(Approximate) Uncertain Skylines*

Peyman Afshani
Faculty of Computer Science
Dalhousie University

Pankaj K. Agarwal
Department of Computer Science
Duke University

Lars Arge
MADALGO & Department of Computer Science
Aarhus University

Kasper Green Larsen
MADALGO & Department of Computer Science
Aarhus Univesity

Jeff M. Phillips
School of Computer
University of Utah

ABSTRACT

Given a set of points with uncertain locations, we consider the problem of computing the probability of each point lying on the skyline, that is, the probability that it is not dominated by any other input point. If each point's uncertainty is described as a probability distribution over a discrete set of locations, we improve the best known exact solution. We also suggest why we believe our solution might be optimal. Next, we describe simple, near-linear time approximation algorithms for computing the probability of each point lying on the skyline. In addition, some of our methods can be adapted to construct data structures that can efficiently determine the probability of a query point lying on the skyline.

Categories and Subject Descriptors

H.2 [**Database Management**]: Database Applications; E.2 [**Data**]: Data Storage RepresentationsComposite Structures

General Terms

Databases, Theory

Keywords

Skylines, Uncertainty, Approximation

*P.K.A. is supported by NSF under grants CNS-05-40347, CCF-06 -35000, IIS-07-13498, and CCF-09-40671, by ARO grants W911NF-07-1-0376 and W911NF-08-1-0452, by an NIH grant 1P50-GM-08183-01, and by a grant from the U.S.–Israel Binational Science Foundation. J.M.P. is supported by subaward CIF-32 from NSF grant 0937060 to CRA and subaward CIF-A-32 from NSF grant 1019343 to CRA. L.A. and K.G.L. are supported by MADALGO - Center for Massive Data Algorithmics - a Center of the Danish National Research Foundation. K.G.L is also supported in part by a Google Europe Fellowship in Search and Information Retrieval. P.A. is supported by Natural Sciences and Engineering Research Council of Canada through a post-doctoral fellowship program.

ICDT 2011, March 21–23, 2011, Uppsala, Sweden.

1. INTRODUCTION

In many applications, data uncertainty is an inherent consequence of data collection methods; for instance, data coming from a sensor network might contain duplicate readings of some data point. Alternatively, the output of noisy robotic sensors may be interpreted under several models, effectively replicating each data point several times. Or several simulation runs obtain different performance characteristics. In other situations, several similar data points are clustered to represent samples from one event, for example, all presidential candidates' performance over the past several elections. As a result of all of these situations, a recent focus in data management is on how to handle these uncertainties. This has generated much recent research in databases [7, 11, 29, 9, 10, 26, 8, 19, 2] and other areas [22, 21] on various types of systems, data structures, and optimization problems for such uncertain data.

Given a set P of points in $\mathbb{R}^d$, a point $p \in P$ is *on the skyline* of P if for every other point $q \in P$, at least one coordinate of p is larger than that of q. Computing the skyline of a set of points is useful in many applications, such as multi-criteria decision making, and has been extensively studied.

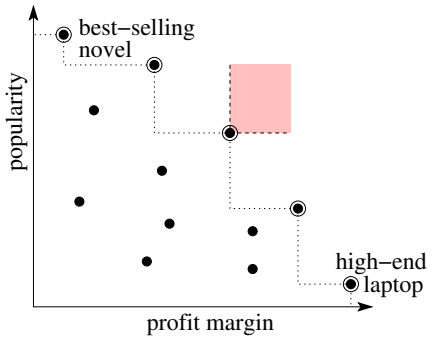


Figure 1. Example of skyline of products by attributes "profit-margin" and "popularity." Both a best-selling book and a high-end laptop may be on the skyline.

As motivation, consider an online store (like Amazon) that has a large number of different products, and it wishes to highlight the most important ones on its front page to boost its sales. Each product may have more than one attribute, for example, profit margin and potential popularity. A high-end laptop may have the largest profit margin but might be projected to sell little, while a sequel to a novel with a small profit margin might be expected to be a best-seller (see Figure 1). Thus, it is clear that when multiple attributes are present, it is highly non-trivial to have direct comparisons of the

products. Similar problems arise in many applications, since meaningfully combining all the attributes of a data point in a single value is difficult. The notion of a skyline enables us to circumvent this difficulty [5, 18]. For instance, in the previous example, if there is another laptop that has a bigger profit margin and is expected to be more popular (i.e., when the high-end laptop is not on the skyline), then the high-end laptop can be removed from the front page, if no such product exists, then the decision is not so easy.

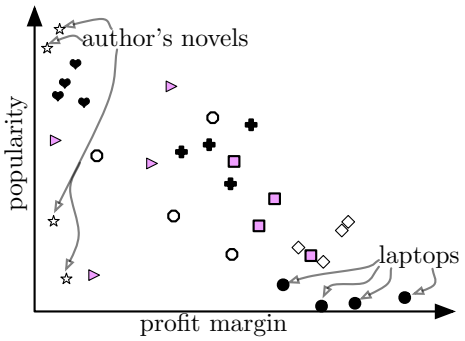


Figure 2. An example of uncertain point set. Uncertain points {♥, ✣, ◊} appear on 0.8-skyline.

In many applications, it is hard to specify the precise value of each attribute of an input point. For example, in the case of the online store, the exact measurement of potential popularity is almost impossible; the new laptop might do surprisingly well or the sequel to a novel might fail to reach the expectation of the fans. In our model, such possibilities are captured by having multiple values for potential popularity with a probability assigned to each value, see Figure 2. In general, in this richer model, there are k different values of an attribute, for a fixed k, with a probability assigned to each value.

Problem statement. Let $p = (p_1, p_2, \ldots, p_d)$ be a point in $\mathbb{R}^d$. For two points p and q in $\mathbb{R}^d$, p *dominates* q, denoted by $p \succ q$, if $p_j \geq q_j$ for all $1 \leq j \leq d$ and at least one of the inequalities is strict; a point does not dominate itself. For a set P of points in $\mathbb{R}^d$, a point $p \in P$ is *on the skyline* of P if no other point in P dominates p.

Defining the skyline of an uncertain data set is more difficult. Let $\mathcal{P} = \{P_1, \ldots, P_n\}$ be a set of n *uncertain* points in $\mathbb{R}^d$. We assume that each uncertain point P_i is described by a discrete probability distribution, defined over k discrete points (for an input parameter k). Namely, $P_i = \{p_{i,1}, p_{i,2}, \ldots, p_{i,k}\} \subset \mathbb{R}^d$, and the probability of P being at location $p_{i,j}$ is $0 < w_{i,j} \leq 1$; $\sum_{j=1}^{k} w_{i,j} = 1$. Note that we assume each $p_{i,j}$ to have nonzero probability of being the location of P_i. If $w_{i,j} = 0$ for some point $p_{i,j}$, we can remove that point. We also assume that the distributions of P_i's are independent. We remark that the assumption $|P_i| = k$ is for ease of exposition, and is not specifically required for any analysis. Set $S = \bigcup_{i=1}^{n} P_i$ and $|S| = m = nk$; the latter is called *the total size of the input*. We assume that no two points share x- or y-coordinates. Since all the algorithms described in this paper have running time $\Omega(m \log m)$, they can be adapted to relax this assumption in a straightforward manner.

Let p be a point in $\mathbb{R}^d$ (not necessarily in S). The probability that a point $P_i \in \mathcal{P}$ dominates p, denoted by $\sigma_i(p)$, is

$$\sigma_i(p) = \sum_{p_{i,j} \in P_i | p_{i,j} \succ p} w_{i,j},$$

and the probability that p is on the skyline of $\mathcal{P}$, also called the *skyline probability* of p, is

$$\mathsf{PS}_{\mathcal{P}}(p) = \prod_{i=1}^{n} (1 - \sigma_i(p)).$$

The probability of a point $P_i \in \mathcal{P}$ being on the skyline of $\mathcal{P}$, denoted by $\mathsf{PS}_{\mathcal{P}}(P_i)$, is

$$\mathsf{PS}_{\mathcal{P}}(P_i) = \sum_{j=1}^{k} w_{i,j} \mathsf{PS}_{\mathcal{P}_{\neq i}}(p_{i,j}),$$

where $\mathcal{P}_{\neq i} = \mathcal{P} \setminus \{P_i\}$ is all uncertain points except P_i. For a parameter $0 \leq \varepsilon < 1$, we call a value ρ_i an *ε-approximate skyline probability* of $p \in \mathbb{R}^2$ if $|\mathsf{PS}_{\mathcal{P}}(p) - \hat{\rho}_i| \leq \varepsilon$ and denote it by ε-$\mathsf{PS}_{\mathcal{P}}(p)$. Similarly we define ε-$\mathsf{PS}_{\mathcal{P}}(P_i)$ for a point $P_i \in \mathcal{P}$.

For $0 < \rho \leq 1$, a *ρ-skyline* of $\mathcal{P}$ [26] consists of all uncertain data points $P_i \in \mathcal{P}$ such that $\mathsf{PS}_{\mathcal{P}}(P_i) \geq \rho$. A subset $\mathcal{Q} \subseteq \mathcal{P}$ is called an *ε-approximate ρ-skyline* of $\mathcal{P}$ if for all $P_i \in \mathcal{Q}$, $\mathsf{PS}_{\mathcal{P}}(P_i) \geq \rho - \varepsilon$ and for all $P_j \in \mathcal{P} \setminus \mathcal{Q}$, $\mathsf{PS}_{\mathcal{P}}(P_j) \leq \rho + \varepsilon$. In this paper, we study the problems of computing skyline probabilities, computing exact and approximate ρ-skylines, and preprocessing $\mathcal{P}$ into a data structure for quickly returning an approximate skyline probability of a query point. We remark that aiming for simple, efficient approximation algorithms for computing ρ-skylines approximately that can guarantee error at a level no larger than what already exists in the input is quite natural.

Previous results. Kung *et.al.* [18] (see also [27, 4.1.3]) presented an algorithm for computing the skyline of a set of n (certain) points in $\mathbb{R}^d$ whose running time was $O(n \cdot (\log n + \log^{d-2} n))$. Koltun and Papadimitriou [16] considered the problem of computing an approximate skyline of a point set P, which they defined as follows: A subset $Q \subseteq P$ is called an *ε-approximate skyline* of S if the set $Q_\varepsilon = \{(1+\varepsilon)q \mid q \in Q\}$ (where $(1+\varepsilon)q$ scales each coordinate of q by $1+\varepsilon$) is the skyline of $P \cup Q_\varepsilon$. They showed that there exists an ε-approximate skyline of size $O(((1/\varepsilon)\log\Delta)^d)$, where Δ is the ratio of the largest and the smallest coordinate values of points in P. They also gave an $O(n \log n)$ algorithm to construct an ε-approximate skyline of the aforementioned size in $\mathbb{R}^2$, and they showed that it is NP-hard to attain an ε-approximate skyline of that size for $d > 3$. We refer to [5, 28, 17, 25, 13] and references therein for other work on skyline computation of certain point sets.

Constructing a skyline over a set $\mathcal{P}$ of uncertain points efficiently is more difficult. The straightforward method to compute $\sigma_i(p)$ and $\mathsf{PS}_{\mathcal{P}}(p)$ for a point $p \in \mathbb{R}^2$ takes $O(k)$ and $O(m)$ time, respectively. Thus, a ρ-skyline can be computed in $O(m^2)$ time. Atallah and Qi [3] improve the running time to $O(m^{5/3}\,\mathrm{polylog}(n))$ in $\mathbb{R}^2$. Their algorithm extends to higher dimensions and yields a subquadratic algorithm in any fixed dimension. Pei *et al.* [26] devise several heuristics for efficiently computing ρ-skylines. See [20] and [31] for other variants of this problem.

