Finding Probabilistic k-Skyline Sets on Uncertain Data

Jinfei Liu
Emory University
jinfei.liu@emory.edu

Haoyu Zhang
Emory University
haoyu.zhang@emory.edu

Li Xiong
Emory University
lxiong@emory.edu

Haoran Li
Emory University
haoran.li@emory.edu

Jun Luo
Lenovo; CAS
jun.luo@siat.ac.cn

ABSTRACT

Skyline is a set of points that are not dominated by any other point. Given uncertain objects, probabilistic skyline has been studied which computes objects with high probability of being skyline. While useful for selecting individual objects, it is not sufficient for scenarios where we wish to compute a *subset* of skyline objects, i.e., a skyline set. In this paper, we generalize the notion of probabilistic skyline to probabilistic k-skyline sets (Pk-SkylineSets) which computes k-object sets with high probability of being skyline set. We present an efficient algorithm for computing probabilistic k-skyline sets. It uses two heuristic pruning strategies and a novel data structure based on the classic layered range tree to compute the skyline set probability for each instance set with a worst-case time bound. The experimental results on the real NBA dataset and the synthetic datasets show that Pk-SkylineSets is interesting and useful, and our algorithms are efficient and scalable.

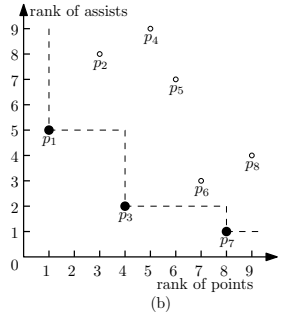
1. INTRODUCTION

The skyline of a set of multi-dimensional points consists of the points for which no other point exists that is better in at least one dimension and at least as good along every dimension. It is important for many applications involving multi-criteria decision making.

Assume that we have a dataset of n points, referred to as P. Each point p of d real-valued attributes can be represented as a d-dimensional point $(p[1], p[2], ..., p[d]) \in \mathbb{R}^d$ where $p[i]$ is the i^{th} attribute of p. Figure 1(a) illustrates a dataset of eight points $P = \{p_1, p_2, ..., p_8\}$, each representing an NBA player with two attributes: rank of points and rank of assists. Without loss of generality, we assume that small values are more preferable in this paper. Figure 1(b) shows the corresponding points in the two dimensional space where x and y coordinates correspond to the value of attributes for rank of points and rank of assists, respectively. Given two points $p = (p[1], p[2], ..., p[d])$ and $p' = (p'[1], p'[2], ..., p'[d])$ in $\mathbb{R}^d$, p dominates p' if for every i, $p[i] \leq p'[i]$ and for at least one i, $p[i] < p'[i]$ $(1 \leq i \leq d)$. We can see that point $p_3(4, 2)$ dominates point $p_6(7, 3)$ as an example of dominance. Given a set of points P, the *skyline* of P is the set of points in P that are not dominated by any other point in P. For an NBA coach who is interested in choosing an NBA player considering both rank of points and rank of assists, the skyline of Figure 1(b) includes p_1, p_3, and p_7, which offer Pareto optimal solutions with various tradeoffs between rank of points and rank of assists: p_1 is the player with best rank of points, p_7 is the player with best rank of assists, and p_3 may be a good compromise of the two factors.

player	rank of Ps	rank of As
p_1	1	5
p_2	3	8
p_3	4	2
p_4	5	9
p_5	6	7
p_6	7	3
p_7	8	1
p_8	9	4

(a)



(b)

Figure 1: A skyline example of NBA players.

Motivation. Skyline is also meaningful on uncertain data. For example, NBA players may have different performances in different games. Assume that we have four NBA players $(O_i, 1 \leq i \leq 4)$, each player O_i has two game records or instances $(I_{i,j}, j = 1, 2)$ with different probabilities as shown in Figure 2 (the eight points have the same coordinates as the points in Figure 1). There are 2^4 possible worlds that correspond to the possible states of the four uncertain objects which are shown in Table 1, each with their associated probability. For each possible world, we can compute a set of skyline instances, also shown in Table 1.

The first study on skyline queries on uncertain data by Pei et al. [23] defines the probability of an uncertain object being a skyline as the aggregated (sum) probability of the possible worlds in which any of its instances is a skyline point. It studied p-skyline problem which returns the objects with skyline probability greater than a threshold p. Zhang et al. [29] studied a similar problem, top$k$-skyline, which returns k uncertain objects with the highest skyline probabilities. For example, as shown in Figure 2, the probability of NBA player O_1 being a skyline, denoted as $P_{sky}\{O_1\}$, equals to the aggregated probability of possible worlds that $I_{1,1}$ or $I_{1,2}$ is skyline. $I_{1,1}$ is skyline in $PW_1, ..., PW_8$ and $I_{1,2}$ is skyline in PW_{15}, PW_{16}. Hence, $P_{sky}\{O_1\} = \sum_{i=1}^{8} p(PW_i) + \sum_{i=15}^{16} p(PW_i) = 0.775$, where

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Permissions@acm.org.
CIKM'15, October 19-23, 2015, Melbourne, VIC, Australia.
© 2015 ACM. ISBN 978-1-4503-3794-6/15/10...$15.00.
DOI: http://dx.doi.org/10.1145/2806416.2806452.

$p(PW_i)$ is the probability of possible world PW_i. Similarly, $P_{sky}\{O_2\} = 0.65, P_{sky}\{O_3\} = 0.625, P_{sky}\{O_4\} = 0.325$. If we set the threshold $p = 0.6$, p-skyline returns O_1, O_2, O_3. If we set $k = 2$, top$k$-skyline returns O_1, O_2.

O_i	$I_{i,j}$	$p(I_{i,j})$
O_1	$I_{1,1}$	0.7
	$I_{1,2}$	0.3
O_2	$I_{2,1}$	0.5
	$I_{2,2}$	0.5
O_3	$I_{3,1}$	0.5
	$I_{3,2}$	0.5
O_4	$I_{4,1}$	0.5
	$I_{4,2}$	0.5

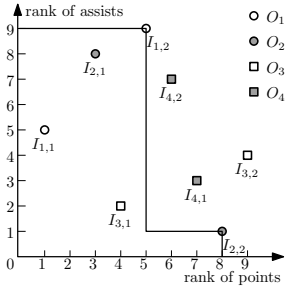


Figure 2: An example with uncertain objects.

While the above definitions compute *individual* skyline objects on uncertain data, there are scenarios where we wish to compute a *subset* of skyline objects, i.e., a skyline set. For example, an NBA coach may wish to choose a set of k best players to compose a team. Without considering uncertain case, a k-skyline set can be simply defined as any subset of k skyline points (assuming k is smaller than the number of skyline points). For uncertain data, existing works do not consider this set notion explicitly. Intuitively, we can use p-skyline or top$k$-skyline notion to choose those k objects with highest skyline probability. However, this is not suitable as each individual object is considered independently while we are more interested in the set of objects with their instances being skyline points simultaneously.

Table 1: Possible worlds semantic.

$Poss.W.$	O_1	O_2	O_3	O_4	Prob.	Skyline
PW_1	$I_{1,1}$	$I_{2,1}$	$I_{3,1}$	$I_{4,1}$	0.0875	$I_{1,1}, I_{3,1}$
PW_2	$I_{1,1}$	$I_{2,1}$	$I_{3,1}$	$I_{4,2}$	0.0875	$I_{1,1}, I_{3,1}$
PW_3	$I_{1,1}$	$I_{2,1}$	$I_{3,2}$	$I_{4,1}$	0.0875	$I_{1,1}, I_{4,1}$
PW_4	$I_{1,1}$	$I_{2,1}$	$I_{3,2}$	$I_{4,2}$	0.0875	$I_{1,1}, I_{3,2}$
PW_5	$I_{1,1}$	$I_{2,2}$	$I_{3,1}$	$I_{4,1}$	0.0875	$I_{1,1}, I_{2,2}, I_{3,1}$
PW_6	$I_{1,1}$	$I_{2,2}$	$I_{3,1}$	$I_{4,2}$	0.0875	$I_{1,1}, I_{2,2}, I_{3,1}$
PW_7	$I_{1,1}$	$I_{2,2}$	$I_{3,2}$	$I_{4,1}$	0.0875	$I_{1,1}, I_{2,2}, I_{4,1}$
PW_8	$I_{1,1}$	$I_{2,2}$	$I_{3,2}$	$I_{4,2}$	0.0875	$I_{1,1}, I_{2,2}$
PW_9	$I_{1,2}$	$I_{2,1}$	$I_{3,1}$	$I_{4,1}$	0.0375	$I_{2,1}, I_{3,1}$
PW_{10}	$I_{1,2}$	$I_{2,1}$	$I_{3,1}$	$I_{4,2}$	0.0375	$I_{2,1}, I_{3,1}$
PW_{11}	$I_{1,2}$	$I_{2,1}$	$I_{3,2}$	$I_{4,1}$	0.0375	$I_{2,1}, I_{4,1}$
PW_{12}	$I_{1,2}$	$I_{2,1}$	$I_{3,2}$	$I_{4,2}$	0.0375	$I_{2,1}, I_{3,2}, I_{4,2}$
PW_{13}	$I_{1,2}$	$I_{2,2}$	$I_{3,1}$	$I_{4,1}$	0.0375	$I_{2,2}, I_{3,1}$
PW_{14}	$I_{1,2}$	$I_{2,2}$	$I_{3,1}$	$I_{4,2}$	0.0375	$I_{2,2}, I_{3,1}$
PW_{15}	$I_{1,2}$	$I_{2,2}$	$I_{3,2}$	$I_{4,1}$	0.0375	$I_{1,2}, I_{2,2}, I_{4,1}$
PW_{16}	$I_{1,2}$	$I_{2,2}$	$I_{3,2}$	$I_{4,2}$	0.0375	$I_{1,2}, I_{2,2}, I_{4,2}$

