The spatial skyline queries, in: VLDB, 2006.
[29] M.A. Soliman, I.F. Ilyas, N. Koudas, Finding skyline and top-*k* bargaining solutions, in: ICDE, 2007.
[30] W. Son, M.-W. Lee, H.-K. Ahn, S. won Hwang, Spatial skyline queries: an efficient geometric algorithm, in: SSTD, 2009, pp. 247–264.
[31] K.-L. Tan, P.-K. Eng, B.C. Ooi, Efficient progressive skyline computation, in: VLDB, 2001.
[32] Y. Tao, R. Cheng, X. Xiao, W.K. Ngai, B. Kao, S. Prabhakar, Indexing multi-dimensional uncertain data with arbitrary probability density functions, in: VLDB, 2005.
[33] Y. Tao, D. Papadias, Maintaining sliding window skylines on data streams, IEEE Transactions on Knowledge and Data Engineering 18 (2) (2006) 377–391.
[34] Y. Tao, X. Xiao, J. Pei, SUBSKY: efficient computation of skylines in subspaces, in: ICDE, 2006.
[35] A. Vlachou, C. Doulkeridis, Y. Kotidis, M. Vazirgiannis, Skypeer: efficient subspace skyline computation over distributed data, in: ICDE, 2007.
[36] T. Xia, D. Zhang, Refreshing the sky: the compressed skycube with efficient support for frequent updates, in: SIGMOD, 2006.
[37] M.L. Yiu, N. Mamoulis, Efficient processing of top-*k* dominating queries on multi-dimensional data, in: VLDB, 2007, pp. 483–494.
[38] Y. Yuan, X. Lin, Q. Liu, W. Wang, J.X. Yu, Q. Zhang, Efficient computation of the skyline cube, in: VLDB, 2005.