Our contributions. Let $\mathcal{P}$ be a set of n uncertain points in $\mathbb{R}^2$ as defined above, and let m be the total input size. In this paper we present a simpler and faster algorithm for computing a ρ-skyline of $\mathcal{P}$, whose running time is $O(m^{3/2})$. The algorithm extends to higher dimensions and yields an $O(m^{2-1/d})$ algorithm in $\mathbb{R}^d$. We also present a construction of an uncertain point set which suggests that the above bound might be optimal. We then show how the running time can be further improved for the natural case when $k \ll n$. More specifically, we describe an $O(mk \log m)$ algorithm for computing the skyline probabilities of all points in $\mathcal{P}$.[1]

[1]We remark that the running times mentioned above bound the number of arithmetic operations performed by the algorithms and not the bit complexity.

Next, we show how to compute, in $O(m(\log m + (1/\varepsilon)\log(1/\varepsilon)))$ time, ε-$\mathsf{PS}_{\mathcal{P}}(P_i)$ for all $1 \le i \le n$. In fact, we build, in $O(m(\log m + 1/\varepsilon))$ time, a data structure of size $O(n/\varepsilon^2)$ that can determine an ε-approximate skyline probability of a query point in $O(\log(n/\varepsilon))$ time. Notice that the size and the query time of the data structure are independent of k, the number of possible locations of each uncertain point, providing a concise representation when $k \gg 1/\varepsilon^2$.

Finally, we present a Monte Carlo algorithm for computing ε-approximate skyline probabilities. For a given parameter $0 < \delta < 1$, we compute in $O(m + (1/\varepsilon^2)n \log m \log(n/\delta))$ time a value ρ_i for each $P_i \in \mathcal{P}$ such that $\|\rho_i - \mathsf{PS}_{\mathcal{P}}(P_i)| \le \varepsilon$ for all $1 \le i \le n$ with probability at least $1 - \delta$. The algorithm is extremely simple and extends to higher dimensions in a straightforward manner, at the cost of increasing the running time to $O(m + (1/\varepsilon^2)n \cdot (\log^{d-2} n + \log k)\log(n/\delta))$. Furthermore, this approach can handle dependent and continuous distributions on the uncertain data points.

2. COMPUTING UNCERTAIN SKYLINES

Let $\mathcal{P}$ denote a set of n uncertain points in $\mathbb{R}^2$, and let m be the total input size. We present an algorithm for computing $\mathsf{PS}_{\mathcal{P}}(P_i)$ for all $1 \le i \le n$. Section 2.1 describes an algorithm with running time $O(m^{3/2})$. The algorithm extends to $\mathbb{R}^d$ at the cost of increasing the running time to $O(m^{2-1/d})$. Next, Section 2.2 presents an algorithm with running time $O(mk \log m)$, thereby improving upon the previous algorithm for $k = o(n/\log^2 n)$. For $k > n$, we can use a similar approach to compute $\mathsf{PS}_{\mathcal{P}}(P_i)$ for all points in time $O(mn \log m)$. Combining the two algorithms, we obtain an algorithm with running time $O(\min\{n,k\}m \log m)$. Finally, we make a conjecture about evaluating arithmetic operations in Section 2.3, which if true would prove an $\Omega(\min\{n,k\}m)$ lower bound on the problem of computing $\mathsf{PS}_{\mathcal{P}}(P_i)$ for all points in $\mathcal{P}$, matching our algorithms (up to the logarithmic factor).

2.1 An $O(m^{3/2})$ Time Algorithm

Recall that $S = \bigcup_i P_i$. At a high level the algorithm constructs a *kd*-tree T over all points in S [4]. Each leaf z of T is associated with one point $p_z \in S$. The algorithm computes $\mathsf{PS}_{\mathcal{P}}(p_z)$, as well as $\mathsf{PS}_{\mathcal{P}_{\neq i}}(p_z)$ assuming $p_z \in P_i$, for each leaf z. The special structure of T allows for an efficient computation of these values, reusing previously computed values whenever possible. We first give a brief review of the *kd*-tree structure along with a small modification we will adopt, and then we describe in detail how to compute $\mathsf{PS}_{\mathcal{P}_{\neq i}}(p_{i,j})$ for all $p_{i,j} \in S$ in a total of $O(m^{3/2})$ time. This trivially allows us to compute $\mathsf{PS}_{\mathcal{P}}(P_i)$ for all i in additional time $O(m)$.

Modified *kd*-tree. We construct a *kd*-tree T, which is a binary tree, on S as follows. Each node v of T is associated with a rectangle $R_v \subseteq \mathbb{R}^2$ and the subset $S_v = S \cap R_v$. For the root node u, $R_u = \mathbb{R}^2$ and $S_u = S$. If $|S_v| = 1$, then v is leaf of T, otherwise it is an interior node. The rectangles associated with the two children of v partition R_v into two rectangles. If v is at an odd (resp. even) level of T, then R_v is partitioned along the horizontal (resp. vertical) line so that each of the two sub-rectangles contains at most $\lceil |S_v|/2 \rceil$ points. We make a small change to the standard *kd*-tree definition at each leaf z: we set R_z to the degenerate rectangle consisting of the only point p_z of S_z.

For a node v, we say that a point p *dominates* v if p dominates all of R_v (not only the points in $S \cap R_v$), see Figure 3(a). Similarly, we say that p *intersects* v if it dominates some part of R_v, but not all it; see Figure 3(b). Finally, we say that p is *disjoint* from v if it does not dominate any part of R_v; see Figure 3(c). Note that with our modified definition of a *kd*-tree, a point cannot intersect a leaf node. With this terminology, we state the following key property

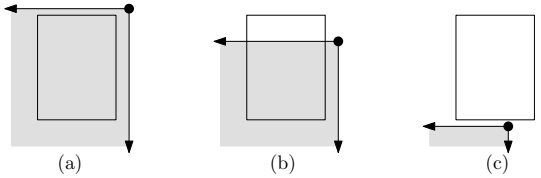


Figure 3. Classification of a point p: (a) p dominates v; (b) p intersects v; (c) p disjoint from v.

of T, which follows from well-known results [4, 14].

LEMMA 2.1. *Any point in $\mathbb{R}^2$ intersects $O(\sqrt{m})$ nodes of T.*

For a node v, let $D_v \subseteq S$ be the subset of points that dominate v, let $I_v \subseteq S$ be the subset of points that intersect v, and let $J_v \subseteq S$ be the set of points that are disjoint from v.

Algorithm. We first construct in $O(m \log m)$ time the *kd*-tree T on S as described above. We then perform a pre-order traversal of T. During the traversal of a node v, the algorithm maintains the following information at v:

(i) the set I_v.

(ii) For each i, the value

$$\sigma_i(v) = \sum_{p_{i,j} \in P_i \cap D_v} w_{i,j};$$

let $\Sigma(v) = \langle \sigma_i(v) \mid 1 \le i \le n \rangle$.

(iii) $\pi(v) = \prod_{i \in [1:n] \mid \sigma_i(v) \neq 1} (1 - \sigma_i(v))$.

(iv) $\chi(v) = |\{i \mid \sigma_i(v) = 1\}|$.

We maintain $\pi(v)$ and $\chi(v)$ instead of $\mathsf{PS}_{\mathcal{P}}(v) = \prod_{i=1}^{n}(1 - \sigma_i(v))$ because the latter quantity may be 0, in which case it will be difficult to update the quantity during the traversal; see below. For a leaf v with $S_v = \{p_{i,j}\}$,

$$\mathsf{PS}_{\mathcal{P}}(p_{i,j}) = \begin{cases} \pi(v) & \text{if } \chi(v) = 0, \\ 0 & \text{if } \chi(v) > 0. \end{cases} \tag{1}$$

Using the fact that $w_{i,j} > 0$, i.e., $\sigma_i(p_{i,j}) < 1$ for all $p_{i,j} \in S$, we obtain

$$\mathsf{PS}_{\mathcal{P}_{\neq i}}(p_{i,j}) = \frac{\mathsf{PS}_{\mathcal{P}}(p_{i,j})}{(1 - \sigma_i(v))}. \tag{2}$$

We now describe how we maintain (i)–(iv) during the traversal. For the root node u, $I_u = S$ since $R_u = \mathbb{R}^2$, which also implies that $\sigma_i(u) = 0$ for all i and $\chi(u) = 0, \pi(u) = 1$. Thus the initial setup for the traversal can easily be established. Suppose the traversal procedure reaches a node v. By induction, the above information is available at the parent $p(v)$ of v. We compute the sets I_v and $\Delta_v = D_v \setminus D_{p(v)}$ in $O(|I_{p(v)}|)$ time by scanning the set $I_{p(v)}$. Note that for any i,

$$\sigma_i(v) = \sigma_i(p(v)) + \sum_{p_{i,j} \in P_i \cap \Delta_v} w_{i,j}. \tag{3}$$

We thus construct $\Sigma(v)$ from $\Sigma(p(v))$ in $O(|\Delta_v|) = O(|I_{p(v)}|)$ time. Next, we compute $\pi(v)$ and $\chi(v)$ from $\pi(p(v))$, $\chi(p(v))$ as follows: We initially set $\chi(v) = \chi(p(v))$ and $\pi(v) = \pi(p(v))$. Let i be an index such that $\sigma_i(v) \neq \sigma_i(p(v))$. By (3), $\sigma_i(v) \ge \sigma_i(p(v))$. If $\sigma_i(v)$ becomes 1, we simply increment $\chi(v)$. Otherwise, we set

$$\pi(v) = \pi(v)\frac{1 - \sigma_i(v)}{1 - \sigma_i(p(v))}.$$

We repeat this step for all i's for which $\sigma_i(v) \neq \sigma_i(p(v))$.[2] The time spent in updating this information is $O(|I_{p(v)}|)$. Hence, the node v can be processed in $O(|I_{p(v)}|)$ time. If v is leaf containing a point $p_{i,j}$, then we report $\mathsf{PS}_{\mathcal{P}}(p)$ and $\mathsf{PS}_{\mathcal{P}_{\neq i}}(p)$ using (1) and (2), otherwise we recursively visit the two children of v. When the algorithm finishes traversing the subtree rooted at v (i.e., just before returning to $p(v)$), we undo the changes we made at v to restore $I_{p(v)}$, $\Sigma(p(v))$, $\chi(p(v))$, and $\pi(p(v))$. This step also takes $O(|I_{p(v)}|)$ time. Summing over all nodes of T and using Lemma 2.1, the total time spent by the algorithm is $\sum_{v \in T} O(|I_v|) = O(m^{3/2})$. Hence, we obtain the following.

Theorem 2.1. *Let $\mathcal{P} = \{P_1, \ldots, P_n\}$ be a set of n uncertain points in $\mathbb{R}^2$, and let m be the total input size. Then $\mathsf{PS}_{\mathcal{P}}(P_i)$ for all i can be computed in $O(m^{3/2})$ time.*

The algorithm presented above easily extends to $\mathbb{R}^d$ for $d > 2$ by constructing a modified *kd*-tree on points in $\mathbb{R}^d$. A point now intersects $O(m^{1-1/d})$ nodes of the tree [4, 14]. Following the same analysis as above, we now obtain the following.

Theorem 2.2. *Let $\mathcal{P} = \{P_1, \ldots, P_n\}$ be a set of n uncertain points in $\mathbb{R}^d$, and let m be the total input size. Then $\mathsf{PS}_{\mathcal{P}}(P_i)$, for all i, can be computed in $O(m^{2-1/d})$ time.*

Corollary 2.1. *Let $\mathcal{P} = \{P_1, \ldots, P_n\}$ be a set of n uncertain points in $\mathbb{R}^d$, and let m be the total input size. For any $0 \leq \rho \leq 1$, the ρ-skyline of $\mathcal{P}$ can be computed in $O(m^{2-1/d})$ time.*

2.2 A Refined Algorithm

We now describe an $O(mk \log m)$ time algorithm for computing $\mathsf{PS}_{\mathcal{P}}(P_i)$ for all points $P_i \in \mathcal{P}$. Roughly speaking, we reduce the problem to the so-called rectangle-stabbing problem, which in turn reduces to range searching under the group model. We first discuss the rectangle-stabbing problem and then describe how the problem of computing $\mathsf{PS}_{\mathcal{P}}(P_i)$ is reduced to it.