In this paper, we extend and generalize the notion of p-skyline which computes *individual* skyline objects to probabilistic k-skyline sets (Pk-SkylineSets) which computes *sets* of skyline objects on uncertain data. Similar to p-skyline, we can define the probability of a set of objects being a skyline set as the aggregated probability of the possible worlds in which the instances of the set of objects are skyline points. We can then compute k-skyline sets with skyline set probabilities greater than a certain threshold p. Intuitively, this gives us a set of uncertain objects that have high probabilities of being skyline points simultaneously in the possible worlds. This is semantically different from choosing top$k$-skyline which consider the objects independently. For example, If we set $k = 2$, top$k$-skyline returns O_1, O_2. However, while O_1, O_2 have the highest probability of being a skyline object independently, $P_{sky}\{O_1\} > P_{sky}\{O_2\} > P_{sky}\{O_3\} > P_{sky}\{O_4\}$, the set of O_1, O_2 may not have a high probability of being a skyline set, i.e., being skyline points simultaneously in the possible worlds. In fact, if we compute the skyline set probability, $P_{sky}\{O_1, O_3\} > P_{sky}\{O_1, O_2\}$ ($0.4375 > 0.425$).

Another related notion on uncertain skyline by Liu et al. [22] is named *U-Skyline* query. U-Skyline searches for a set of objects that has the highest probability (aggregated from all possible worlds) as the answer. Back to Table 1, Set $\{I_{1,1}, I_{3,1}\}$ is the answer in PW_1 and PW_2, Set $\{I_{1,1}, I_{2,2}, I_{3,1}\}$ is the answer in PW_5 and PW_6. Those two subsets are returned as U-Skyline because they have the highest probability (0.175) aggregated from all possible worlds. Obviously, U-Skyline is not suitable for choosing a skyline set with fixed size because the results of U-Skyline can have various sizes.

In summary, if we treat the skyline in each possible world as a transaction and each skyline instance as an item, p-skyline and top$k$-skyline essentially compute the most frequent items, while U-skyline computes the most frequent transactions. In this paper, we generalize the notion of p-skyline to Pk-SkylineSets which can be considered as computing frequent itemsets of size k. Obviously, p-skyline and top$k$-skyline are special cases of our problem for $k = 1$.

Given the generalized Pk-SkylineSets problem, how can we compute the skyline sets efficiently? Assume that we have n objects, each object has m instances, we have $N = nm$ total number of instances. With the possible world semantic, a direct approach is to compute the skyline for each possible world and aggregate the results to obtain the answers. Unfortunately, we have $O(m^n)$ possible worlds, each possible world requires $O(nlogn)$ time to compute its skyline points. Then the problem is converted to finding most frequent itemsets with fixed size k from $O(m^n)$ transactions which requires $O(m^n C_n^k)$. Therefore, the total time complexity is $O(m^n nlogn + m^n C_n^k)$ which is prohibitively costly as n is an exponential term.

This motivates us to search for more efficient algorithms. Alternatively, we can enumerate all k-object set from the n objects, and compute their probability to be a skyline set. Since each object has m instances, each object set has m^k possible worlds (instance sets). In total, we need to enumerate $C_n^k m^k$ instance sets. For each instance set, we can scan all possible N instances to compute its probability to be a skyline set. Thus, a baseline implementation of this method leads to $O(C_n^k m^k kN)$ time complexity. Because $k \ll n$ in practical applications, this is much faster than the previous approach. In this paper, we investigate two efficient pruning strategies: object pruning and instance pruning, to further improve this method by reducing the search space. In addition, we propose a novel algorithm to compute the skyline set probability for each candidate instance set based on the layered range tree which achieves a worst-case time bound $O(C_n^k m^k k log^{d-1} N)$ for the entire algorithm. This improvement is significant because N is required in baseline algorithm while $nlog^{d-1}N$ is only required in the worst-case.

Contributions. We briefly summarize our contributions as follows.

- We propose a new probabilistic k-Skyline Sets notion on uncertain data, called pk-SkylineSets. It generalizes existing work for choosing individual skyline points to choosing sets of skyline points on uncertain data and is useful in finding Pareto solutions of subsets in practical applications. This is the first work to study how to choose skyline sets with fixed size on uncertain data.
- We present an efficient algorithm for computing probabilistic k-skyline sets. It includes two heuristic pruning strategies, object pruning and instance pruning,

which efficiently reduce the search space by reducing the number of candidate object sets and their instances.
- Our algorithm computes the skyline set probability for each instance set using a novel data structure based on the classic layered range tree which achieves a worst-case time bound $O(C_n^k m^k k n log^{d-1} N)$ for the entire algorithm.
- We conduct comprehensive experiments on real and synthetic datasets. Our experimental results demonstrate that the proposed algorithms are much faster than the baseline methods.

The rest of the paper is organized as follows. Section 2 presents the related work to probabilistic skyline. Section 3 formalizes the problem. Section 4 provides an overview of our proposed algorithm and the two pruning strategies. Section 5 presents the algorithm for computing skyline set probability of candidate sets. We report the experimental results and findings for performance evaluation in Section 6. Section 7 concludes the paper.

2. RELATED WORK

The problem of computing skyline (Maxima) is a fundamental problem in computational geometry field [7, 8, 17, 16, 21] because the skyline is an interesting characterization of the boundary of a set of points.

Since the introduction of the skyline operator by Börzsönyi et al. [9], skyline has been extensively studied in the database field [24, 20]. With regard to the concept of "uncertain", there are two series of works: *uncertain preferences* [26, 25, 5, 6] and *uncertain objects* [23, 18, 3, 28, 19, 1, 27, 4, 22, 15]. Our work belongs to the latter. Among these works, Pei et al. [23] introduced the first probabilistic skyline problem, p-skyline. They presented bottom-up and top-down algorithms to return those objects with skyline probability larger than the given threshold p. Lian and Chen [18] studied the monochromatic and bichromatic reverse skyline search problem over uncertain databases. Atallah and Qi [3] focused on the worst-case time complexity to compute all skyline probabilities for uncertain data, they presented an algorithm with the worst time bound of $O(N^{2-\frac{1}{d+1}} log^{\frac{d(d-1)}{d+1}} N)$ where N is the number of instances. In their journal version [4], they improved the worst-case time bound to $O(N^{2-\frac{1}{d}} log^{d-1} N)$. Independently, Afshani et al. [1] presented the algorithm with worst-case time complexity $O(N^{2-\frac{1}{d}})$. Zhang et al. [28] studied the probabilistic skyline operator over sliding windows. [19, 27] studied the problem of finding an uncertain object with the maximum expected utility. Liu et al. [22] studied the problem of searching a set of tuples that has the highest probability (aggregated from all possible worlds) as the skyline answer. Khalefa et al. [15] focused on the continuous datasets.

Our work focuses on uncertain data and is the first to study how to choose skyline sets with fixed size on uncertain data.

3. PROBLEM DEFINITIONS

In this section, we introduce some preliminary knowledge and formally define the probabilistic k-skyline sets (Pk-SkylineSets) problem.

Skyline and Skyline Sets. We start with the original skyline definition as follows.

Definition 1. (**Skyline**). Given a dataset P of n points in d-dimensional space. Let p and p' be two different points in P, we say p dominates p', denoted by $p \prec p'$, if for all i, $p[i] \leq p'[i]$, and for at least one i, $p[i] < p'[i]$, where $p[i]$ is the i^{th} dimension of p and $1 \leq i \leq d$. The skyline points are those points in P that are not dominated by any other point in P.

We extend the original skyline definition to k-skyline sets as follows.

Definition 2. (k-**SkylineSets**) A k-point set is a k-Skyline Sets (k-SkylineSets) if those k points are all skyline points.

