The most-likely skyline problem for stochastic points

Akash Agrawal[*] Yuan Li[†] Jie Xue[‡] Ravi Janardan[§]

Abstract

For a set O of n points in $\mathbb{R}^d$, the *skyline* consists of the subset of all points of O where no point is dominated by any other point of O. Suppose that each point $o_i \in O$ has an associated probability of existence $p_i \in (0, 1]$. The problem of computing the skyline with the maximum probability of occurrence is considered. It is shown that in $\mathbb{R}^d$, $d \geq 3$, the problem is NP-hard and that the desired skyline cannot even be well-approximated in polynomial-time unless $P = NP$. In $\mathbb{R}^2$, an optimal $\mathcal{O}(n \log n)$-time and $\mathcal{O}(n)$-space algorithm is given.

1 Introduction

In $\mathbb{R}^d$, a point u *dominates* a point v if each coordinate of u is at least as large as the corresponding coordinate of v, with strict inequality in at least one dimension. The *skyline* of a set of points consists of the subset of all points where no point is dominated by any other point of the set. (See Figure 1a.) The skyline (or *Pareto set* or *maximal vector*) is useful in multi-criteria decision-making as it yields a set of viable candidates for further exploration. It has been well-studied in the database, optimization, and computational geometry literature; e.g., [3, 5, 6].

We investigate skylines in a setting where there is uncertainty associated with the existence of the points. Such stochastic datasets can model, for instance, experimental observations with associated confidence values or physical entities that may not always be available (e.g., sensors whose activity level depends on battery life or hotel rooms where availability depends on demand).

We consider the problem of computing the skyline that has the greatest probability of being present, hence the one that the user is most likely to encounter and explore further. We call this the most-likely skyline. Our results include an optimal algorithm in the plane and hardness results in higher dimensions.

[*]Dept. of Computer Science and Engg., Univ. of Minnesota-Twin Cities, akash@umn.edu

[†]Dept. of Computer Science and Engg., Univ. of Minnesota-Twin Cities, lixx2100@umn.edu

[‡]Dept. of Computer Science and Engg., Univ. of Minnesota-Twin Cities, xuexx193@umn.edu

[§]Dept. of Computer Science and Engg., Univ. of Minnesota-Twin Cities, janardan@umn.edu

1.1 Problem formulation, contributions, and related work

Let $O = \{o_1, o_2, \ldots, o_n\}$ be a set of points in $\mathbb{R}^d$, where $x_k(o_i)$ denotes the kth coordinate of o_i. Point o_i *dominates* o_j (i.e., $o_i \succ o_j$) if $x_k(o_i) \geq x_k(o_j)$ for $1 \leq k \leq d$, with strict inequality in at least one dimension.

Suppose that each $o_i \in O$ has an associated real $p_i \in (0, 1]$. (The p_i's are known and independent of each other.) We call p_i (resp. $q_i = 1 - p_i$) the *existence* (resp. *non-existence*) *probability* of o_i and call O a *stochastic set*.

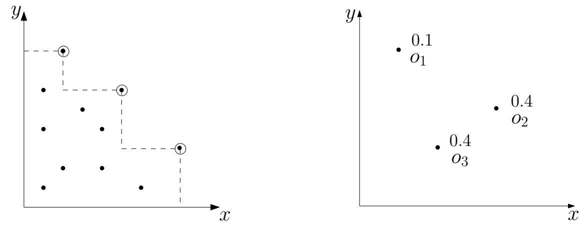
Let $O' \subseteq O$, where no point of O' dominates another of O'; thus, O' itself is the skyline of O'. Now, when is O' also a skyline of O? Let $F(O') \subseteq O \setminus O'$ be the points that are not dominated by any point of O'; intuitively, these are the points "above" the staircase contour defined by O'. Clearly, as long as no point of $F(O')$ is present, O' is also the skyline of O. (The points of $O \setminus O'$ that are dominated by one or more points of O', i.e., the ones "below" the staircase, do not affect the skyline property of O'.) Thus, for O' to be a skyline of O, each point of O' must be present and no point of $F(O')$ should be present. So, the probability that O' is a skyline of O is $PrSky(O') = \prod_{o_i \in O'} p_i \times \prod_{o_i \in F(O')} q_i$.