Rectangle-stabbing problem. Let $(\mathbb{X}, \oplus)$ be a commutative group, let $\mathcal{R} = \{R_1, \ldots, R_s\}$ be a set of rectangles in $\mathbb{R}^2$, and let $\omega : \mathcal{R} \to \mathbb{X}$ be a weight function. The *rectangle-stabbing* problem asks for preprocessing $\mathcal{R}$ into a data structure so that for a query point $q \in \mathbb{R}^2$, $\Omega(q, \mathcal{R}) = \bigoplus_{q \in R \in \mathcal{R}} \omega(R)$, the (group) sum of the weights of rectangles of $\mathcal{R}$ containing q, can be computed quickly. There is a simple reduction from rectangle stabbing to orthogonal range searching in the group model: replace each rectangle R with its four vertices. The weights of the lower left and upper right vertices of R is set to $\omega(R)$, and the weights of the other two vertices is set to $-\omega(R)$ (i.e., the inverse of $\omega(R)$). Let $\mathcal{R}^*$ denote the resulting set of $4s$ points in $\mathbb{R}^2$, and let $\omega : \mathcal{R}^* \to \mathbb{X}$ be their weights. For a point $q \in \mathbb{R}^2$, let Θ_q be the quadrant with q as its upper-right vertex, i.e., the set of points dominated by q. Then for any point $q \in \mathbb{R}^2$,

$$\Omega(q, \mathcal{R}) = \bigoplus_{\xi \in \mathcal{R}^* \cap \Theta_q} \omega(\xi).$$

See Figure 4.

Using a data structure by Willard [30], $\mathcal{R}^*$ can be preprocessed in $O(s \log s)$ time into a data structure of size $O(s \log s)$ so that a range query can be answered in $O(\log s)$ time. Hence, a rectangle-stabbing query under the group model can also be answered within the same time bound.

Reduction to rectangle stabbing. We construct $O(k^2)$ rectangles for each uncertain point P_i using the following procedure. For each point $p_{i,j} \in P_i$, we shoot two rays from $p_{i,j}$: one in $(-y)$-direction and another in $(-x)$-direction; we draw a quadrant $\Theta_{i,j}$ from $p_{i,j}$. See Figure 5(b). Let Ξ_i be planar decomposition (arrangement [1]) induced by these rays (quadrants). Then we construct a trapezoidal decomposition Ξ_i^{∇} of Ξ_i by shooting a ray upward from each point $p_{i,j}$ until it intersects an edge of Ξ_i; see Figure 5(c).

[2]The above step is the reason why we maintain $\pi(v), \chi(v)$ instead of $\mathsf{PS}_{\mathcal{P}}(v)$.

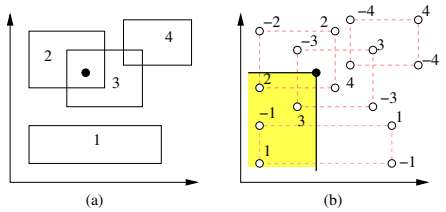


Figure 4. Reduction of rectangle stabbing to range searching: (a) an instance of rectangle stabbing; number inside each rectangle is its weight. (b) The corresponding instance of range searching; that shaded region is Θ_q.

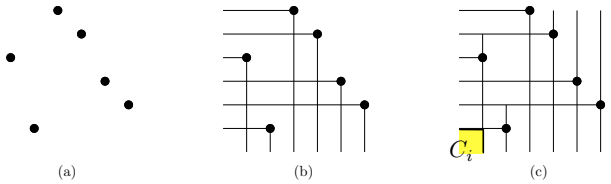


Figure 5. (a) A point set P_i. (b) The arrangement Ξ_i formed by shooting two rays from each point in P_i. (c) The trapezoidal decomposition Ξ_i^{∇} of Ξ_i. Shaded region is C_i.

Each cell of Ξ_i^{∇} is a rectangle, and there are $O(k^2)$ rectangles. For technical reasons we regard each rectangle R semi-open – its top and right edges belong to the rectangle but its left and bottom edges do not belong to R. This ensures that each point in $\mathbb{R}^2$ belongs to exactly one rectangle, and that $\sigma_i(q)$ is the same for all points in R except possible for its top right vertex if it is one of the points of P_i. Let $\sigma_i(R)$ denote this value, and we set $\omega(R) = 1 - \sigma_i(R)$. Let $C_i = \bigcap_{j=1}^{k} \Theta_{i,j}$ denote the quadrant among these rectangles that contains the point $(-\infty, -\infty)$, and let $\mathcal{R}_i$ be the remaining set of rectangles. Then $\omega(C_i) = 0$ (for any $q \in C_i$ and $p_{i,j} \in P_i$, $q \prec p_{i,j}$) and $\omega(R) > 0$ for all $R \in \mathcal{R}_i$ (there is a point $p_{i,j}$ such that $q \not\prec p_{i,j}$ for all $q \in R$).

Lemma 2.2. *Given a set P_i of k weighted points, we can compute $\mathcal{R}_i$ and C_i as well as their weights in time $O(k^2)$.*

Proof. Each ray intersects at most k other rays, thus there are at most k^2 intersections, and as many distinct rectangles. After sorting all points by x-coordinates (in $O(k \log k)$ time), we can construct the rectangles using a line sweep from large x-coordinates (right) to small x-coordinates (left). Given a sweep value h, we maintain all points with x-coordinate greater than h in sorted order according to their y-coordinates. It takes $O(\log k)$ time to identify the location of a new point $p_{i,j}$, and $O(k)$ time to create the rectangle above $p_{i,j}$ and the $O(k)$ rectangles below $p_{i,j}$. As we construct each rectangle R, we also compute $\sigma_i(R)$. The right most rectangle has $\sigma_i(R) = 0$. Then as we build $O(k)$ new rectangles by adding a point $p_{i,j}$, $\sigma_i(R)$ for the rectangle R above $p_{i,j}$ is the same as that of the rectangle just to the right of $p_{i,j}$. For a rectangle R below $p_{i,j}$, let R^+ be the rectangle immediately to its right. Then $\sigma_i(R) = \sigma_i(R^+) + w_{i,j}$. Thus all rectangles and their weights can be computed in $O(k \log k + k(\log k + k)) = O(k^2)$ time. □

Set $\mathcal{R} = \bigcup_{i=1}^{n} \mathcal{R}_i$ and $\mathcal{C} = \{C_i \mid 1 \le i \le n\}$. We choose the group $(\mathbb{R}^+, \cdot)$, i.e., positive real numbers with multiplication and 1 as the zero-element of the group. Let $\tilde{\mathcal{C}}$ be the union of quadrants in $\mathcal{C}$, i.e., any point lying in $\tilde{C}$ lies in at least one quadrant of $\mathcal{C}$. $\tilde{\mathcal{C}}$ is a staircase polygon with $O(n)$ vertices and can be computed in $O(n \log n)$ time. We preprocess $\mathcal{R}$ into a data structure for answering rectangle-stabbing queries as described above and preprocess $\tilde{\mathcal{C}}$ for point-location queries. For any point $q \in \mathbb{R}^2$, if $q \in \tilde{\mathcal{C}}$, then $\mathsf{PS}_{\mathcal{P}}(q) = 0$, as $\sigma_i(q) = 1$ for at least one i. Otherwise,

$$\begin{aligned}\Omega(q, \mathcal{R}) &= \prod_{i=1}^{n} \prod_{q \in R \in \mathcal{R}_i} \omega(R) \\ &= \prod_{i=1}^{n} (1 - \sigma_i(p)) = \mathsf{PS}_{\mathcal{P}}(q).\end{aligned}$$

Hence, we obtain the following:

Lemma 2.3. *Let $\mathcal{P} = \{P_1, \ldots, P_n\}$ be a set of n uncertain points in $\mathbb{R}^2$, and let m be the total input size. $\mathcal{P}$ can be preprocessed in $O(mk \log m)$ time into a data structure of size $O(mk \log m)$ so that for a query point $q \in \mathbb{R}^2$, $\mathsf{PS}_{\mathcal{P}}(q)$ can be computed in $O(\log m)$ time.*

To compute $\mathsf{PS}_{\mathcal{P}}(P_i)$ for all i, we first compute $\mathsf{PS}_{\mathcal{P} \neq i}(p_{i,j})$ for all points $p_{i,j} \in S$. Notice that each $p_{i,j}$ is a vertex of a rectangle in Ξ_i^{∇}, so we have already computed $\sigma_i(p_{i,j})$ for each $p_{i,j} \in S$. As mentioned earlier, $\sigma_i(p_{i,j}) < 1$, therefore

$$\mathsf{PS}_{\mathcal{P} \neq i}(p_{i,j}) = \frac{\mathsf{PS}_{\mathcal{P}}(p_{i,j})}{(1 - \sigma_i(p_{i,j}))}.$$

We now obtain the main result of this subsection.

Theorem 2.3. *Let $\mathcal{P} = \{P_1, \ldots, P_n\}$ be a set of n uncertain points in $\mathbb{R}^2$, and let m be the total input size. $\mathsf{PS}_{\mathcal{P}}(P_i)$ for all P_i can be computed in time $O(mk \log m)$.*

Remark. (i) The above algorithm does not seem to generalize well to higher dimensions. The main issue is that the complexity of Ξ_i increases to k^d in $\mathbb{R}^d$. Extending this technique thus only gives an algorithm with running time $\tilde{O}(nk^d)$ in $\mathbb{R}^d$.

A similar approach, whose details are omitted from here, can be used to compute skyline probabilities of all points of P in time $O(mn \log m)$. By combining this with the above theorem, we conclude the following:

Theorem 2.4. *Let $\mathcal{P} = \{P_1, \ldots, P_n\}$ be a set of n uncertain points in $\mathbb{R}^2$, and let m be the total input size. $\mathsf{PS}_{\mathcal{P}}(P_i)$ for all P_i can be computed in time $O(\min\{k, n\}m \log m)$.*

2.3 Optimality

In this section we discuss the optimality of our algorithms for computing the skyline probabilities of all points in $\mathbb{R}^2$. While we do not have a formal proof of optimality, we provide evidence that our algorithms might be optimal. We first make a conjecture about the hardness of evaluating certain arithmetic formulas, and then construct a specific set $\mathcal{P}^*$ of n uncertain points in $\mathbb{R}^2$ with k possible locations each. Under our conjecture, any algorithm that computes skyline probabilities of all points using a sequence of simple arithmetic operations (addition, subtraction, multiplication, division) on the weights of the input points and a set of predefined constants, where the sequence depends only on the positions of the points and not their weights, needs $\Omega(\min\{nk^2, n^2k\})$ time in the worst case, which is $\Omega(m^{3/2})$ in the worst case. We call such an algorithm a *truthful arithmetic algorithm*. All the known algorithms for computing skyline probabilities presented are truthful arithmetic algorithms. Apart from providing evidence of the hardness of computing all skyline probabilities, one can use the results in this section in two ways: (1) prove our conjecture and obtain an unconditional lower bound, or (2) use our specific input point set and conjecture to guide the search for faster algorithms.