EXAMPLE 1. *In Figure 1, $\{p_1\}$, $\{p_3\}$, and $\{p_7\}$ are 1-SkylineSets, $\{p_1, p_3\}$, $\{p_1, p_7\}$, and $\{p_3, p_7\}$ are 2-SkylineSets, and $\{p_1, p_3, p_7\}$ is a 3-SkylineSets.*

Probabilistic Data Model. Given a dataset P of n uncertain objects in d-dimensional space. Without loss of generality, we assume each object has m possible instances. If one object has m' possible instances where $m' < m$, we can assume that there are $m - m'$ instances with probability 0. We denote the i^{th} object as O_i, and the j^{th} instance of the i^{th} object as $I_{i,j}$, where $1 \leq i \leq n$ and $1 \leq j \leq m$. We use $p(I_{i,j})$ to denote the probability of instance $I_{i,j}$. We assume $\sum_{j=1}^{m} p(I_{i,j}) = 1$.

EXAMPLE 2. *Figure 2 shows a set of uncertain objects where $n = 4, m = 2, d = 2$. For example, O_1 has two possible instances $I_{1,1}$ and $I_{1,2}$ with probabilities of 0.7 and 0.3, respectively.*

Probabilistic Skyline. For uncertain data, we need to reason about the probability of an uncertain object to be a skyline object. This can be computed as the aggregated probability of each of its instances being a skyline point, since the instances are exclusive of each other. The probability of an object O_a being a skyline is,

$$P_{sky}\{O_a\} = \sum_{b=1}^{m} P_{sky}\{I_{a,b}\}.$$

where $P_{sky}\{I_{a,b}\}$ is the probability of $I_{a,b}$ being a skyline.

To compute the probability of $I_{a,b}$ being a skyline, $P_{sky}\{I_{a,b}\}$, we first define the *dominating set* which contains all instances that can dominate $I_{a,b}$, denoted as $\mathcal{DS}\{I_{a,b}\}$,

$$\mathcal{DS}\{I_{a,b}\} = \{I_{i,j(i\neq a)} \mid I_{i,j} \prec I_{a,b}; i = 1, ..., n; j = 1, ..., m\}.$$

The instances of O_l ($l \neq a$) in the dominating set of $I_{a,b}$ is denoted as $\mathcal{DS}_l\{I_{a,b}\}$,

$$\mathcal{DS}_l\{I_{a,b}\} = \{I_{l,j} \mid I_{l,j} \in \mathcal{DS}\{I_{a,b}\}; j = 1, ..., m\}.$$

Thus, we have $\mathcal{DS}\{I_{a,b}\} = \bigcup_{l=1,l\neq a}^{n} \mathcal{DS}_l\{I_{a,b}\}$. We use $Ps(S)$ to denote the aggregated probability of the instances of the same object in a set S. Therefore, the probability of $I_{a,b}$ to be skyline can be represented as follows.

$$P_{sky}\{I_{a,b}\} = p(I_{a,b}) \times \prod_{l=1,l\neq a}^{n} (1 - Ps(\mathcal{DS}_l\{I_{a,b}\})).$$

where $p(I_{a,b})$ is the probability that $I_{a,b}$ appears, $\prod_{l=1,l\neq a}^{n}(1 - Ps(\mathcal{DS}_l\{I_{a,b}\}))$ is the probability that all those instances dominate $I_{a,b}$ do not appear.

Definition 3. (p-**skyline** [23]) p-skyline are those objects $O_a, 1 \leq a \leq n$ with $P_{sky}\{O_a\} \geq p$.

EXAMPLE 3. *As shown in Figure 2, O_1 has two instances $I_{1,2}$ and $I_{1,1}$. The dominating set of $I_{1,2}$ is $\mathcal{DS}\{I_{1,2}\} = \{I_{2,1}, I_{3,1}\}$, which consists of the instance(s) of O_2, $\mathcal{DS}_2\{I_{1,2}\} = \{I_{2,1}\}$, and instance(s) of O_3, $\mathcal{DS}_3\{I_{1,2}\} = \{I_{3,1}\}$. Therefore, $I_{1,2}$ is a skyline only when $I_{1,2}$ appears and the points in $\mathcal{DS}\{I_{1,2}\}$ do not appear, with probability $P_{sky}\{I_{1,2}\} = p(I_{1,2}) \times (1-p(I_{2,1})) \times (1-p(I_{3,1})) = 0.3 \times 0.5 \times 0.5 = 0.075$. Similarly, $P_{sky}\{I_{1,1}\} = 0.7$. Therefore, the probability of O_1 being a skyline is $P_{sky}\{O_1\} = P_{sky}\{I_{1,1}\} + P_{sky}\{I_{1,2}\} = 0.775$.*

Probabilistic k-Skyline Sets. Now we generalize the notion of probabilistic skyline to probabilistic skyline sets. Let $\{O_{a_1}, ..., O_{a_k}\}$ be a set of k uncertain objects, the probability of the set being a k-SkylineSets is the aggregated probability of each of its possible instance set being a k-SkylineSets.

$$P_{sky}\{O_{a_1}, ..., O_{a_k}\} = \sum_{b_1=1}^{m} ... \sum_{b_k=1}^{m} P_{sky}\{I_{a_1,b_1}, ..., I_{a_k,b_k}\}. \quad (1)$$

Given an instance set $\{I_{a_1,b_1}, ..., I_{a_k,b_k}\}$, we can find its dominating set as follows, which is the union of the points dominating each instance in the set.

$$\mathcal{DS}\{I_{a_1,b_1}, ..., I_{a_k,b_k}\} = \{I_{i,j(i\neq a_1,...,i\neq a_k)} \mid I_{i,j} \in \mathcal{DS}\{I_{a_1,b_1}\}$$

$$\cup ... \cup \mathcal{DS}\{I_{a_k,b_k}\}; i = 1, ..., n; j = 1, ..., m\}.$$

The instance set $\{I_{a_1,b_1}, ..., I_{a_k,b_k}\}$ is a skyline set, i.e., all the instances are skyline points, only when each of the instances appears and all of the points in its dominating set do not appear. The probability can be represented as follows.

$$P_{sky}\{I_{a_1,b_1}, ..., I_{a_k,b_k}\} = p(I_{a_1,b_1}) \times ... \times p(I_{a_k,b_k}) \times$$

$$\prod_{l=1,l\neq a_1,...,l\neq a_k}^{n} (1 - Ps(\mathcal{DS}_l\{I_{a_1,b_1}, ..., I_{a_k,b_k}\})). \quad (2)$$

We note that if one instance dominates another instance within the same instance set $\{I_{a_1,b_1}, ..., I_{a_k,b_k}\}$, e.g., $I_{a_1,b_1} \prec I_{a_k,b_k}$, then $P_{sky}\{I_{a_1,b_1}, ..., I_{a_k,b_k}\} = 0$ because I_{a_1,b_1}, I_{a_k,b_k} cannot be skyline points simultaneously.

EXAMPLE 4. *Recall Figure 2, assume $k = 2$, we show how to compute $P_{sky}\{O_1, O_2\}$. We have $P_{sky}\{I_{1,1}, I_{2,1}\} = 0$ because $I_{1,1} \prec I_{2,1}$ and $P_{sky}\{I_{2,1}, I_{1,2}\} = 0$ because $I_{2,1} \prec I_{1,2}$. For the instance set $\{I_{1,2}, I_{2,2}\}$, we have its dominating set, $\mathcal{DS}\{I_{1,2}, I_{2,2}\} = \{I_{3,1}\}$ which consists of $\mathcal{DS}_3\{I_{1,2}, I_{2,2}\} = \{I_{3,1}\}$. Therefore, $P_{sky}\{I_{1,2}, I_{2,2}\} = p(I_{1,2}) \times p(I_{2,2}) \times (1 - Ps(\mathcal{DS}_3\{I_{1,1}, I_{2,2}\})) = 0.3 \times 0.5 \times 0.5 = 0.075$. Similarly, $P_{sky}\{I_{1,1}, I_{2,2}\} = 0.35$. Hence, $P_{sky}\{O_1, O_2\} = P_{sky}\{I_{1,1}, I_{2,1}\} + P_{sky}\{I_{1,1}, I_{2,2}\} + P_{sky}\{I_{1,2}, I_{2,1}\} + P_{sky}\{I_{1,2}, I_{2,2}\} = 0 + 0.35 + 0 + 0.075 = 0.425$.*

Problem Statement. We now define our problem that aims to find Pk-SkylineSets which is the k-skyline set with the highest probability of being a skyline set.

Definition 4. (**Pk-SkylineSets**) Pk-SkylineSets is the k-skyline set $\{O_{a_1}, ..., O_{a_k}\}$ with the highest probability of $P_{sky}\{O_{a_1}, ..., O_{a_k}\}$.

4. ALGORITHM OVERVIEW

To solve the probabilistic k-Skyline Sets problem, intuitively, we can enumerate C_n^k candidate k-object sets, and compute their skyline probability as defined in Equation 1. In order to compute the skyline probability for each k-object set, we need to enumerate m^k instance sets since each object has k possible instances. For each instance set, we can scan all possible N instances to compute its probability to be a skyline set as in Equation 2. Thus, a baseline implementation of this method leads to $O(C_n^k m^k k N)$ time complexity.