Our problem is to compute a skyline O' of O for which $PrSky(O')$ is maximum. This skyline, called the *most-likely skyline* of O, is denoted by $MLSky(O)$. (See Figure 1b and Figure 1c for an example.)

Note that the introduction of uncertainty makes our problem challenging as there might be an exponential number of candidate skylines—as many as one for each possible subset of existent points. By contrast, in the non-stochastic setting, there is exactly one skyline for a given set of points.

We make three contributions to the most-likely skyline problem. We prove that computing such a skyline is NP-hard in $\mathbb{R}^3$, hence also in $\mathbb{R}^d$ for $d > 3$ (Section 2). Furthermore, we prove that the most-likely skyline in $\mathbb{R}^d$ ($d \geq 3$) cannot even be well-approximated in polynomial-time unless $P = NP$ (Section 3). We complement these results with an $\mathcal{O}(n \log n)$-time and $\mathcal{O}(n)$-space algorithm to compute the most-likely skyline in $\mathbb{R}^2$ (Section 4), which is optimal in the comparison model due to the known $\Omega(n \log n)$ lower bound for the non-stochastic skyline problem [5].

To our knowledge, this paper is the first to consider skylines in the *unipoint stochastic model*, where



(a) (b)

O'	$F(O')$	$PrSky(O')$
$\emptyset$	$\{o_1, o_2, o_3\}$	$(1-0.1)\times(1-0.4)\times(1-0.4) = 0.324$
$\{o_1\}$	$\{o_2, o_3\}$	$0.1\times(1-0.4)\times(1-0.4) = 0.036$
$\{\mathbf{o_2}\}$	$\{\mathbf{o_1}\}$	$\mathbf{(1-0.1)\times 0.4 = 0.36}$
$\{o_3\}$	$\{o_1, o_2\}$	$(1-0.1)\times(1-0.4)\times 0.4 = 0.216$
$\{o_1, o_2\}$	$\emptyset$	$0.1\times 0.4 = 0.04$
$\{o_1, o_3\}$	$\{o_2\}$	$0.1\times(1-0.4)\times 0.4 = 0.024$

(c)

Figure 1: (a) Skyline of a conventional (i.e., non-stochastic) point-set, with skyline points circled. (b) A set, O, of stochastic points and associated existence probabilities. (c) Computing the most-likely skyline for the example in (b); here it consists of just $\{o_2\}$. (Note that if p_2 is decreased to 0.2 and the example is re-worked, then the most-likely skyline is $\emptyset$, with probability 0.432.)

the points have fixed locations and associated existence probabilities. An alternative setting is the *multipoint stochastic model*, where each point is described by discretely-many locations, with associated existence probabilities, or by a continuous probability distribution. Examples of work here include computing skylines whose points have existence probabilities above a threshold [7], computing for each dataset point (or for a query point) the probability that it is not dominated by any other point [1, 2], computing stochastic skyline operators to find a minimum set of candidate points with respect to a certain scoring function [9], etc.

2 NP-Hardness of computing the most-likely skyline in $\mathbb{R}^d$, $d \geq 3$

We give a polynomial-time reduction from the minimum ε-ADR problem in $\mathbb{R}^3$, which is known to be NP-hard [4], to the most-likely skyline problem in $\mathbb{R}^3$. (Here ADR stands for "Approximately Dominating Representatives" [4].)

An instance of the ε-ADR problem in $\mathbb{R}^3$ consists of a set, S, of n (non-stochastic) points and a real $\varepsilon > 0$. An *ε-ADR* of S is a set $S' \subseteq S$ such that every $s \in S$ is dominated by some $s' \in S'$ when s' is boosted by ε, i.e., $(1+\varepsilon)\cdot s' \succ s$. The *minimum ε-ADR problem* seeks the smallest such set S'.