Let $X_1, \ldots, X_m$ be variables taking positive real values. We say that a formula $C = 1 - X_{i_1} - X_{i_2} - \cdots - X_{i_j}$ is an *arithmetic clause* if all variables X_{i_j} take values in the range $(0, 1]$ and that $X_{i_1} + X_{i_2} + \cdots + X_{i_j} \le 1$. Furthermore, we say that a formula $F = C_1 \cdot C_2 \cdot \cdots \cdot C_k$ is in *arithmetic normal form* if each C_ℓ is an arithmetic clause. We make the following conjecture:

Conjecture 2.1. *Let $X_1, \ldots, X_m$ be variables taking positive real values, and let $\mathcal{F}$ be a set of formulas over $X_1, \ldots, X_m$ such that each formula is in arithmetic normal form. Let μ denote the total number of unique arithmetic clauses appearing in the formulas in $\mathcal{F}$. Any truthful arithmetic algorithm for evaluating all formulas in $\mathcal{F}$ needs to perform $\Omega(\mu)$ operations.*

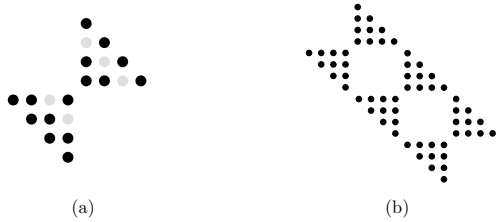
Intuitively this conjecture says that a truthful algorithm spend at least one arithmetic operation to evaluate each unique clause in order to evaluate all formulas of $\mathcal{F}$.

We are now ready to describe the specific input set $\mathcal{P}^*$. We regard the weight of each point $p_{i,j}$ as a variable taking positive real values, and show that any truthful arithmetic algorithm for computing all skyline probabilities on $\mathcal{P}^*$ is also an algorithm for evaluating all formulas in a set $\mathcal{F}^*$ of formulas in arithmetic normal form, where the number of unique arithmetic clauses in $\mathcal{F}^*$ is $\Omega(mk)$.

In the following we assume $n \ge k$, and afterwards briefly argue what happens in the opposite case. Our uncertain point set $\mathcal{P}^*$ consists of n/k smaller uncertain point sets $M_1, \ldots, M_{n/k}$ placed on a diagonal (if $k > n$, then just one point set). We first describe the smaller uncertain point sets M_i, and then show how these are translated onto a diagonal. Each uncertain point set M_i is a translation of the same uncertain point set M. M contains k uncertain points $P_1, \ldots, P_k$, each with k possible locations. The possible locations of uncertain point P_j are

$$\{(-j, -1), (-j+1, -2), \ldots, (-1, -j))\} \cup$$
$$\{(0, k-j-1), (1, k-j-2), \ldots, (k-j-1, 0)\}.$$

That is, the possible locations of each uncertain point constitutes two diagonals, see Figure 6(a). Now each small point set M_i in $\mathcal{P}^*$ is obtained by taking a copy of M and translating each point therein by $(ik, -ik)$. This places each M_i on a common diagonal, see Figure 6(b).



(a) (b)

Figure 6. (a) The point set M with $k = 5$, where possible locations of uncertain point P_2 are drawn in gray. (b) The point set $\mathcal{P}^*$ with $n = 15$.

Now consider point set M, and let $p \in M$ be a point on the jth diagonal in the positive quadrant, counting from longest to shortest diagonal (outer most to inner most). The expression $\mathsf{PS}_M(p)$ contains precisely j arithmetic clauses (one for each diagonal $1, \ldots, j$),

and none of these clauses appear in $\mathsf{PS}_M(q)$ for any other point $q \in M$ lying in the positive quadrant. Since $\mathsf{PS}_M(p)$ does not change by translating M into M_i, and since other point sets M_j where $i \neq j$ can only add new clauses to $\mathsf{PS}_M(p)$ and not alter previous clauses, we get that there are at least

$$n/k \cdot \sum_{j=1}^{k} (j \cdot (k-j)) = \Omega(nk^2)$$

unique arithmetic clauses in the set of formulas $\bigcup_{p \in \mathcal{P}^*} \mathsf{PS}_{\mathcal{P}^*}(p)$.

If $n < k$, we construct only one point set M_1. It is still constructed by taking a copy of M, but only using the points corresponding to uncertain points $P_1, \ldots, P_n$. In this setting, the number of unique arithmetic clauses become $\sum_{j=1}^{n}(j \cdot (k-j)) = \Omega(n^2 k)$. We thus conclude

THEOREM 2.5. *If Conjecture 2.1 is true, then any truthful arithmetic algorithm must spend $\Omega(\min\{nk^2, n^2k\})$ time in the worst case to compute all uncertain skyline probabilities for $\mathcal{P}^*$. This is maximized for $n = k$, yielding $\Omega(m^{3/2})$.*

3. APPROXIMATE UNCERTAIN SKYLINES

Let $\mathcal{P}$ be a set of uncertain points in $\mathbb{R}^2$ as above. In this section, we present a deterministic algorithm (Sections 3.1 and 3.2) and a Monte Carlo algorithm (Section 3.3) for computing ε-$\mathsf{PS}_{\mathcal{P}}(P_i)$ for all $P_i \in \mathcal{P}$. The deterministic algorithm uses the notion of ρ-chains, which is, roughly speaking, a level-set at height ρ of the function $\mathsf{PS}_{\mathcal{P}}$. Using ε-approximations of ρ-chains, we show that $\mathcal{P}$ can be preprocessed in time $O(m \log m + m/\varepsilon)$ into a data structure of size $O(n/\varepsilon^2)$ so that for any point $q \in \mathbb{R}^2$, ε-$\mathsf{PS}_{\mathcal{P}}(q)$ can be computed in $O(\log(n/\varepsilon))$ time; see Section 3.1. Building on this data structure, we show that ε-$\mathsf{PS}_{\mathcal{P}}(P_i)$, for all $1 \le i \le n$, can be computed in a total of $O(m(\log m + (1/\varepsilon)\log(1/\varepsilon)))$ time. The Monte Carlo algorithm, described in Section 3.3, basically computes skylines on fixed instantiations of $\mathcal{P}$ $O((1/\varepsilon^2)\log(n/\delta))$ times, and counts how many times each P_i's instantiation appears on the skyline.

3.1 ρ-Chain and Its Approximation

ρ-chain. For a paramater $0 < \rho \le 1$, let $\mathbb{X}_{>\rho} = \{x \in \mathbb{R}^2 \mid \mathsf{PS}_{\mathcal{P}}(x) > \rho\}$. The *$\rho$-chain* of $\mathcal{P}$, denoted by Γ_ρ, is the (lower) boundary of $\mathbb{X}_{>\rho}$, i.e., Γ_ρ separates points x with $\mathsf{PS}_{\mathcal{P}}(x) > \rho$ from those with $\mathsf{PS}_{\mathcal{P}}(x) \le \rho$. If we vary a point ξ, the value of $\mathsf{PS}_{\mathcal{P}}(\xi)$ changes only when its x- or y-coordinate becomes that of a point in S, and the value does not decrease if ξ moves in $(+x)$- or $(+y)$-direction. Therefore Γ_ρ is a staircase polygonal chain; see Figure 8. For $0 < \rho_1 < \rho_2 \le 1$, Γ_{ρ_1} and Γ_{ρ_2} may overlap (see Figure 7), but Γ_{ρ_1} never appears above Γ_{ρ_2}, i.e., no ray in $(+y)$-direction crosses Γ_{ρ_2} before crossing Γ_{ρ_1}.

Approximate ρ-chain. Let $0 < \varepsilon < 1$ be a parameter. A staircase polygonal chain Γ is called an *ε-approximate ρ-chain* if $\mathsf{PS}_{\mathcal{P}}(q) < \rho + \varepsilon$ for all points q lying below Γ and $\mathsf{PS}_{\mathcal{P}}(q) > \rho - \varepsilon$ for all points q lying above Γ. Note that Γ lies below $\Gamma_{\rho+\varepsilon}$. However, Γ may appear above $\Gamma_{\rho-\varepsilon}$ if there are points with skyline probability $\rho - \varepsilon$ (see Figure 7) but it always lies above $\Gamma_{\rho-\varepsilon-\delta}$ for any $\delta > 0$. A natural question is whether there is always an ε-approximate ρ-chain of small size. We prove this in affirmative by describing an algorithm that constructs an ε-approximate ρ-chain of size $O(n/\varepsilon)$.

Computing approximate ρ-chain. Let $0 < \rho, \varepsilon < 1$ be two parameters. We construct an ε-approximate ρ-chain Γ of $\mathcal{P}$ by performing two sweeps simultaneously: (i) *x-sweep*, a vertical line sweeps the plane from left to right, and (ii) *y-sweep*, a horizontal line sweeps the plane from top to bottom. We alternate between the two sweeps: intuitively, the x-sweep discovers the horizontal edges of Γ, and the y-sweep discovers the vertical edges of Γ. The algorithm traces a point ξ on Γ during the two sweeps and maintains the following invariant:

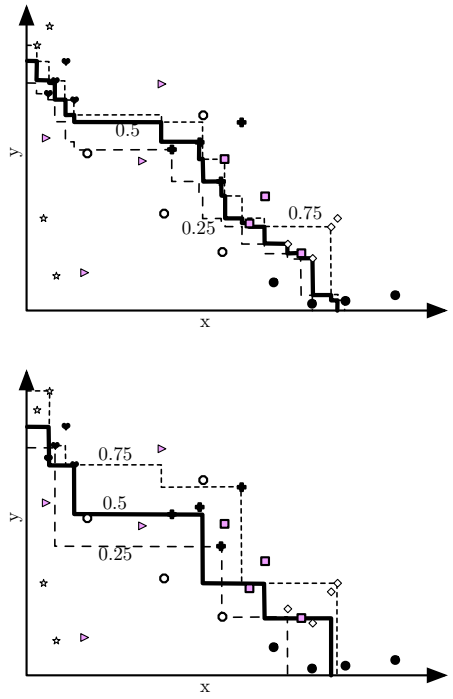


Figure 7. Example of ρ-chains for $\rho = 0.25, 0.5, 0.75$ (top), and ε-approximate ρ-chains for $\varepsilon = 0.25$ (bottom) on uncertain data. Each of $n = 8$ uncertain data points (denoted by different shapes) has $k = 4$ possible instantiations. The 0.5-chain and 0.25-approximate 0.5-chain have been made bold for easier comparison.

($\star$) $\mathsf{PS}_{\mathcal{P}}(q) > \rho - \varepsilon$ for every point $q \succ \xi$ and $\mathsf{PS}_{\mathcal{P}}(q) < \rho + \varepsilon$ for every point $q \prec \xi$.

Note that we do not make any claim for $\mathsf{PS}_{\mathcal{P}}(\xi)$. In order to maintain this invariant, we define a point $\xi^{\searrow}$ and monitor $\mathsf{PS}_{\mathcal{P}}(\xi^{\searrow})$ during the sweeps; see Lemma 3.1.

Initially we set $x(\xi) = -\infty$ and compute the initial y-coordinate of ξ (at $x = -\infty$) as follows. Let ℓ be a vertical line to the left of the leftmost point of S. Let $q_1, \ldots, q_m$ be the points on ℓ with the y-coordinates of points in S in increasing order; set q_0 be the point on ℓ at $-\infty$ and q_{m+1} the point on ℓ at $+\infty$. Then the following properties follow immediately from the definition of $\mathsf{PS}_{\mathcal{P}}$:

(i) $\mathsf{PS}_{\mathcal{P}}(q)$ remains the same for all points q in the (open) interval (q_i, q_{i+1}) for $0 \le i \le m$.

(ii) $\mathsf{PS}_{\mathcal{P}}(q) = 0$ for $q \in (q_0, q_1)$ and $\mathsf{PS}_{\mathcal{P}}(q) = 1$ for $q \in (q_m, q_{m+1})$.