Algorithm 1 Computation Pk-SkylineSets

```
1: object pruning (see Section 4.1).
2: instance pruning (see Section 4.2).
3: for each k-object set {O_{a_1}, ..., O_{a_k}} chosen from remaining objects do
4:     for each possible instance set {I_{a_1,b_1}, ..., I_{a_k,b_k}} of the k-object set do
5:         compute P_sky{I_{a_1,b_1}, ..., I_{a_k,b_k}} (see Section 5).
6:     end for
7:     compute P_sky{O_{a_1}, ..., O_{a_k}} as in Equation 1.
8: end for
9: return the k-object set with highest probability P_sky.
```

We propose an efficient algorithm for computing probabilistic k-skyline sets. It includes two heuristic pruning strategies, object pruning and instance pruning, which efficiently reduce the search space for candidate object sets (C_n^k) and possible instance sets of each object set (m^k). The algorithm then iterate through all candidate object sets from the remaining objects, and compute its skyline probability by enumerating the possible instance sets from remaining instances and computing its skyline probability. It uses a novel algorithm for computing the skyline set probability for each instance set based on the layered range tree with worst-case time bound $O(knlog^{d-1}N)$. The sketch of the algorithm is shown in Algorithm 1. Overall, our algorithm achieves a worst-case time bound $O(C_n^k m^k knlog^{d-1}N)$ while practically it can be much more efficient.

Next we present the pruning strategies in detail. In the remainder of this section, we assume that no two points in P have the same coordinates. This restriction is not very realistic, but it can be overcome with a nice trick [12].

4.1 Object Pruning

We first observe an important property of the skyline probability. Similar to the apriori property in frequent itemset mining [2], the skyline probability of an object set never exceeds the skyline probability of its subsets. We present it as a theorem below.

THEOREM 1. ***(Monotone Property)** Given any two object (instance) subsets $\mathcal{S}_1$ and $\mathcal{S}_2$ of P, and $\mathcal{S}_1 \subseteq \mathcal{S}_2$, we have $P_{sky}\{\mathcal{S}_1\} \geq P_{sky}\{\mathcal{S}_2\}$.*

PROOF. We prove this theorem from possible worlds semantic. For each possible world i, $P_{sky}\{\mathcal{S}_1\}_i \geq P_{sky}\{\mathcal{S}_2\}_i$, where $P_{sky}\{\mathcal{S}\}_i$ is the probability of $\mathcal{S}$ being a SkylineSets in possible world i. Therefore, we can obtain $P_{sky}\{\mathcal{S}_1\} \geq P_{sky}\{\mathcal{S}_2\}$ by aggregating all possible worlds. □

Based on this property, the basic idea of our pruning strategy is to find an object set $\{O_{i1}, O_{i2}, ..., O_{ik}\}$ and use it to prune object O_i if $P_{sky}\{O_i\} < P_{sky}\{O_{i1}, O_{i2}, ..., O_{ik}\}$. The reason is that any superset of O_i will have skyline probability $\leq P_{sky}\{O_i\}$, hence cannot exceed $P_{sky}\{O_{i1}, O_{i2}, ..., O_{ik}\}$. The next question is how to choose the set $\{O_{i1}, O_{i2}, ..., O_{ik}\}$.

The higher $P_{sky}\{O_{i1}, O_{i2}, ..., O_{ik}\}$, the more objects can be pruned. To this end, our idea is to compute the skyline probability for each single object O_i and select the k objects with highest $P_{sky}\{O_i\}$. The intuition is that this set will likely have a high skyline set probability, although by no means it is guaranteed to have the highest probability. We summarize the pruning process in Algorithm 2. This pruning strategy proves to be very beneficial in practice as we show later in our experiments. As we recall the overall time complexity $O(C_n^k m^k nlogN)$, it is easy to see the factor C_n^k has a large impact because $O(C_n^k) \approx O(n^k)$. By reducing the number of objects (n), although without worst-case guarantee, it can improve the time efficiency significantly in practice.

Algorithm 2 Object pruning

```
1: for i=1 to n do
2:    compute P_sky{O_i}.
3: end for
4: choose k objects, {O_i1, ..., O_ik}, with highest P_sky{O_i}, compute
   P_sky{O_i1, ..., O_ik}.
5: for i=1 to n do
6:    if P_sky{O_i} < P_sky{O_i1, ..., O_ik} then
7:       delete O_i directly.
8:    end if
9: end for
```

EXAMPLE 5. *Recall the example in Figure 2, we can compute* $P_{sky}\{O_4\} = 0.325$ *and* $P_{sky}\{O_1, O_2\} = 0.425$. *Hence,* O_4 *can be pruned directly.*

4.2 Instance Pruning

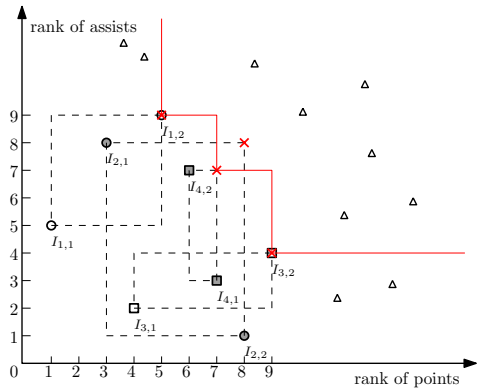


Figure 3: Instance pruning example.

The intuition of the instance pruning strategy is that if an instance is dominated by all possible instances of another object, then we can prune it directly. The reason is that the probability of it being a skyline point is 0, and hence the probability of any of its superset being a skyline set is also 0 based on the monotone property. The challenge is that it is time consuming to check whether an instance is dominated by all possible instances of each object. Hence, our idea is to compute a minimum bounding rectangle for all possible instances of each object. Then if an instance is dominated by its maximum corner, it is dominated by all the instances in the box. Formally, given an uncertain object O_l and let $I_{l_{max}} = (max_{j=1}^m\{I_{l,j}[1]\}, ..., max_{j=1}^m\{I_{l,j}[d]\})$ be the maximum corner of the minimum bounding rectangle of O_l. Note that $I_{l_{max}}$ is not necessarily an actual instance of O_l. In this case, we treat it as a virtual instance.

THEOREM 2. *For any instance* $I_{i,j}, 1 \le i \le n, 1 \le j \le m$, *if it can be dominated by any of* $I_{l_{max}}, l \ne i, 1 \le l \le n$, *then* $P_{sky}\{I_{i,j}\} = 0$.

PROOF. $P_{sky}\{I_{i,j}\} = p(I_{i,j}) \times \prod_{l \ne i, l=1}^{n}(1 - Ps(\mathcal{DS}_l\{I_{i,j}\}))$. Because $I_{l_{max}} \prec I_{i,j}$, i.e., $I_{l,1} \prec I_{i,j}$, $I_{l,2} \prec I_{i,j}$, ..., $I_{l,m} \prec I_{i,j}$. Thus, $Ps(\mathcal{DS}_l\{I_{i,j}\}) = 1$. Therefore, $P_{sky}\{I_{i,j}\} = 0$. □

Together with Theorem 1, the probability of any instance set containing $I_{i,j}$ being a skyline set is also 0. Hence, we can prune $I_{i,j}$ directly. For example, in Figure 3, we compute the maximum corners of the minimum bounding box of O_1, O_2, O_3, O_4, which are shown as crosses. Those instances on the upper right of the four crosses can be deleted directly.

5. COMPUTING SKYLINE SET PROBABILITY FOR INSTANCE SET

The key step of Algorithm 1 is to compute the skyline set probability $P_{sky}\{I_{a_1,b_1}, ..., I_{a_k,b_k}\}$ given an instance set $\{I_{a_1,b_1}, ..., I_{a_k,b_k}\}$. A naive way is to scan all remaining $O(N)$ instances after pruning to find the dominating set of the instance set which takes $O(kN)$ time. In this section, we present a novel algorithm based on the layered range tree which achieves $O(knlog^{d-1}N)$ worst-case time bound.

It is easy to see that the key to the probability computation is Equation 2, i.e., to find those instances in the range that can dominate any of the instances in $\{I_{a_1,b_1}, ..., I_{a_k,b_k}\}$. Going back to Figure 2, and take $\{I_{1,2}, I_{2,2}\}$ as an example. There are three instances $I_{1,1}, I_{2,1}, I_{3,1}$ that can dominate $\{I_{1,2}, I_{2,2}\}$. Since $I_{1,2}$ and $I_{2,2}$ exclude $I_{1,1}$ and $I_{2,1}$ respectively, we have the dominating set $\mathcal{DS}\{I_{1,2}, I_{2,2}\} = I_{3,1}$. To summarize, the key operation here is to retrieve all instances from the range shown with bold line. In order to find the dominating instances efficiently, we use a range query data structure, layered range tree, as an index to enhance the search process. In addition, we augment the layered range tree with cumulative information in order to compute the skyline set probability efficiently.