The reduction: Given any ε-ADR instance in $\mathbb{R}^3$, we compute the (conventional) skyline, $Sky(S)$, of S and boost its points by ε to get a set $\hat{S}$. To each point in $Sky(S)$ (resp. $\hat{S}$) we assign an existence probability β (resp. α), where $1/3 < \alpha < 1/2 < \beta < 1$ and $\beta(1-\alpha) < \alpha$; e.g., $\alpha = 0.4$ and $\beta = 0.6$. The set $\bar{S} = Sky(S) \cup \hat{S}$ is an instance of the most-likely skyline problem in $\mathbb{R}^3$. The reduction takes polynomial time. We observe the following:

(i) The probability that the skyline of $\bar{S}$ is empty is the probability that no point of $\bar{S}$ exists, i.e., $(1-\beta)^K(1-\alpha)^K$, where $K = |Sky(S)| = |\hat{S}|$. Since $\beta > 1/2$, the probability of the empty skyline is less than $\beta^K(1-\alpha)^K$. The latter is the probability of that skyline of $\bar{S}$ where the points of $Sky(S)$ exist and those of $\hat{S}$ do not. Thus, the most-likely skyline of $\bar{S}$ is non-empty.

(ii) Consider the skyline of $\bar{S}$ that consists of just $Sky(S)$. Suppose that a point $s \in Sky(S)$ is replaced by a point $\hat{s} \in \hat{S}$ that dominates it. The probability expression for $Sky(S)$ contains terms β and $1-\alpha$, since s is present in it and $\hat{s}$ is not. In the probability expression for the new skyline, the term $1-\alpha$ is replaced by α, since $\hat{s}$ is present and the term β is excluded since s is not present. (The term β is not replaced by $1-\beta$ since s is dominated by $\hat{s}$. Indeed, $\hat{s}$ may also dominate other points of $Sky(S)$, so the term β for each such point is also excluded.) Since $\beta(1-\alpha) < \alpha$, the new skyline has a higher probability than the previous skyline of $\bar{S}$. The replacement process is continued until a skyline consisting of only points drawn from $\hat{S}$ is obtained. Since each replacement yields a skyline of higher probability, it follows that the most-likely skyline of $\bar{S}$ consists of only points from $\hat{S}$. Let $k \leq K$ be the number of such points from $\hat{S}$.

(iii) The points of $\hat{S}$ are mutually non-dominating since their pre-images are in $Sky(S)$. Thus, each point of $\hat{S}$ that is not in the most-likely skyline of $\bar{S}$ contributes $(1-\alpha)$ to the probability of this skyline, so its probability is $\alpha^k(1-\alpha)^{K-k} = (\alpha/(1-\alpha))^k(1-\alpha)^K$. Since $\alpha < 1/2$, we have $\alpha/(1-\alpha) < 1$. Since $(1-\alpha)^K$ is fixed for a given set S and the skyline under consideration has maximum probability, it follows that k is minimum.

Suppose that there is a polynomial-time algorithm for the most-likely skyline problem in $\mathbb{R}^3$. We generate $\bar{S}$ and compute its most-likely skyline, all in polynomial time. These skyline points, which are a subset of $\hat{S}$ of some minimum size k, dominate the points of $Sky(S)$, hence also the points of S. Therefore, the pre-images of these skyline points in S (i.e., prior to boosting) are an ε-ADR of S of minimum size k and can be computed in polynomial time. This contradicts the known NP-hardness of the ε-ADR problem in $\mathbb{R}^3$ and yields the following theorem.

Theorem 1 *The most-likely skyline problem in $\mathbb{R}^d$, $d \geq 3$, is NP-hard.*

3 Inapproximability in $\mathbb{R}^d$, $d \geq 3$

Let $\mathcal{A}$ be an algorithm that computes a skyline whose probability is greater than c times the probability of the most-likely skyline, $0 < c < 1$. We call $\mathcal{A}$ a *c-approximation* algorithm. Our first result is based on the reduction in Section 2.