(iii) If $q \in (q_i, q_{i+1})$ and $q' \in (q_{i+1}, q_{i+2})$, then $\mathsf{PS}_{\mathcal{P}}(q) \le \mathsf{PS}_{\mathcal{P}}(q')$.

If we know the values $\sigma_j(q)$ for all j, $|\{j \mid \sigma_j(q) = 1\}|$, $\prod_{\sigma_j(q)\neq 1}(1 - \sigma_j(q))$, and $\mathsf{PS}_{\mathcal{P}}(q)$ for any point in the interval (q_{i-1}, q_i), then these values for the points in the interval (q_i, q_{i+1}) can be updated in $O(1)$ time. Therefore we can compute the values of $\mathsf{PS}_{\mathcal{P}}$ in all intervals (q_i, q_{i+1}) in $O(m)$ time. Let q_i be the unique point such that $\mathsf{PS}_{\mathcal{P}}(q) > \rho$ for $q \in (q_i, q_{i+1})$ and $\mathsf{PS}_{\mathcal{P}}(q) \le \rho$ for $q \in (q_{i-1}, q_i)$. We set the initial y-coordinate of ξ to $y(q_i)$.

The algorithm maintains the following two auxiliary pieces of information during the sweeps:

I. For a point $q \in \mathbb{R}^2$, let $S^{\succ}(q) = \{p \in S \mid p \succ q\}$. The algorithm maintains $S^{\succ}(\xi^{\searrow})$.

II. The algorithm maintains

$$\begin{aligned}\Sigma(\xi^{\searrow}) &= \langle \sigma_1(\xi^{\searrow}), \ldots, \sigma_n(\xi^{\searrow})\rangle,\\ \chi(\xi^{\searrow}) &= |\{i \in [1:n] \mid \sigma_i(\xi^{\searrow}) = 1\}|,\\ \pi(\xi^{\searrow}) &= \prod_{i\in[1:n]|\sigma_i(\xi^{\searrow})\neq 1} (1-\sigma_i(\xi^{\searrow})).\end{aligned}$$

Each of the two sweeps pauses whenever the corresponding sweep line crosses a point of S, called an *event point*, and updates the coordinates of ξ as well as the above auxiliary information about $\xi^{\searrow}$. After processing an event, the algorithm either continues with the current sweep or switches to the other sweep. The point ξ traces a horizontal (resp. vertical) line with its y-coordinate (resp. x-coordinate) aligned with that of a point in S during the x-sweep (resp. y-sweep).

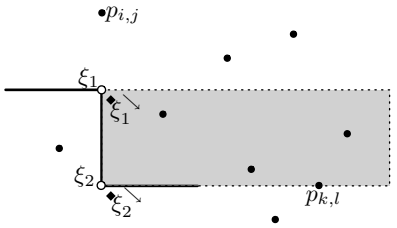


Figure 8. Small disks represent points in S. Thick solid line represents the path of ξ. ξ_1 and ξ_2 represent the positions of ξ at which x- and y-sweep pauses, respectively. The points in the gray area dominate both ξ_2 and $\xi_2^{\searrow}$, but not ξ_1 or $\xi_1^{\searrow}$.

The algorithms begins with the x-sweep starting at $x = -\infty$. Suppose the sweep line reaches a point $p_{i,j} \in S$. By our assumption, a sweep line never crosses two points of S at the same time. If $p_{i,j} \notin S^{\succ}(\xi^{\searrow})$, no processing is required at $p_{i,j}$, and the x-sweep continues. If $p_{i,j} \in S^{\succ}(\xi^{\searrow})$, the algorithm recomputes $\sigma_i(\xi^{\searrow})$ and $\mathsf{PS}_{\mathcal{P}}(\xi^{\searrow})$. By definition, $S^{\succ}(\xi^{\searrow}) = S^{\succ}(\xi^{\searrow}) \setminus \{p_{i,j}\}$, so we delete $p_{i,j}$ from $S^{\succ}(\xi^{\searrow})$, and update $\sigma_i(\xi^{\searrow})$, $\pi(\xi^{\searrow})$, and $\chi(\xi^{\searrow})$ in $O(1)$ time. If $\mathsf{PS}_{\mathcal{P}}(\xi^{\searrow})$ remains less than $\rho+\varepsilon$ (i.e., $\mathsf{PS}_{\mathcal{P}}(q) < \rho+\varepsilon$ for $q \prec \xi$), we continue with the x-sweep. If it becomes larger than $\rho+\varepsilon$, we interrupt the x-sweep, add $(x(p_{i,j}), y(\xi))$ as a vertex to Γ, and begin the y-sweep starting from the y-coordinate of ξ; ξ now traces the vertical line $x = x(p_{i,j})$. Suppose the y-sweep line reaches a point $p_{k,l}$. If $x(p_{k,l}) \le x(\xi)$, no action is taken since $p_{k,l}$ cannot be added to $S^{\succ}(\xi^{\searrow})$ and we continue with the y-sweep. If $x(p_{k,l}) > x(\xi)$, we add $p_{k,l}$ to $S^{\succ}(\xi^{\searrow})$ and update $\sigma_k(\xi^{\searrow}), \pi(\xi^{\searrow}), \chi(\xi^{\searrow})$. If $\mathsf{PS}_{\mathcal{P}}(\xi^{\searrow})$ remains larger than $\rho-\varepsilon$ (i.e., $\mathsf{PS}_{\mathcal{P}}(q) > \rho-\varepsilon$ for $q \succ \xi$), we continue with the y-sweep; otherwise, we interrupt the y-sweep, output $(x(\xi), y(p_{k,l}))$ as a vertex of Γ and resume the x-sweep at $p_{i,j}$; ξ now starts tracing the horizontal line $y = y(p_{k,l})$. The algorithm terminates when the y-sweep has processed the bottom-most point of S.

We spend $O(m \log m)$ to sort the points of S along x- and y-coordinates. After the sorting, the algorithm spends $O(1)$ time at each event point, and each of x- and y-sweep visits a point of S at most once. Hence, the total time spent by the two sweeps is $O(m)$. We first prove the correctness of the algorithm.

Lemma 3.1. *The algorithm always maintains the invariant ($\star$).*

Proof. It suffices to check whether the invariant is maintained in a sufficiently small neighborhood of ξ and whether it holds at all event points since the skyline probability changes only at x- or y-coordinates of a point in S. Let ξ^+ be the point traced by the algorithm as it moves forward from ξ by an infinitesimal amount. More precisely, let δ be a suffiently small parameter, as defined above. If the algorithm performs the x-sweep beyond ξ, then $\xi^+ = \xi + (\delta, 0)$; otherwise $\delta^+ = \delta + (0, -\delta)$.

Suppose, on the contrary, ($\star$) is violated for the first time at a point $p \in S$; let ξ be the point on Γ at that event. First consider the case when this event was encountered by the x-sweep. Since $\mathsf{PS}_{\mathcal{P}}(\xi)$ is not decreasing during the x-sweep, ($\star$) can be violated at p only because $\mathsf{PS}_{\mathcal{P}}(q) > \rho+\varepsilon$ for some point $q \prec \xi^+$. If Γ bends at ξ, then $p \succ \xi^+$, and we can infer that $\max_{q^+\in\xi^+} \mathsf{PS}_{\mathcal{P}}(q^+) = \max_{q^-\in\xi^-} \mathsf{PS}_{\mathcal{P}}(q^-)\rho+\varepsilon$ where ξ^- the point ξ immediately before the x-sweep reached p, and therefore the invariant is not violated at p. So assume that Γ does not bend at ξ, and the algorithms continues the x-sweep beyond p. Consider the point

$$\xi^+ + (0, -\delta) = \xi + (\delta, -\delta) = \xi^{\searrow}.$$

It can be checked that $\max_{q\prec\xi^+} \mathsf{PS}_{\mathcal{P}}(q) = \mathsf{PS}_{\mathcal{P}}(\xi^{\searrow})$. If ($\star$) does not hold at ξ^+, then $\mathsf{PS}_{\mathcal{P}}(\xi^{\searrow}) > \rho+\varepsilon$. However, the algorithm did not interrupt the x-sweep at p, implying that $\mathsf{PS}_{\mathcal{P}}(\xi^{\searrow}) \le \rho+\varepsilon$, a contradiction.

Next, suppose that the above event was encountered during the y-sweep. Since $\mathsf{PS}_{\mathcal{P}}(\xi)$ is not increasing during the y-sweep, ($\star$) is violated because $\mathsf{PS}_{\mathcal{P}}(q) < \rho-\varepsilon$ for some $q \succ \xi^+$. If Γ bends at this event, then again we can argue, as above, that ($\star$) is not violated at p. If Γ does not bend, i.e., the y-sweep continues beyond p, then consider the point

$$\xi^+ + (\delta, 0) = \xi + (\delta, -\delta) = \xi^{\searrow};$$

It can be checked that $\min_{q\succ\xi^+} \mathsf{PS}_{\mathcal{P}}(q) = \mathsf{PS}_{\mathcal{P}}(\xi^{\searrow})$. If ($\star$) is violated at ξ^+, then this quantity is less than $\rho-\varepsilon$. On the other hand, since the y-sweep was not interrupted at p, $\mathsf{PS}_{\mathcal{P}}(\xi^{\searrow}) \ge \rho-\varepsilon$, again a contradiction. Hence, we conclude that ($\star$) is always maintained by the algorithm. □

The above lemma immediately implies that $\mathsf{PS}_{\mathcal{P}}(q) \le \rho+\varepsilon$ for any point q lying below Γ and $\mathsf{PS}_{\mathcal{P}}(q) \ge \rho-\varepsilon$, thereby implying that Γ is an ε-approximate ρ-chain. The following lemma bounds the size of Γ.

Lemma 3.2. *The chain constructed by the algorithm has $O(n/\varepsilon)$ vertices.*

Proof. Consider an execution of the x-sweep. Let ξ_1 be the position of ξ when one of the x-sweep pauses, i.e., the vertex reported when the x-sweep paused. Note that $x(\xi_1) = x(p_1)$ for some $p_1 \in S$. Let ξ_2 be the position of ξ when the next y-sweep pauses, i.e., the other endpoint of the vertical edge adjacent to ξ_1. By construction, $\mathsf{PS}_{\mathcal{P}}(\xi_1^{\searrow}) > \rho+\varepsilon$ and $\mathsf{PS}_{\mathcal{P}}(\xi_2^{\searrow}) < \rho-\varepsilon$. Let $\mathsf{PS}_{\mathcal{P}}(\xi_1^{\searrow}) = \prod_{i=1}^n X_i$, in which $X_i = 1-\sigma_i(\xi_1^{\searrow})$ and assume $\mathsf{PS}_{\mathcal{P}}(\xi_2^{\searrow}) = \prod_{i=1}^n (X_i-\alpha_i)$, in which $X_i - \alpha_i = 1-\sigma_i(\xi_2^{\searrow})$, $\alpha_i \ge 0$, and $X_i - \alpha_i \le 1$. Notice that for a parameter $t \le 1$, we have $(X_i-\alpha_i)t \ge X_i t - \alpha_i$ and thus

$$\mathsf{PS}_{\mathcal{P}}(\xi_2^{\searrow}) \ge \left(\prod_{i=1}^n X_i\right) - \left(\sum_{i=1}^n \alpha_i\right) = \mathsf{PS}_{\mathcal{P}}(\xi_1^{\searrow}) - \left(\sum_{i=1}^n \alpha_i\right),$$

which implies $\sum_{i=1}^n \alpha_i \ge 2\varepsilon$. By definition, $\alpha_i = \sum w_{i,j}$ where the sum is taken over all points $p_{i,j} \in P_i \cap (S^{\succ}(\xi_2^{\searrow}) \setminus S^{\succ}(\xi_1^{\searrow}))$. Hence,

$$\sum_{i=1}^n \alpha_i = \sum_{p_{i,j}\in S^{\succ}(\xi_2^{\searrow})\setminus S^{\succ}(\xi_1^{\searrow})} w_{i,j}.$$

Since each point of S is deleted from the set $S^{\succ}(\xi^{\searrow})$ at most once and $\sum_{p_{i,j}\in S} w_{i,j} = n$, Γ has $O(n/\varepsilon)$ vertical edges. □

Computing ε-$\mathsf{PS}_{\mathcal{P}}$. The next step is to build several ε-approximate ρ-chains so that ε-$\mathsf{PS}_{\mathcal{P}}(q)$, for any point $q \in \mathbb{R}^2$, can be computed quickly. With this goal, we set $u = \lfloor 2/\varepsilon \rfloor$. For $1 \le r \le u$, let $\rho_r = r\varepsilon/2$. We construct an $(\varepsilon/6)$-approximate ρ_r chain Γ_r using the above algorithm. Γ_r lies between $(2r-1)\varepsilon/4$- and $(2r+1)\varepsilon/4$-chains, therefore Γ_r and Γ_{r+1} may overlap but Γ_r never appears above Γ_{r+1}. See Figure 9.