We first provide a brief description of the layered range tree in Section 5.1. We show how to build an augmented cumulative layered range tree in two dimensional space with time complexity $O(NlogN)$ in Section 5.2. We then show how to compute the skyline set probability for each instance set in $O(knlogN)$ in Section 5.3. Finally, we show how to extend it to higher dimensional space in Section 5.4. The reasons why we employ layered range tree rather than R-tree [14] are: i) R-tree cannot provide good worst-case guarantee, ii) it is not easy to adapt R-tree to report the cumulative information which is required for the probability computation due to the irregularity of MBRs.

5.1 Layered Range Tree

Layered Range Tree [12]. Let P be a set of N points in d-dimensional space. A layered range tree for P can be constructed in $O(Nlog^{d-1}N)$ time. With this layered range tree one can report the points in P that lie in a rectangular query range in $O(log^{d-1}N + v)$ time, where v is the number of reported points.

The layered range tree of Figure 1 is shown in Figure 4. The main tree is essentially a binary search tree based on the x-coordinate of the points. Each node n_i has an associated structure a_i. The detailed associated structure of Figure 4 implemented by the **fractional cascading** technique

[11, 10] is shown in Figure 5. Each subset of P associated with a node is stored in an array which is sorted on the y-coordinate of the points. Each element in the associated structure $a(v)$ of node v stores a point and two pointers: a pointer to the associated structure of the left child of node v, denoted by $a(lc(v))$, and a pointer to the associated structure of the right child of node v, $a(rc(v))$. In more detail, assume that the $(i+1)^{th}$ element of $a(v)$, $a(v)[i]$, stores a point $p(p_x, p_y)$. Then we store a pointer from $a(v)[i]$ to the element of $a(lc(v))$ storing $p'(p'_x, p'_y)$ such that p'_y is the smallest one that is larger than or equal to p_y. The pointer to $a(rc(v))$ is defined in the same way: it points to the element of $a(rc(v))$ storing $p''(p''_x, p''_y)$ such that p''_y is the smallest one that is larger than or equal to p_y.

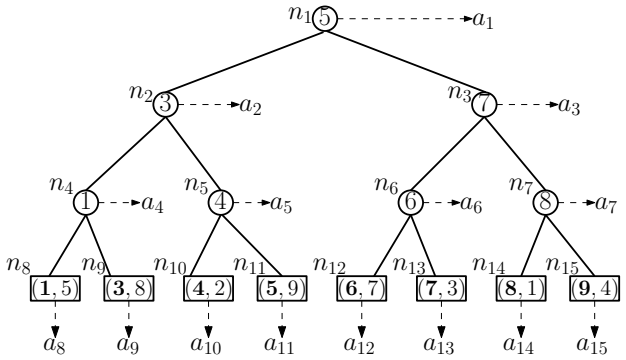


Figure 4: A layered range tree example.

EXAMPLE 6. *The fractional cascading technique is shown in Figure 5. The 5^{th} element of the associated structure a_1 of node n_1, denoted by $a_1[4]$, has two pointers. Its left pointer points to element $a_2[1]$ in the associated structure a_2 of node n_2 (the left child of n_1) because the y-coordinate of the point p stored in $a_2[0]$, denoted by $a_2[0].p_y$, we have $a_2[0].p_y < a_1[4].p_y$ $(2 < 5)$ and $a_2[1].p_y = a_1[4].p_y$ $(5 = 5)$. Its right pointer points to element $a_3[3]$ in the associated structure a_3 of node n_3 (the right child of n_1) because $a_3[2].p_y < a_1[4].p_y$ $(4 < 5)$ and $a_3[3].p_y > a_1[4].p_y$ $(7 > 5)$.*

Given a range query on layered range tree, we go through the main tree to find all the nodes that are associated with the range on the x-coordinate. And then, for the corresponding associated structures of those nodes, we find those points in the range by the pointers of fractional cascading technique based on the y-coordinate.

5.2 Building Cumulative Layered Range Tree

Given an instance $I_{i,j}$, we know that with layered range tree, we can retrieve the points (instances) in P that lies in a rectangular query range, i.e., the dominating instances of $I_{i,j}$, in $O(logN+v)$ time, where N is the number of instances and v is the number of reported instances. Unfortunately, v can be as large as N which leads to $O(N)$ time complexity, i.e., no improvement for the worst-case time complexity because we can achieve $O(N)$ by scanning all the N instances. Analyzing the expression of $O(logN + v)$, part $logN$ is required because we need to scan $O(logN)$ nodes on the tree, part v is required if we need to report all the points in the rectangular range individually. Fortunately, we do not need to retrieve those v points individually, we just need to know the cumulative probability information of those v points, in order to compute the skyline set probability. This is similar to range count, in which case, we do not need to report all v points individually, we only need to know the cumulative count, i.e., how many points there are. If we could build an index structure that can report those cumulative information in $O(logN)$ time, then the range query can be finished in $O(logN)$ time. And our goal is to build such an index structure in $O(NlogN)$ time. In the follows, we show how the accumulative information is defined and then show how to build the accumulative layered range tree with such accumulative information.

Recall the query processing of layered range tree. For any query range $[x, x'], [y, y']$, we need to search those nodes in $[x, x']$ based on the main tree in $O(logN)$ time. Then we need to search $[y, y']$ in those $O(logN)$ associated structures. Hence, if those associated structures record some cumulative information and it can be obtained in constant time, then we can finish the entire search in $O(logN)$ time. Unfortunately, for each associated structure with N' elements and a random range $[y, y']$, there are $O(N'^2)$ different combinations of elements corresponding to different combinations of y and y' for which we need to store the cumulative information. It is both time and space consuming to compute and store such information. Fortunately, in our problem as we will show later, we can always decompose the query range into several $[x, x'], [0, y']$ ranges. Given a $[0, y']$ range, there are only $O(N')$ different combinations of the elements corresponding to different y' for which we need to store the cumulative information. We formally define the *cumulative information* of each element as follows.

Definition 5. (**Cumulative Information**). For the $(j+1)^{th}$ element of i^{th} associated structure a_i, denoted by $a_i[j]$, its corresponding cumulative information $C(a_i[j])$ is an array of n elements (recall n is the number of uncertain objects) recording the aggregated probability of the points (instances) stored at $a_i[0], a_i[1], ..., a_i[j]$. The l^{th} element of the accumulative information array records the aggregated probability of those instances corresponding to object O_l.

EXAMPLE 7. *The associated structure of the cumulative layered range tree of Figure 2 is shown in Figure 5. Each element in the associated structure stores an accumulative information array. In the root associated structure a_1, for the element of $I_{2,2}$, i.e., $a_1[0]$, we store $[0, 0.5, 0, 0]$ because probability of $I_{2,2}$, $p(I_{2,2}) = 0.5$. For the element of $I_{3,1}$, i.e., $a_1[1]$, we store $[0, 0.5, 0.5, 0]$ because it records the aggregated probability of two instances $I_{2,2}$ and $I_{3,1}$.*

In our accumulated layered range tree, in addition to storing the point and the pointers in each element of the associated structure, we store the accumulative information array in this element.

We now show how to build the accumulative layered range tree. The detailed algorithm is shown in Algorithm 3. Line 1-10 is similar to constructing layered range tree which can be finished in $O(NlogN)$ time. Line 1 constructs the associated structure, such as a_1 in Figure 5. Line 2 determines the terminal condition. P is split into two subsets P_{left} and P_{right} by x_{mid} in Line 5. Line 6 and 7 recursively compute the cumulative layered range tree of P_{left} and P_{right}. Line 8 creates the main tree by x_{min}, such as n_1 in Figure 4. Fractional cascading technique is used in Line 9. The cumulative information of each element is computed in Line 11-15. For the cumulative information of each element $a_i[j]$, we need to add the probability of the instance $I_{l,k}$ stored in

Algorithm 3 Build 2D Cumulative Layered Range Tree(P)

1: construct the associated structure: build an array $a(v)$ which is sorted on the y-coordinates of the points in P.
2: **if** P contains only one point **then**
3: create a leaf v storing this point, and make $a(v)$ the associated structure of v;
4: **else**
5: split P into two subsets; one subset P_{left} contains the points with x-coordinate less than or equal to x_{mid}, the median x-coordinate, and the other subset P_{right} contains the points with x-coordinate larger than x_{mid};
6: $v_{left} \leftarrow$ Build2DCumulativeLayeredRangeTree(P_{left});
7: $v_{right} \leftarrow$ Build2DCumulativeLayeredRangeTree(P_{right});
8: create a node v storing x_{mid}, make v_{left} the left child of v, make v_{right} the right child of v, and make $a(v)$ the associated structure of v;
9: add pointers from $a(v)$ to $a(v_{left(right)})$ by fractional cascading technique as we described in Section 5.1;
10: **end if**
11: **for** each associated structure **do**
12: **for** each element of associated structure **do**
13: compute cumulative information;
14: **end for**
15: **end for**

element $a_i[j]$ to the l^{th} element (corresponding to object O_l) of the cumulative information of $a_i[j-1]$. There are in total $O(NlogN)$ elements in the associated structures. Therefore, we can compute the cumulative information in $O(NlogN)$ time. Because Line 1-10 can be finished in $O(NlogN)$, we obtain a theorem as follows.