Theorem 2 *For $c > 1/2$, there exists no polynomial-time c-approximation algorithm, $\mathcal{A}$, for the most-likely skyline problem in $\mathbb{R}^d$, $d \geq 3$, unless $P = NP$.*

Proof. We show that $\mathcal{A}$ cannot exist for $c = \alpha/(1-\alpha)$, where $1/3 < \alpha < 1/2$. (Recall that α is the existence probability assigned to each point of $\hat{S}$ in Section 2.) The theorem follows since $\alpha > 1/3$ implies $c > 1/2$. (A similar result is stated in [8] for the most-likely convex hull problem.)

Suppose $\mathcal{A}$ exists for $c = \alpha/(1-\alpha)$. We run $\mathcal{A}$ on $\bar{S} = Sky(S) \cup \hat{S}$. The probability of the resulting skyline is greater than $\alpha/(1-\alpha)$ times the probability of the most-likely skyline of $\bar{S}$, i.e., greater than $(\alpha/(1-\alpha)) \cdot (\alpha^k(1-\alpha))^{K-k}) = \alpha^{k+1}(1-\alpha)^{K-(k+1)}$. Assume without loss of generality that the computed skyline contains points from $\hat{S}$ only. (If it contains points of $Sky(S)$, then each of these can be replaced, in polynomial time, by the corresponding boosted point in $\hat{S}$ and the probability of the resulting skyline only increases.)

How many points does the computed skyline have? Note that a skyline with $k+1$ points of $\hat{S}$ has probability $\alpha^{k+1}(1-\alpha)^{K-(k+1)}$. For each additional point of $\hat{S}$ that is included in the skyline, the probability gets multiplied by a factor $\alpha/(1-\alpha)$, which is less than 1 since $\alpha < 1/2$. It follows that the skyline computed by $\mathcal{A}$ has fewer than $k+1$ points of $\hat{S}$. Thus, the ε-ADR of S corresponding to the pre-images of the points of the skyline computed by $\mathcal{A}$ has fewer than $k+1$ points. This ε-ADR of S must be a minimum ε-ADR of S, since the latter has size k. This yields a polynomial-time algorithm for computing a minimum ε-ADR, which is not possible unless $P = NP$. □

We can use Theorem 2 and the notion of "product composability" to show that, for any $\delta > 0$, there is not even a polynomial-time $2^{-\mathcal{O}(n^{(1-\delta)})}$-approximation algorithm for the most-likely skyline problem in $\mathbb{R}^d$, $d \geq 3$, unless $P = NP$.

An optimization problem is *product composable* [8] if any given set of problem instances $I_1, \ldots, I_k$ can be combined to yield a new instance I^* whose objective function is expressible as the product of the objective functions of $I_1, \ldots, I_k$. We require that $|I^*| = \sum_{j=1}^{k} |I_j|$, that I^* is constructible in time polynomial in $|I^*|$, and that there exists a polynomial-time computable bijection between the set of feasible solutions of I^* and those of $I_1, \ldots, I_k$.

The following lemma relates product composability to inapproximability.

Lemma 3 ([8]) *If a maximization problem of size n is product composable and cannot be approximated within a constant $c < 1$ in polynomial time, then it has no polynomial-time $2^{-\mathcal{O}(n^{(1-\delta)})}$-approximation algorithm, for any $\delta > 0$.*

A proof of this lemma can be found in Appendix G of [8] (full version).

Intuitively, the lemma is proved by showing that the existence of a $2^{-\mathcal{O}(n^{(1-\delta)})}$-approximation algorithm together with product composability would imply the existence of a c-approximation algorithm. Recall that Theorem 2 has established that, for $1/2 < c < 1$, no c-approximation algorithm exists for the most-likely skyline problem, unless $P = NP$. In fact, the proof of Theorem 2 shows that this is true even for the subset of instances consisting of the set $\bar{S} = Sky(S) \cup \hat{S}$ and the associated probabilities, as defined in Section 2.