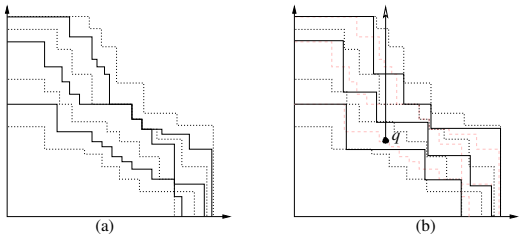


Figure 9. (a) ρ_r chains (in solid lines) and $(2i-1)\varepsilon/4$ lines for $1 \le r \le u$. (b) $\Gamma_1, \ldots, \Gamma_u$ and the "border" chains.

For a query point $q \in \mathbb{R}^2$, let $q^{\uparrow}$ be the (closed) ray emanating from q in $(+y)$-direction. We define $\varphi(q)$ as follows: if $q^{\uparrow}$ does not intersect any Γ_r, we set $\varphi(q) = 1$, otherwise $\varphi(q) = \rho_r$ if r is the smallest index such that $q^{\uparrow}$ intersects Γ_r. In other words, if q lies on one or more Γ_r's, then $\varphi(q)$ is set to the value corresponding to the smallest index chain that contains q, otherwise it is set to the value corresponding to the chain that lies immediately above q. It can be verified that $|\varphi(q) - \mathsf{PS}_{\mathcal{P}}(q)| \le \varepsilon$. Hence, the it suffices to return $\varphi(q)$ as ε-$\mathsf{PS}_{\mathcal{P}}(q)$ for the query point q.

Let Ξ be the planar subdivision induced by $\Gamma_1, \ldots, \Gamma_u$. We label each edge e of Ξ with the lowest index of the chain that contains e. Since $\Gamma_1, \ldots \Gamma_u$ partition the edges of Ξ into x-monotone chains, using the algorithm by Edelsbrunner *et al.* [15], Ξ can be preprocessed in $O(n/\varepsilon^2)$ time into a data structure of size $O(n/\varepsilon^2)$ so that $\varphi(q)$ can be computed in $O(\log(n/\varepsilon))$ time. We thus obtain the following.

THEOREM 3.1. *Let $\mathcal{P} = \{P_1, \ldots, P_n\}$ be a set of n uncertain points in $\mathbb{R}^2$, and let m be the total input size. $\mathcal{P}$ can be preprocessed in $O(m(\log m + 1/\varepsilon))$ time into a data structure of size $O(n/\varepsilon^2)$ so that for a query point $q \in \mathbb{R}^2$, ε-$\mathsf{PS}_{\mathcal{P}}(q)$ can be computed in $O(\log(n/\varepsilon))$ time.*

3.2 Approximate Skyline Probabilities

The above data structure is useful for a point $q \notin \mathcal{P}$, but to calculate $\mathsf{PS}_{\mathcal{P}}(P_i)$ for an uncertain point $P_i \in \mathcal{P}$ the main subroutine we need is to calculate $\mathsf{PS}_{\mathcal{P}_{\neq i}}(p_{i,j})$ for each $p_{i,j} \in P_i$. However, computing a data structure as above for each $\mathcal{P}_{\neq i}$ is not efficient. To overcome this challenge, we realize that

$$\mathsf{PS}_{\mathcal{P}_{\neq i}}(q) = \mathsf{PS}_{\mathcal{P}}(q)/(1-\sigma_i(q)).$$

However, another problem remains. If we invoke Theorem 3.1 to produce an estimate $\hat{\rho}(p_{i,j})$ that is an ε-$\mathsf{PS}_{\mathcal{P}}(p_{i,j})$, we cannot guarantee that $\hat{\rho}_{i,j} = \hat{\rho}(p_{i,j})/(1-\sigma_i(p_{i,j}))$ has the required approximation guarantee $|\hat{\rho}_{i,j} - \mathsf{PS}_{\mathcal{P}_{\neq i}}(p_{i,j})| \le \varepsilon$. Error reported in each of these chains, or between two such chains, is multiplied by $1/(1-\sigma_i(p_{i,j})) \ge 1$.

We overcome this problem in three ways. First, we adjust the computation of the chains so as to omit the smallest $(1-\sigma_i(q))$ in the calculation of $\pi(\xi^{\searrow})$. Let this term be $(1-\sigma^*)$; then we can multiply our modified estimate $\pi^*(p_{i,j})$ by $(1-\sigma^*)/(1-\sigma_i(p_{i,j}))$ which is at most 1 (note that $\sigma_i(p_{i,j}) < 1$ by definition). Second, we build $O((1/\varepsilon)\log(1/\varepsilon))$ chains instead of $O(1/\varepsilon)$ to deal with increased error for smaller ρ-values. Third, we build the computation of each ε-$\mathsf{PS}_{\mathcal{P}_{\neq i}}(p_{i,j})$ into the construction of all chains. Then, as we process the chain which is directly above each $p_{i,j}$ we have σ^* maintained.

Modified approximate ρ-chains.. We construct a modified chain similar to ε-approximate ρ-chains through alternating x-sweeps and y-sweep. We maintaining the following information corresponding to a point ξ on the chain:

I. The set of dominating points $S^{\succ}(\xi^{\searrow})$.

II. The auxiliary information:

$$\begin{aligned}
\Sigma(\xi^{\searrow}) &= \langle \sigma_1(\xi^{\searrow}), \ldots, \sigma_n(\xi^{\searrow}) \rangle, \\
\chi(\xi^{\searrow}) &= |\{i \in [1:n] \mid \sigma_i(\xi^{\searrow}) = 1\}|, \\
i^* &= \arg\max_i \sigma_i(\xi^{\searrow}), \\
\pi^*(\xi^{\searrow}) &= \prod_{i\in[1:n]|\sigma_i(\xi^{\searrow})\neq 1, i\neq i^*} (1-\sigma_i(\xi^{\searrow})).
\end{aligned}$$

Ties are broken arbitrarily for i^*. Then we can evaluate

$$\mathsf{PS}^*_{\mathcal{P}}(p) = \begin{cases} \pi^*(p) & \text{if } \chi(p) = 0 \\ 0 & \text{if } \chi(p) > 0 \end{cases}$$

and

$$\mathsf{PS}_{\mathcal{P}_{\neq i}}(p) = \begin{cases} \mathsf{PS}^*_{\mathcal{P}}(p)\dfrac{(1-\sigma_{i^*}(p))}{(1-\sigma_i(p))} & \text{if } \sigma_i(p) \neq 1, \\ \pi^*(p) & \text{if } \sigma_i(p) = 1, \chi(p) = 1, \\ 0 & \text{if } \sigma_i(p) = 1, \chi(p) > 1. \end{cases}$$

Now we can describe the modified sweeping algorithm to construct each chain Γ_r. It is a *modified α-approximate ρ_r-chain*, a staircase polygonal chain such that any q lying below Γ_r has $\mathsf{PS}^*_{\mathcal{P}}(q) < \rho_r + \alpha$ and any q lying above Γ_r has $\mathsf{PS}^*_{\mathcal{P}}(q) > \rho_r - \alpha$. The algorithm for each chain is identical to that described in Section 3.1 except for three modifications. (1) The auxiliary information maintained (above) is slightly different. (2) The decision to alternate between an x-sweep and y-sweep is determined by if $\mathsf{PS}^*_{\mathcal{P}}(\xi^{\searrow}) > \rho+\alpha$ or $\mathsf{PS}^*_{\mathcal{P}}(\xi^{\searrow}) < \rho - \alpha$, as opposed to the similar inequalities with $\mathsf{PS}_{\mathcal{P}}(\xi^{\searrow})$ in place of $\mathsf{PS}^*_{\mathcal{P}}(\xi^{\searrow})$. (3) To begin we say all $p_{i,j}$ are *unmarked*; when a chain passes above each $p_{i,j}$ for the first time, it will becomes *marked*. We handle unmarked points $p_{i,j}$ below chain Γ_r and calculate each ε-$\mathsf{PS}_{\mathcal{P}_{\neq i}}(p_{i,j})$ as described next.

We decompose the range $[0,1]$ into $w+1 = \log(2/\varepsilon)+1$ layers $L_1, L_2, \ldots, L_w, L_{w+1}$ where $\cup_{h=1}^{w+1} L_h = [0,1]$. Let $L_{w+1} = [0, \varepsilon/2]$ and each layer $L_h = (1/2^h, 1/2^{h+1}]$ for $1 \le h \le w$. Then ρ (the levels of our chains) takes on $u = 4/\varepsilon$ values[3] in each layer, except L_{w+1} which takes no values. Specifically, in layer h for $1 \le r \le u$ let $\rho_r = (1/2^h) + r\varepsilon(1/2^{h+2})$. Then we construct u modified $(\varepsilon/2^{h+4})$-approximate ρ_r-chains Γ_r in each layer h. We will construct chains starting at layer $h = w$ to layer $h = 1$, and within each layer, we construct each Γ_r starting from $r = 1$ to $r = u$. Thus a point $p_{i,j}$ becomes marked when processing chain Γ_r if it lies between chains Γ_{r-1} and Γ_r.

Note that we do not require to actually store the modified α-approximate ρ-chains, since all computation of $\mathsf{PS}_{\mathcal{P}_{\neq i}}(p_{i,j})$ will occur during the sweeping of the chains. However, if we did desire to

[3] If $1/\varepsilon$ is not a power of 2, we let γ be the smallest power of 2 greater than $1/\varepsilon$, and use $1/\gamma$ in place of ε everywhere. This ensures chains from different layers line up nicely.

construct the chains, we could do so, and each would have at most $O(n2^h/\varepsilon)$ vertices. The proof of Lemma 3.2 can be modified to only consider the product based on $\pi^*(\xi^{\searrow})$ instead of $\pi(\xi^{\searrow})$. More importantly, the construction of each chain still requires $O(m)$ time.

Constructing ε-$\mathsf{PS}_{\mathcal{P}_{\neq i}}(p_{i,j})$.. Finally, we discuss how to construct each ε-$\mathsf{PS}_{\mathcal{P}_{\neq i}}(p_{i,j})$, denoted $\varphi(p_{i,j})$. First some boundary cases. Any point $p_{i,j}$ marked on the first Γ_1 in layer w has

$$\mathsf{PS}_{\mathcal{P}_{\neq i}}(p_{i,j}) \leq \mathsf{PS}^*_{\mathcal{P}}(p_{i,j}) \leq \varepsilon/2 + \varepsilon^2/4,$$

so we can set $\varphi(p_{i,j}) = 0$. We handle all points $p_{i,j}$ not marked after the last Γ_u with one additional pass of a modified 0-approximate 1-skyline that goes through or above all points, and we analyze it along with other chains in layer 1. From here on, we assume any point $p_{i,j}$ in layer h is marked between two chains Γ_{r-1} and Γ_r; if a point is marked during Γ_1, the first chain of layer h, then the last chain of layer $h+1$ serves as Γ_0 for analysis purposes.