THEOREM 3. *Algorithm 3 can be finished in $O(NlogN)$ time.*

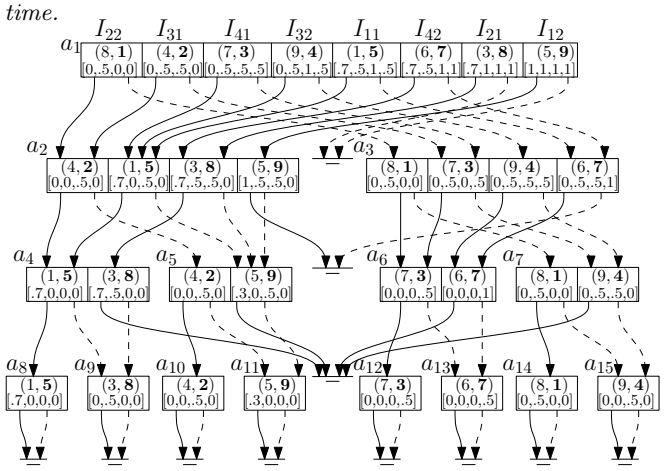


Figure 5: A cumulative layered range tree example.

5.3 Cumulative Layered Range Tree Queries

Given the cumulative layered range tree built in Section 5.2, we illustrate how to compute the skyline set probability $P_{sky}\{I_{a_1,b_1}, ..., I_{a_k,b_k}\}$ in $O(knlogN)$ time. We first show how to query the cumulative information of a range $[x, x'], [0, y']$ in $O(logN)$ time in Algorithm 4. It works by finding the nodes in the main tree corresponding to the range $[x, x']$, and then reporting the accumulative information in the associated structures corresponding to the range $[0, y']$. We search the first node v_{split} such that $x \leq v_{split} < x'$ in Line 1. If v_{split} is a leaf node, we report its cumulative information. Otherwise, we follow the path from $lc(v_{split})$ to x and report the cumulative information associated with its right subtrees ($\geq x$). For each node v on the path, if $x \leq x_v$, we report the cumulative information of right child of v and make left child of v be the new v. Otherwise, we make right child of v be the new v. If v is a leaf, we just report the cumulative information. Then similarly, we follow the path from $rc(v_{split})$ to x' and report the cumulative information associated with its left subtrees ($< x'$).

Algorithm 4 2D Cumulative Layered Range Query

1: from root, find the first node v_{split} that $x \leq v_{split} < x'$.
2: **if** v_{spilt} is a leaf **then**
3: report the cumulative information;
4: **else**
5: //follow the path to x and report the cumulative information in right subtrees;
6: $v \leftarrow lc(v_{split})$;
7: **while** v is not a leaf **do**
8: **if** $x \leq x_v$ **then**
9: report the cumulative information of $rc(v)$;
10: $v \leftarrow lc(v)$;
11: **else**
12: $v \leftarrow rc(v)$;
13: **end if**
14: **end while**
15: check if the cumulative information stored at v must be reported;
16: similarly, follow the path from $rc(v_{split})$ to x', report the cumulative information in left subtrees, and check if the cumulative information stored at the leaf where the path ends must be reported.
17: **end if**

From the root to leaves, we need to scan $O(logN)$ nodes: $O(logN)$ to x and $O(logN)$ to x'. To report the cumulative information corresponding to $[0, y']$, we need to be a bit careful. If we report cumulative information of each node by binary search to determine y' individually, it requires $O(logN)$ for each node. However, by the technique of fractional cascading, we can use binary search to find the cumulative information in root node, and use the pointers to find the cumulative information in its children and grandchildren in constant time. Therefore, we obtain a theorem as follows.

THEOREM 4. *Algorithm 4 can be finished in $O(logN)$ time.*

Given an instance $I_{a,b}$, we can query the cumulative information of those instances in a rectangular range that dominates $I_{a,b}$ using the cumulative layered range tree as above. After obtaining those $O(logN)$ cumulative information arrays each with n elements, we sum those $O(logN)$ arrays, the l^{th} element is the sum of all l^{th} elements of the $O(logN)$ arrays, which is equal to $Ps(\mathcal{DS}_l\{I_{a,b}\})$ used in Equation 2, the aggregated probability of all instances corresponding to object O_l that dominate $I_{a,b}$. We can then use Equation 2 to compute the skyline set probability $P_{sky}\{I_{a,b}\}$. Therefore, we can obtain the theorem as follows.

THEOREM 5. *Given any instance $I_{a,b}$, $P_{sky}\{I_{a,b}\}$ can be computed in $O(nlogN)$ time.*

EXAMPLE 8. *We show how to compute $P_{sky}\{I_{1,2}\}$. Because the coordinate of $I_{1,2}$ is $(5, 9)$, we set the query range as $[0, 5), [0, 9)$. First, we need to query $[0, 5)$ in the main tree of Figure 4. We determine v_{split} by a binary search algorithm, which is node n_2. Because 0 is smaller than the x-coordinate of node $lc(n_2)$, i.e., 1. We report the cumulative information of the associated structure of rc(n2) which is a_9. Follow the algorithm, we need to report the cumulative information of associated structure a_9, a_8, a_{10}.*

Follow the pointers from element of a_1 to associated structure a_9, a_8, a_{10}, we determine the cumulative information, $[0, 0.5, 0, 0]$, $[0.7, 0, 0, 0]$, and $[0, 0, 0.5, 0]$, respectively. The sum is $[0.7, 0.5, 0.5, 0]$. Therefore, $P_{sky}\{I_{1,2}\} = p(I_{1,2}) \times (1 - 0.5) \times (1 - 0.5) \times (1 - 0) = 0.075$.

Given an instance set, we need to decompose the dominating range into multiple rectangular ranges with the requirement that the range of y coordinates in each rectangle begins with 0. For k instances $I_{a_1,b_1}, ..., I_{a_k,b_k}$, without lose of generality, assume $a_1 < ... < a_k$, then it is easy to see $b_1 > ... > b_k$, otherwise, some point of those k points will dominate another one. We decompose the dominating range into k query ranges as $[0, a_1), [0, b_1)$ then $(a_1, a_2), [0, b_2)$ and so on. Therefore, we can query the cumulative information associated with each range to compute the skyline set probability. We can obtain a theorem as follows.

THEOREM 6. *Given any instance set $\{I_{a_1,b_1}, ..., I_{a_k,b_k}\}$, $P_{sky}\{I_{a_1,b_1}, ..., I_{a_k,b_k}\}$ can be computed in $O(knlogN)$ time.*

EXAMPLE 9. *We show how to compute $P_{sky}\{I_{2,1}, I_{4,1}\}$. Because the coordinates of $I_{2,1}$, $I_{4,1}$ are (3,8), (7,3), respectively. We decompose the query range as [0,3), [0,8) and (3,7), [0,3). For [0,3), [0,8) and (3,7), [0,3), we obtain the cumulative information [0.7,0,0,0], [0,0,0.5,0], respectively. Therefore, $P_{sky}\{I_{2,1}, I_{4,1}\} = (1 - 0.7) \times p(I_{2,1}) \times (1 - 0.5) \times p(I_{4,1}) = 0.0375$.*

5.4 Higher Dimensional Space

It is fairly straightforward to generalize two dimensional cumulative layered range tree to higher dimensional space. Let P be a set of N points in d-dimensional space. The first $d-2$ dimensions can be processed by traditional layered range tree and the last two dimensions can be handled by the algorithms shown in Section 5.2 and 5.3. Detailed discussion is omitted due to the limited space.

6. EXPERIMENTS

In this section, we first study the effectiveness of Pk-SkylineSets on real NBA dataset and then perform an extensive empirical study to examine the heuristic pruning strategies and the skyline set probability computation algorithm based on layered range tree on synthetic and real datasets. Since this is the first work for k-skyline sets on uncertain data, our performance evaluation was conducted against baseline only. We implemented the following algorithms.

- **BL**: **B**ase**L**ine algorithm with worst-case time complexity $O(C_n^k m^k kN)$.
- **RT**: The algorithm based on cumulative layered **R**ange **T**ree index structure with worst-case time complexity $O(C_n^k m^k knlog^{d-1}N)$ without pruning.
- **HBL**: We first use the two **H**euristic pruning strategies and then use **B**ase**L**ine algorithm.
- **HRT**: This is our complete algorithm. We first use the two **H**euristic pruning strategies and then use our algorithm based on cumulative layered **R**ange **T**ree index structure.

6.1 Experiment Setup

We implemented all algorithms in C++ and ran experiments on a machine with Intel Core i7 running Ubuntu with 8GB memory. For the layered range tree index structure, we adapted the implementation of [13]. We used both real NBA dateset and synthetic datasets in our experiments.

6.2 Effectiveness of Pk-SkylineSets

We first demonstrate the effectiveness of Pk-SkylineSets through a small real NBA dataset. We chose 42 players with the highest skyline probability in Table 2 of Pei et al. [23]. We downloaded their recent ten years' statistics from http://www.basketball-reference.com. Please note that the career years of some players are less than ten years and some players stopped their career several years ago. We treat each player as an uncertain object and the annual records of each player as the instances of the object. Three attributes are selected in our experiment: points (PTS), assists (AST), and rebounds (REB). The larger those attribute values, the better. We show how to choose five players to compose a "gold team". Therefore, in this NBA dataset, $n = 42, m = 10, d = 3, k = 5$.

Table 3: Skyline probability of sets.

ID	Set	Prob.	ID	Set	Prob.
0	{0,1,3,9,21}	0.597696	4	{0,3,9,10,21}	0.431482
1	{0,1,3,10,21}	0.488009	5	{0,3,7,9,21}	0.431089
2	{0,1,3,7,21}	0.479059	6	{0,1,7,9,21}	0.426672
3	{0,1,9,10,21}	0.437064	7	{1,3,7,9,21}	0.42394

The skyline probabilities of the individuals and the sets are shown in Table 2 and 3, respectively. Table 2 shows the skyline probabilities of those 42 players. Because we only consider 42 players while [23] considered 1313 players, the skyline probabilities as well as the resulting p-skyline players in our experiment are much higher than and different from those in Table 2 of [23]. For example, Magic Johnson has the highest probability in our example because he did very well in his last ten years career while not so good between 1991 and 2005 (stopped in 1996). The probabilities of the sets being a skyline set are shown in Table 3. Since there are $C_{42}^5 = 850668$ candidate sets from the 42 players, we only show the first 8 sets with highest probabilities. We can see that the first five players ({0,1,3,9,21}) with highest skyline probability in Table 2 compose the Pk-SkylineSets with highest skyline set probability. However, $P_{sky}\{0, 1, 3, 10, 21\}$ is higher than $P_{sky}\{0, 1, 3, 7, 21\}$ while the only differing player Michael Jordan (ID=7) has higher skyline probability than Chris Webber (ID=10). The results verify that the set of objects with higher individual skyline probabilities does not necessarily lead to higher skyline set probability.

6.3 Efficiency and Scalability

We use both real NBA dataset and synthetic datasets to study the efficiency and scalability of our methods. We downloaded recent ten years' statistics of 1000 players from http://www.basketball-reference.com/ with two more attributes (steals (STL) and blocks (BLK)) than Section 6.2, which composes the NBA dataset. We also generated independent (INDE), correlated (CORR), and anti-correlated (ANTI) datasets following the seminal work of [9]. For the uncertain case, we generalized uncertain instances following the method of [23].

We first study the efficiency of the four algorithms on INDE dataset (m=10,k=5,d=5). In Figure 6, we show the time cost for BL, RT, HBL, and HRT on different number of objects n. Although both RT and BL are fairly slow, RT is significantly faster than BL, which validates the efficiency of our probability computation algorithm based on layered range tree. HBL and HRT are much faster than BL and RT, which validates the benefit of our pruning strategies. In the

Table 2: Skyline probability of individuals.

ID	Name	Probability	ID	Name	Probability	ID	Name	Probability
0	LeBron James	0.905062	14	Dwyane Wade	0.358879	28	Steve Francis	0.10701
1	Dennis Rodman	0.839869	15	Tracy Mcgrady	0.265222	29	Dirk Nowitzki	0.063802
2	Shaquille O'Neal	0.453626	16	Grant Hill	0.027655	30	Paul Pierce	0.012171
3	Charles Barkley	0.693831	17	John Stockton	0.108985	31	Gary Payton	0.390831
4	Kevin Garnett	0.400346	18	David Robinson	0.307053	32	Baron Davis	0.150622
5	Jason Kidd	0.393144	19	Stephon Marbury	0.202866	33	Vince Carter	0.002854
6	Allen Iverson	0.440725	20	Tim Hardaway	0.291563	34	Antoine Walker	0.187002
7	Michael Jordan	0.609803	21	Magic Johnson	0.928804	35	Steve Nash	0.118144
8	Tim Duncan	0.049988	22	Chris Paul	0.327774	36	Andre Miller	0.003283
9	Karl Malone	0.664875	23	Gilbert Arenas	0.120904	37	Isiah Thomas	0.219151
10	Chris Webber	0.567586	24	Clyde Drexler	0.188748	38	Elton Brand	0.011283
11	Kevin Johnson	0.328127	25	Patrick Ewing	0.136455	39	Scottie Pippen	0.081097
12	Hakeem Olajuwon	0.357442	26	Rod Strickland	0.073434	40	Dominique Wilkins	0.209298
13	Kobe Bryant	0.345385	27	Brad Daugherty	0.098185	41	Lamar Odom	0.157486

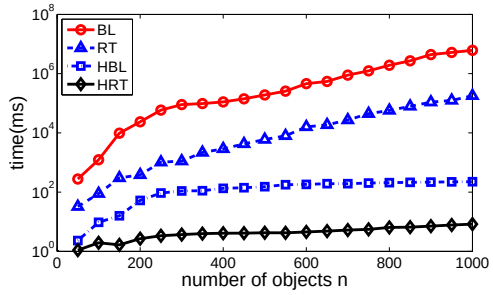
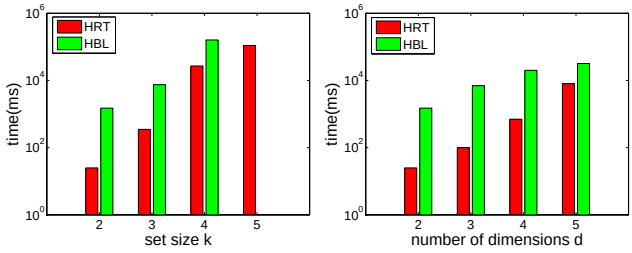


Figure 6: The impacts of RT and BL on different n.

following experiments, because the time cost of RT and BT are prohibitively high, we only compare HRT and HBL.



(a) different k (b) different d

Figure 7: The impacts of different k, d on NBA dataset.

Study on NBA dataset. Figure 7 shows the results on different k and d. The time cost grows exponentially with k and HRT significantly outperforms HBL. We did not report the result of HBL algorithm for some figures due to the high cost when k is big. The time cost also grows exponentially with d, HRT significantly outperforms HBL when d is small. However, when d is big, the difference is small due to the effect of "curse of dimensionality".

Study on Synthetic datasets. The results for the synthetic datasets are shown in Figure 8, 9, and 10. Generally speaking, the time cost for ANTI dataset is higher than INDE, and the time cost for INDE is higher than CORR. This is easy to understand because more instances are pruned in CORR dataset by our heuristic pruning strategies. On the other hand, it is harder to prune instances in ANTI dataset. From the global perspective, our algorithms are shown to be efficient and scalable, most of our experiments can be finished in 10^6 ms.

The impacts of different n are shown in (a) of Figure 8, 9, and 10 (m=10, k=5, d=5). Although the time complexity $C_n^k m^k knm < C_n^k m^k knlognm$ when m is small, HRT is much faster than HBL. There are two reasons: 1) the number of objects is much smaller than n after using heuristic pruning strategies, 2) the factor $nlognm$ of HRT is the worst-case while the factor nm is required. Comparing HRT to RT in Figure 6, HRT is almost 10^4 times faster than RT. Similar improvement can be observed for HBL and BL.

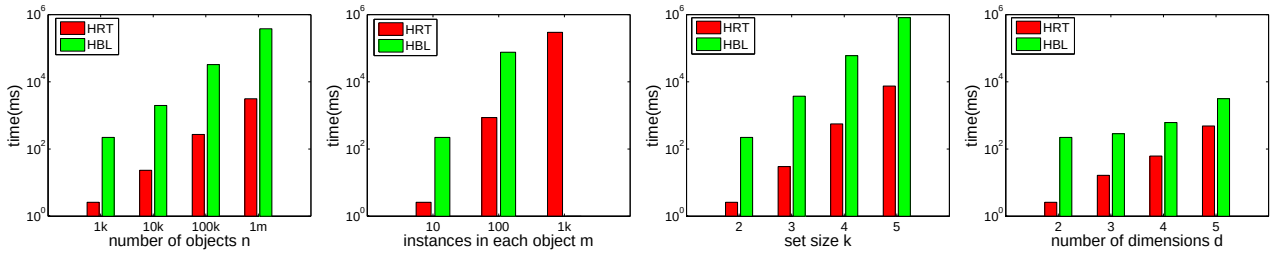
The impacts of different m are shown in (b) of Figure 8, 9, and 10 (n=1k, k=5, d=5). When $m = 10$ and $m = 100$, the time cost is fairly small because we can prune most of the instances in the heuristic pruning phase. However, when $m = 1k$, HRT can finish in 10^6 ms but HBL requires too much time which was not reported. The reason is that when m is big, although we can prune some instances, those objects with highest probability to be a skyline cannot be pruned and each object has m instances. For example, even when $n = 10, m = 1k$, $C_n^k m^k N = 4.5 \times 10^{11}$ while $C_n^k m^k nlogN = 6 \times 10^9$ and as we discussed before, $C_n^k m^k nlogN$ is the worst-case.