Each $p_{i,j}$ is handled when it is marked on an x-sweep such that $x(p_{i,j}) = x(\xi)$ and $y(p_{i,j}) \leq y(\xi)$. It will be useful to refer to this point ξ as ξ_r. We can draw a vertical ray from ξ_r through $p_{i,j}$ in the $(-y)$-direction, and it will intersect Γ_{r-1} at a point ξ_{r-1}. We observe that $x(\xi_{r-1}) = x(p_{i,j})$ and $y(\xi_{r-1}) < y(p_{i,j})$. Since in the $(+y)$-direction $\mathsf{PS}^*_{\mathcal{P}}(\cdot)$ is monotonically non-decreasing, then

$$\mathsf{PS}^*_{\mathcal{P}}(\xi_{r-1}) \leq \mathsf{PS}^*_{\mathcal{P}}(p_{i,j}) \leq \mathsf{PS}^*_{\mathcal{P}}(\xi_r^{\searrow}).$$

Hence, for r in layer h, we can approximate $\mathsf{PS}^*_{\mathcal{P}}(p_{i,j})$ using $\mathsf{PS}^*_{\mathcal{P}}(\xi_r^{\searrow})$, incurring at most $\varepsilon/2^{h+1}$ error because

$$(r-1)\frac{\varepsilon}{2^{h+2}} - \frac{\varepsilon}{2^{h+4}} \leq \mathsf{PS}^*_{\mathcal{P}}(\xi_{r-1}) \leq \mathsf{PS}^*_{\mathcal{P}}(\xi_r^{\searrow}) \leq r\frac{\varepsilon}{2^{h+2}} + \frac{\varepsilon}{2^{h+4}}.$$

We also have the property of a point $p_{i,j}$ that is marked during a chain Γ_r in layer h that $\mathsf{PS}^*_{\mathcal{P}}(p_{i,j}) \geq 1/2^{h+1}$.

Note that, since by definition $\sigma_i(p_{i,j}) < 1$ so $1/(1-\sigma_i(p_{i,j}))$ does not divide by 0. Also, define $i^*_r = \arg\max_l \sigma_l(\xi_r^{\searrow})$ so when $p_{i,j}$ is handled we have i^*_r maintained. Let $i^*_{i,j} = \arg\max_l \sigma_l(p_{i,j})$; we do not know $i^*_{i,j}$. But we show that using i^*_r is a good enough approximation. Specifically we can generate an ε-$\mathsf{PS}_{\mathcal{P}_{\neq i}}(p_{i,j})$, as

$$\varphi(p_{i,j}) = \frac{1-\sigma_{i^*_r}(p_{i,j})}{1-\sigma_i(p_{i,j})}\mathsf{PS}^*_{\mathcal{P}}(\xi_r^{\searrow}).$$

In proving this result we have two basic types of inequalities to use: (1) change in query value: $\sigma_l(\xi_r^{\searrow}) \leq \sigma_l(p_{i,j}) \leq \sigma_l(\xi_{r-1})$; and (2) compare to star-index: $\sigma_{i^*_r}(\xi_r^{\searrow}) \geq \sigma_l(\xi_r^{\searrow})$ and $\sigma_{i^*_{i,j}}(p_{i,j}) \geq \sigma_l(p_{i,j})$. We first prove a helpful lemma involving $\pi_{k,l}(q) = \prod_{i\neq k,l}(1-\sigma_i(q))$.

Lemma 3.3. *If $p_{i,j}$ is handled during chain Γ_r at ξ_r in layer h, then*

$$\sigma_{i^*_{i,j}}(p_{i,j}) - \sigma_{i^*_r}(p_{i,j}) \leq \sigma_{i^*_{i,j}}(p_{i,j}) - \sigma_{i^*_{i,j}}(\xi_r^{\searrow}) < \varepsilon/(2^{h+1}\pi_{i^*_{i,j},i^*_r}(\xi_r^{\searrow})) \leq \varepsilon.$$

Proof. Assume the middle inequality of the lemma is false so $\sigma_{i^*_{i,j}}(p_{i,j}) - \sigma_{i^*_{i,j}}(\xi_r^{\searrow}) \geq \varepsilon/2^{h+1}\pi_{i^*_{i,j},i^*_r}(\xi_r^{\searrow})$. We show this implies $\mathsf{PS}^*_{\mathcal{P}}(\xi_r^{\searrow}) - \mathsf{PS}^*_{\mathcal{P}}(\xi_{r-1}^{\searrow}) \geq \varepsilon/2^{h+1}$, a contradiction.

$$\begin{aligned}
&\mathsf{PS}^*_{\mathcal{P}}(\xi_r^{\searrow}) - \mathsf{PS}^*_{\mathcal{P}}(\xi_{r-1}) \\
&= \pi_{i^*_r,i^*_{i,j}}(\xi_r^{\searrow})(1-\sigma_{i^*_{i,j}}(\xi_r^{\searrow})) - \pi_{i^*_{r-1},i^*_{i,j}}(\xi_{r-1})(1-\sigma_{i^*_{i,j}}(\xi_{r-1})) \\
&\geq \pi_{i^*_r,i^*_{i,j}}(\xi_r^{\searrow})\left((1-\sigma_{i^*_{i,j}}(\xi_r^{\searrow})) - (1-\sigma_{i^*_{i,j}}(\xi_{r-1}))\right) \\
&= \pi_{i^*_r,i^*_{i,j}}(\xi_r^{\searrow})\left(\sigma_{i^*_{i,j}}(\xi_{r-1}) - \sigma_{i^*_{i,j}}(\xi_r^{\searrow})\right) \\
&\geq \pi_{i^*_r,i^*_{i,j}}(\xi_r^{\searrow})\left(\sigma_{i^*_{i,j}}(p_{i,j}) - \sigma_{i^*_{i,j}}(\xi_r^{\searrow})\right) \\
&\geq \pi_{i^*_r,i^*_{i,j}}(\xi_r^{\searrow})\varepsilon/(2^{h+1}\pi_{i^*_r,i^*_{i,j}}(\xi_r^{\searrow})) \\
&= \varepsilon/2^{h+1}.
\end{aligned}$$

The first inequality uses $\pi_{i^*_r,i^*_{i,j}}(\xi_r^{\searrow}) \geq \pi_{i^*_{r-1},i^*_{i,j}}(\xi_{r-1})$ since $\pi_{i^*_{i,j},i^*_k}(\xi_k^{\searrow})$ is monotonically non-decreasing as k increases. The second inequality follows from $\sigma_l(\xi_{r-1}) \geq \sigma_l(p_{i,j})$. The final inequality follows from our (false) assumption.

Finally we get the left-hand-side by $\sigma_{i^*_r}(p_{i,j}) \geq \sigma_{i^*_r}(\xi_r^{\searrow}) \geq \sigma_{i^*_{i,j}}(\xi_r^{\searrow})$, and we get the right-hand-side since in layer h we have $\pi_{i^*_{i,j},i^*_r}(\xi_r^{\searrow}) \geq \mathsf{PS}^*_{\mathcal{P}}(\xi_r^{\searrow}) \geq 1/2^{h+1}$. □

Lemma 3.4. *$\varphi(p_{i,j}) = \mathsf{PS}^*_{\mathcal{P}}(\xi_r^{\searrow}) \cdot (1-\sigma_{i^*_r}(p_{i,j}))/(1-\sigma_i(p_{i,j}))$ is an ε-$\mathsf{PS}_{\mathcal{P}_{\neq i}}(p_{i,j})$.*

Proof. Observe that $\sigma_{i^*_{i,j}}(\xi_r^{\searrow}) \leq \sigma_{i^*_r}(\xi_r^{\searrow}) \leq \sigma_{i^*_r}(p_{i,j}) \leq \sigma_{i^*_{i,j}}(p_{i,j})$. Now using our bound on $\mathsf{PS}^*_{\mathcal{P}}(\xi_r^{\searrow}) - \mathsf{PS}^*_{\mathcal{P}}(p_{i,j}) < \varepsilon/2^{h+1}$ and Lemma 3.3

$$\begin{aligned}
&\frac{1-\sigma_{i^*_r}(p_{i,j})}{1-\sigma_i(p_{i,j})}\mathsf{PS}^*_{\mathcal{P}}(\xi_r^{\searrow}) - \frac{1-\sigma_{i^*_{i,j}}(p_{i,j})}{1-\sigma_i(p_{i,j})}\mathsf{PS}^*_{\mathcal{P}}(p_{i,j}) \\
&< \left(\frac{1-\sigma_{i^*_r}(p_{i,j})}{1-\sigma_i(p_{i,j})} - \frac{1-\sigma_{i^*_{i,j}}(p_{i,j})}{1-\sigma_i(p_{i,j})}\right)\mathsf{PS}^*_{\mathcal{P}}(p_{i,j}) + \frac{\varepsilon}{2^{h+1}}\frac{1-\sigma_{i^*_r}(p_{i,j})}{1-\sigma_i(p_{i,j})} \\
&\leq \frac{\varepsilon/2^{h+1}}{\pi_{i^*_{i,j},i^*_r}(\xi_r^{\searrow})}\frac{\mathsf{PS}^*_{\mathcal{P}}(p_{i,j})}{1-\sigma_i(p_{i,j})} + \frac{\varepsilon}{2^{h+1}}\frac{1-\sigma_{i^*_{i,j}}(p_{i,j})+\varepsilon}{1-\sigma_i(p_{i,j})} \\
&\leq \frac{\varepsilon}{2^{h+1}}\frac{1-\sigma_{i^*_{i,j}}(p_{i,j})}{1-\sigma_i(p_{i,j})} + \frac{\varepsilon}{2^{h+1}} + \frac{\varepsilon}{2^{h+1}}\varepsilon \\
&\leq \frac{\varepsilon}{2^{h-1}} \leq \varepsilon. \quad \square
\end{aligned}$$

Once all ε-$\mathsf{PS}_{\mathcal{P}_{\neq i}}(p_{i,j})$ have been computed we can calculate $\hat{\rho}_i = \sum_{j=1}^k w_{i,j}\varphi(p_{i,j})$. Since $\sum_{j=1}^k w_{i,j} = 1$, their total error on the sum is at most ε, and $\hat{\rho}_i$ is an ε-$\mathsf{PS}_{\mathcal{P}}(P_i)$ as desired.

To bound the runtime, we need to construct $O((1/\varepsilon)\log(1/\varepsilon))$ approximate skylines, each in time $O(m)$. To handle each point $p_{i,j}$ we need to calculate $\sigma_l(p_{i,j})$ for two different values l. Each takes time $O(\log k)$ time after preprocessing each uncertain point P_i in $O(k\log k)$ time. Over all m points this requires $O(m\log k)$ time.

Theorem 3.2. *Consider a discrete uncertain data set $\mathcal{P} = \{P_1, \ldots, P_n\}$ where each $P_i = \{p_{i,1}, \ldots, p_{i,k}\}$ so $|P_i| = k$ and $\sum_{i=1}^n |P_i| = nk = m$. For all uncertain points P_i we can compute a value $\hat{\rho}_i$ such that $|\hat{\rho}_i - \mathsf{PS}_{\mathcal{P}}(P_i)| \leq \varepsilon$ in $O(m((1/\varepsilon)\log(1/\varepsilon) + \log m))$ total time.*

Remark. This algorithm does not directly generalize to $\mathbb{R}^d$. The results on approximate (certain) skylines [16] combined with our results in $\mathbb{R}^2$ indicate that there should exist ε-approximate ρ-chains of size independent of k but exponential in d, however, for $d > 3$, there may be no polynomial time algorithm. Proof of the existence of such ε-approximate ρ-chains and possible polynomial time algorithms in dimensions $d \geq 3$ remains an open question.