The impacts of different k are shown in (c) of Figure 8, 9, and 10 (n=1k, m=10, d=5). The set size k has significant effect on both HBL and HRT because k is the exponential term. Again, HRT outperforms HBL significantly.

The impacts of different d are shown in (d) of Figure 8, 9, and 10 (n=1k, m=10, k=5). They show that HRT is much faster than HBL when d is small. But the time cost of HRT grows exponentially because d is the exponential term. The time cost of HBL should increase linearly. However, in the experiment, Figure 10(d) shows that the time cost of HBL grows exponentially, the reason is when d increases, it is hard to prune instances by heuristic pruning strategies on ANTI datasets. Because most of the instances are pruned on CORR datasets, Figure 9(d) shows the time cost of HBL almost does not change for different d.

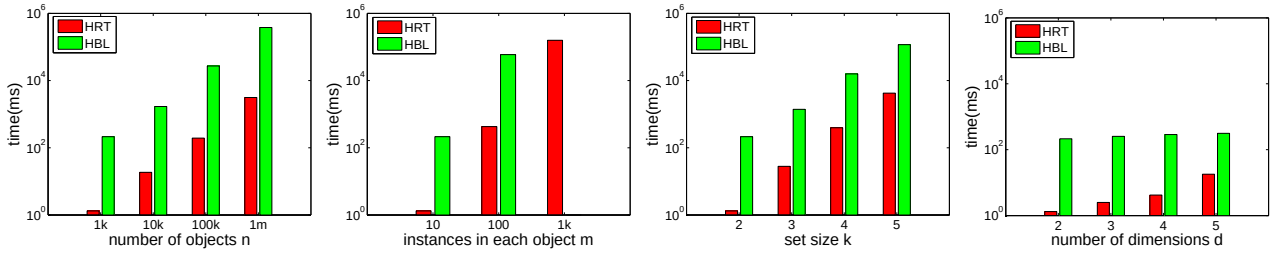
7. CONCLUSIONS

In this paper, for the first time, we studied probabilistic k-skyline sets problem on uncertain data. We show two significant heuristic pruning strategies which are efficient and simple. We adapted the classic layered range tree index structure to enhance the baseline algorithm from worst-case time bound aspect. The idea should be of independent interest for other problems. The experimental results on the



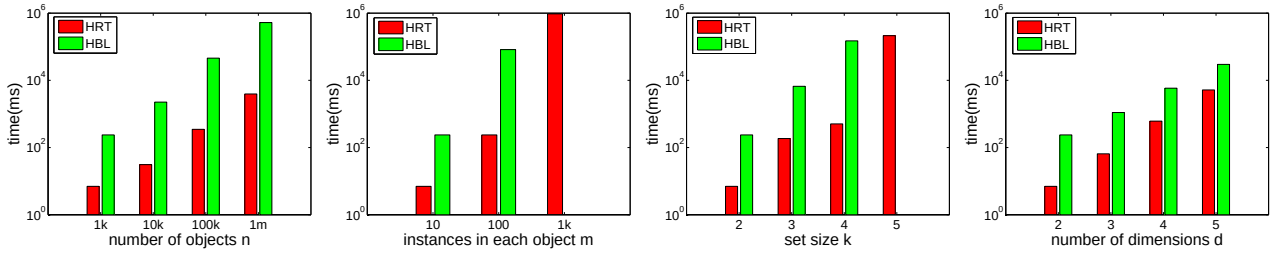
(a) different n (b) different m (c) different k (d) different d

Figure 8: The impacts of different n, m, k, d on INDE dataset.



(a) different n (b) different m (c) different k (d) different d

Figure 9: The impacts of different n, m, k, d on CORR dataset.



(a) different n (b) different m (c) different k (d) different d

Figure 10: The impacts of different n, m, k, d on ANTI dataset.

real NBA dataset and the synthetic datasets show that Pk-SkylineSets is interesting and useful, and our algorithms are efficient and scalable.

Acknowledgement

This research is partially supported by the Air Force Office of Scientific Research (AFOSR) DDDAS Program under award number FA9550-12-1-0240, the National Natural Science Foundation of China (Grant No. 11271351) and Guangdong Science and Technology Program Fund 2013B091300019.

8. REFERENCES

[1] P. Afshani, P. K. Agarwal, L. Arge, K. G. Larsen, and J. M. Phillips. (approximate) uncertain skylines. In *ICDT*, pages 186–196, 2011.

[2] R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. In *VLDB*, 1994.

[3] M. J. Atallah and Y. Qi. Computing all skyline probabilities for uncertain data. In *PODS*, pages 279–287, 2009.

[4] M. J. Atallah, Y. Qi, and H. Yuan. Asymptotically efficient algorithms for skyline probabilities of uncertain data. *ACM Trans. Database Syst.*, 36(2):12, 2011.

[5] I. Bartolini, P. Ciaccia, and M. Patella. The skyline of a probabilistic relation. *IEEE Trans. Knowl. Data Eng.*, 25(7):1656–1669, 2013.

[6] I. Bartolini, P. Ciaccia, and M. Patella. Domination in the probabilistic world: Computing skylines for arbitrary correlations and ranking semantics. *ACM Trans. Database Syst.*, 39(2):14, 2014.

[7] J. L. Bentley. Multidimensional divide-and-conquer. *Commun. ACM*, 23(4):214–229, 1980.

[8] J. L. Bentley, H. T. Kung, M. Schkolnick, and C. D. Thompson. On the average number of maxima in a set of vectors and applications. *J. ACM*, 25(4):536–543, 1978.

[9] S. Börzsönyi, D. Kossmann, and K. Stocker. The skyline operator. In *ICDE*, pages 421–430, 2001.

[10] B. Chazelle and L. J. Guibas. Fractional cascading: I. A data structuring technique. *Algorithmica*, 1(2):133–162, 1986.

[11] B. Chazelle and L. J. Guibas. Fractional cascading: II. applications. *Algorithmica*, 1(2):163–191, 1986.

[12] M. De Berg, M. Van Kreveld, M. Overmars, and O. C. Schwarzkopf. *Computational geometry*. Springer, 2000.

[13] V. Fisikopoulos. An implementation of range trees with fractional cascading in c++. *CoRR*, abs/1103.4521, 2011.

[14] A. Guttman. R-trees: A dynamic index structure for spatial searching. In *SIGMOD*, pages 47–57, 1984.

[15] M. E. Khalefa, M. F. Mokbel, and J. J. Levandoski. Skyline query processing for uncertain data. In *CIKM*, pages 1293–1296, 2010.

[16] D. G. Kirkpatrick and R. Seidel. Output-size sensitive algorithms for finding maximal vectors. In *Symposium on Computational Geometry*, pages 89–96, 1985.

[17] H. T. Kung, F. Luccio, and F. P. Preparata. On finding the maxima of a set of vectors. *J. ACM*, 22(4):469–476, 1975.

[18] X. Lian and L. Chen. Monochromatic and bichromatic reverse skyline search over uncertain databases. In *SIGMOD Conference*, 2008.

[19] X. Lin, Y. Zhang, W. Zhang, and M. A. Cheema. Stochastic skyline operator. In *ICDE*, pages 721–732, 2011.

[20] J. Liu, L. Xiong, J. Pei, J. Luo, and H. Zhang. Finding pareto optimal groups: group-based skyline. *PVLDB*, 2015.

[21] J. Liu, L. Xiong, and X. Xu. Faster output-sensitive skyline computation algorithm. *Inf. Process. Lett.*, 114(12):710–713, 2014.

[22] X. Liu, D.-N. Yang, M. Ye, and W.-C. Lee. U-skyline: A new skyline query for uncertain databases. *IEEE Trans. Knowl. Data Eng.*, 25(4):945–960, 2013.

[23] J. Pei, B. Jiang, X. Lin, and Y. Yuan. Probabilistic skylines on uncertain data. In *VLDB*, pages 15–26, 2007.

[24] J. Pei, W. Jin, M. Ester, and Y. Tao. Catching the best views of skyline: A semantic approach based on decisive subspaces. In *VLDB*, 2005.

[25] D. Sacharidis, A. Arvanitis, and T. K. Sellis. Probabilistic contextual skylines. In *ICDE*, pages 273–284, 2010.

[26] Q. Zhang, P. Ye, X. Lin, and Y. Zhang. Skyline probability over uncertain preferences. In *EDBT*, pages 395–405, 2013.

[27] W. Zhang, X. Lin, Y. Zhang, M. A. Cheema, and Q. Zhang. Stochastic skylines. *ACM Trans. Database Syst.*, 37(2):14, 2012.

[28] W. Zhang, X. Lin, Y. Zhang, W. Wang, and J. X. Yu. Probabilistic skyline operator over sliding windows. In *ICDE 2009*, pages 1060–1071, 2009.

[29] Y. Zhang, W. Zhang, X. Lin, B. Jiang, and J. Pei. Ranking uncertain sky: The probabilistic top-k skyline operator. *Inf. Syst.*, 36(5):898–915, 2011.