3.3 A Monte Carlo Algorithm

In this section, we present a Monte Carlo algorithm for computing ε-$\mathsf{PS}_{\mathcal{P}}(P_i)$ for all $1 \leq i \leq n$. We fix a parameter t to be chosen later. The algorithm runs in t steps. In each step, we instantiatae a specific location π_i of uncertain point P_i; the location $p_{i,j} \in P_i$ is chosen with probability $w_{i,j}$. Let $\Pi = \{\pi_i \mid 1 \leq i \leq n\} \subset \mathbb{R}^2$ be the resulting set of n points. We compute the skyline of Π. We also maintain a vector $\nu = \langle \nu_1, \ldots, \nu_n \rangle$, which is initially set to $\mathbf{0}$. If π_i appears on the skyline of Π, we increment the value of ν_i, so ν_i keeps track of the number of instantiations of $\mathcal{P}$ in which P_i appears on the skyline. After the completion of t steps, we return the value $\hat{\rho}_i = \nu_i/t$ as ε-$\mathsf{PS}_{\mathcal{P}}(P_i)$.

After preprocessing the weights $\{w_{i,1}, \ldots, w_{i,k}\}$ in $O(k)$ time into a minimum-height binary tree, an instantiation π_i of P_i can be done

in $O(\log k)$ time [23]. Thus instantiating n uncertain points takes $O(n \log k)$ time after $O(m)$ preprocessing. Given an instantiation Π of $\mathcal{P}$, finding the skyline of Π takes $O(n \log n)$ time [18]. Thus the overall algorithm takes

$$O(tn(\log n + \log k) + m) = O(m + tn \log m)$$

time. What remains is to determine the value of t that guarantees $|\mathsf{PS}_{\mathcal{P}}(P_i) - \hat{\rho}_i| \leq \varepsilon$ for all i with high probability.

First consider a fixed uncertain point P_i. For $1 \leq j \leq t$, let $\mathrm{X}_j \in \{0, 1\}$ be a random indicator variable, which is 1 if P_i appears on the skyline in iteration j and 0 otherwise. Clearly these variables are independent. Set $\mathrm{X} = \sum_{j=1}^{t} \mathrm{X}_j$. Furthermore, $\mathrm{X} = \nu_i = \hat{\rho}_i t$ and $\mathsf{E}[\mathrm{X}] = \mathsf{PS}_{\mathcal{P}}(P_i)t$. Using a simplified version of Chernoff-Hoeffding theorem (see e.g. [23]), we obtain

$$\begin{aligned}\Pr\left[|\mathsf{PS}_{\mathcal{P}}(P_i) - \hat{\rho}_i| > \varepsilon\right] &= \Pr[|\mathsf{PS}_{\mathcal{P}}(P_i)t - \nu_i| > \varepsilon t] \\ &\leq \Pr[|\mathsf{E}[\mathrm{X}] - \nu_i| > \varepsilon \mathsf{E}[\mathrm{X}]] \\ &< \exp(-\varepsilon^2 t).\end{aligned}$$

If we choose

$$t \geq \frac{1}{\varepsilon^2} \ln \frac{n}{\delta},$$

then $\Pr[|\mathsf{PS}_{\mathcal{P}}(P_i) - \hat{\rho}_i| > \varepsilon] < \delta/n$. Hence,

$$\Pr[\exists i \leq n \mid |\mathsf{PS}_{\mathcal{P}}(P_i) - \hat{\rho}_i| > \varepsilon] < \delta,$$

and, we obtain the following theorem.

THEOREM 3.3. *Let $\mathcal{P} = \{P_1, \ldots, P_n\}$ be a set of n uncertain points in $\mathbb{R}^2$, and let m be the total input size of m. Let $0 < \delta, \varepsilon < 1$ be two parameters. For each $P_i \in \mathcal{P}$ a value $\hat{\rho}_i$ can be computed in $O(m + (1/\varepsilon^2)n \log m \log(n/\delta))$ time such that $|\hat{\rho}_i - \mathsf{PS}_{\mathcal{P}}(P_i)| \leq \varepsilon$ for all $1 \leq i \leq n$ with probability at least $1 - \delta$.*

Remarks. (i) The algorithm presented above extends to higher dimensions: we simply use the algorithm for computing the skyline of a set of points in $\mathbb{R}^d$ [18], which takes $O(n \log^{d-2} n)$ time. The running thus becomes

$$O(m + (1/\varepsilon^2)n(\log^{d-2} n + \log k) \log(n/\delta)).$$

(ii) Since all instantiations Π of uncertain points in $\mathcal{P}$ are chosen together, the algorithms can be extended to operate on joint distributions over the uncertainty, given that we have a model to sample from that joint distribution.

4. DISCUSSION

This paper studies asymptotic results in computing skylines for uncertain data. We first present two new algorithms for exactly computing the probability that each uncertain point is on the skyline, and we show that under a specific realistic model, these are optimal up to a polylogarithmic factor. Then we present two new near-linear time algorithms for approximately computing the probability that each uncertain points is on the skyline. For both problems, one algorithm easily extends to higher dimensions, and the other provides a data structure which allows (approximate) determination of the probability a query point will be on the skyline in logarithmic time.

It would be interesting to construct a data structure in high dimensions that has size near-linear in m and polynomial in d, and that would allow for querying the probability a point is on the skyline in logarithmic time.

An important direction is determining the empirical implications of these algorithms: how well do they work in practice on real-world data sets? Since they are based on well-studied data structures, we suspect that they can be made I/O efficient, and hence suitable for enormous data sets.

Another important empirical direction is exploring the best (most useful) concise representation of an approximate skyline. Returning all uncertain points which have probability greater than ρ of being on the skyline can produce both large answers, and not contain entire regions of the skyline. Since a skyline is supposed to be a concise summary, and uncertainty (and inherently approximation) is involved, we argue that returning sparse approximate skylines are important, rather than all points on or near the skyline. A consumer of this data will likely not care about two nearly identical points that occupy similar parts of the skyline. In higher dimensions, providing approximate levels of uncertain skylines with guarantees remains a challenge (as with similar problems on precise data [16]). A similar argument for the importance of concise skylines is made regarding work on k-dominant skylines [6] and other formulations [24, 12]; the difference is that our work provides approximation guarantees in the value of the attributes and operates on data that has defined uncertainty.

5. REFERENCES

[1] P. K. Agarwal and M. Sharir. Arrangements of surfaces in higher dimensions. In J. Sack and J. Urrutia, editors, *Handbook of Computational Geometry*, pages 49–119. North-Holland, 2000.

[2] P. Agrawal, O. Benjelloun, A. D. Sarma, C. Hayworth, S. Nabar, T. Sugihara, and J. Widom. Trio: A system for data, uncertainty, and lineage. In *ACM Symposium on Principles of Database Systems*, 2006.

[3] M. J. Atallah and Y. Qi. Computing all skyline probabilities for uncertain data. In *ACM Symposium on Principles of Database Systems*, pages 279–287, 2009.

[4] J. L. Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.

[5] S. Börzsönyi, D. Kossman, and K. Stocker. The skyline operator. In *IEEE International Conference on Data Engineering*, 2001.

[6] C.-Y. Chan, H. V. Jagadish, K.-L. Tan, A. K. H. Tung, and Z. Zhang. Finding k-dominant skylines in high dimensional space. In *ACM-SIGMOD International Conference on Management of Data*, 2006.

[7] R. Cheng, D. V. Kalashnikov, and S. Prabhakar. Evaluating probabilitic queries over imprecise data. In *ACM-SIGMOD International Conference on Management of Data*, 2003.

[8] G. Cormode, A. Deligiannakis, M. Garafalakis, and A. McGregor. Probabilistic histograms for probabilistic data. In *International Conference on Very Large Data Bases*, 2009.

[9] G. Cormode and M. Garafalakis. Histograms and wavelets of probabilitic data. In *IEEE International Conference on Data Engineering*, 2009.

[10] G. Cormode, F. Li, and K. Yi. Semantics of ranking queries for probabilistic data and expected ranks. In *IEEE International Conference on Data Engineering*, 2009.

[11] N. Dalvi and D. Suciu. Efficient query evaluation on probabilitic databases. *The VLDB Journal*, 16:523–544, 2007.

[12] A. Das Sarma, A. Lall, D. Nanongkai, R. J. Lipton, and J. Xu. Representative skylines using threshold-based

preference distributions. In *IEEE International Conference on Data Engineering*, 2011.

[13] A. Das Sarma, A. Lall, D. Nanongkai, and J. Xu. Randomized mutli-pass streaming skyline algorithms. In *International Conference on Very Large Data Bases*, 2009.

[14] M. de Berg, O. Cheong, M. van Kreveld, and M. Overmars. *Computational Geometry Algorithms and Applications*. Springer, 2008.

[15] H. Edelsbunner, L. J. Guibas, and J. Stolfi. Optimal point location on a monotone subdivision. *SIAM Journal of Computing*, 15:317–340, 1986.

[16] V. Koltun and C. H. Papadimitriou. Approximately dominating representatives. *Theoretical Computer Science*, 371:148–154, 2007.

[17] D. Kossman, F. Ramsak, and S. Rost. Shooting stars in the sky: an optimal algorithm for skyline queries. In *International Conference on Very Large Data Bases*, 2002.

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

[19] J. Li, B. Saha, and A. Deshpande. A unified approach to ranking in probabilistic databases. In *International Conference on Very Large Data Bases*, 2009.

[20] X. Lian and L. Chen. Monochromatic and bichromatic reverse skyline search over uncertain databases. In *ACM-SIGMOD International Conference on Management of Data*, 2008.

[21] M. Löffler and J. M. Phillips. Shape fitting of point sets with probability distributions. In *European Symposium on Algorithms*, 2009.

[22] M. Löffler and J. Snoeyink. Delaunay triangulations of imprecise points in linear time after preprocessing. In *Symposium on Computational Geometry*, pages 298–304, 2008.

[23] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.

[24] D. Nanongkai, A. Das Sarma, A. Lall, R. J. Lipton, and J. Xu. Regret-minimizing representative databases. In *International Conference on Very Large Data Bases*, 2010.

[25] D. Papadias, Y. Tao, G. Fu, and B. Seeger. An optimal and progressive algorithm for skyline queries. In *ACM-SIGMOD International Conference on Management of Data*, 2003.

[26] J. Pei, B. Jiang, X. Lin, and Y. Yuan. Probabilistic skylines on uncertain data. In *International Conference on Very Large Data Bases*, 2007.

[27] F. P. Preparata and M. I. Shamos. *Computational Geometry An Introduction*. Springer, 1985.

[28] K.-L. Tan, P.-K. Eng, and B. C. Ooi. Efficient progressive skyline computation. In *International Conference on Very Large Data Bases*, 2001.

[29] Y. Tao, R. Cheng, X. Xiao, W. K. Ngai, B. Kao, and S. Prabhakar. Indexing multi-dimensional uncertain data with arbitrary probability density functions. In *International Conference on Very Large Data Bases*, 2005.

[30] D. E. Willard. New data structures for orthogonal range queries. *SIAM Journal of Computing*, 14(1):232–253, 1985.

[31] W. Zhang, X. Lin, Y. Zhang, W. Wang, and J. X. Yu. Probabilistic skyline operator over sliding windows. In *IEEE International Conference on Data Engineering*, 2009.