So it suffices to show that the most-likely skyline problem consisting of instances $\bar{S} = Sky(S) \cup \hat{S}$ and the aforementioned probabilities is product composable. Specifically, we form the instances $I_1, \ldots, I_k$ by partitioning $\bar{S}$ using the k points on its most-likely skyline.

Let $\hat{S}' = \{\hat{s}_1, \ldots, \hat{s}_k\} \subseteq \hat{S}$ be the points on the most-likely skyline of $\bar{S}$, sorted by non-increasing x_1-coordinates. For each $\hat{s}_i \in \hat{S}'$, we define two sets S_i and $\hat{S}_i$. S_i contains points $s_j \in Sky(S)$ such that $\hat{s}_i \succ s_j$, $\hat{s}_l \not\succ s_j$ for $l < i$ and either $(1+\varepsilon) \cdot s_j = \hat{s}_i$ or $(1+\varepsilon) \cdot s_j \notin \hat{S}'$. $\hat{S}_i$ contains the boosted points of S_i. For $1 \leq i \leq k$, let $I_i = S'_i = S_i \cup \hat{S}_i$. (See Figure 2.) It is easy to verify that $I_1, \ldots, I_k$ can be combined to form a new instance, I^*, in polynomial-time such that,

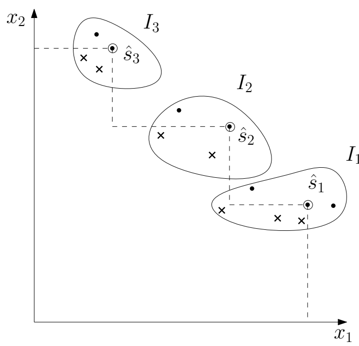


Figure 2: Example of a partition of $\bar{S}$ to form the problem instances I_1, I_2, and I_3. The points in $Sky(S)$ are represented by crosses ($\times$) and the boosted points, i.e., points in $\hat{S}$, are represented by disks ($\bullet$). Points on the most-likely skyline are circled.

given the most-likely skyline for $I_1, \ldots, I_k$, the most-likely skyline for I^* can be computed in polynomial-time, and vice-versa. This establishes product composability. Lemma 3 now yields the following result.

Theorem 4 *For any $\delta > 0$, there is no polynomial-time $2^{-\mathcal{O}(n^{(1-\delta)})}$-approximation algorithm for computing the most-likely skyline of n stochastic points in $\mathbb{R}^d$, $d \geq 3$, unless $P = NP$.*

Finally, we note that there is a simple, but uninteresting, polynomial-time 2^{-n}-approximation algorithm for the most-likely skyline problem: Simply compute the skyline of the points of S whose existence probability is more than $1/2$. (A similar observation appears in [8] for the most-likely convex hull problem.)

4 An efficient algorithm in $\mathbb{R}^2$

We now describe an algorithm to compute the most-likely skyline, $MLSky(O)$, for a set O of n points, in $\mathbb{R}^2$. Our algorithm runs in $\mathcal{O}(n \log n)$ time and $\mathcal{O}(n)$ space, which is optimal in the comparison model.

We assume w.l.o.g. that all points of O are in the first quadrant and that no two points have the same x_1- or x_2-coordinate. Let the points of O be $o_1, o_2, \ldots o_n$, in decreasing order of their x_1-coordinates. For convenience, we augment O with dummy points $o_0 = (\infty, 0)$ and $o_{n+1} = (0, \infty)$, with $p_0 = p_{n+1} = 1$. The proof of the following lemma is fairly easy, hence omitted.

Lemma 5 *For any subset O' of O, $O' = MLSky(O)$ iff $O' \cup \{o_0, o_{n+1}\} = MLSky(O \cup \{o_0, o_{n+1}\})$.*

Based on Lemma 5, we augment the input set with o_0 and o_{n+1}, and, hereafter, we will focus on finding the most-likely skyline for $\{o_0, o_1, \ldots, o_n, o_{n+1}\}$. For notational convenience, let $O_i = \{o_0, o_1, \ldots, o_i\}$.

We sweep a vertical line from right to left over O, stopping at each point o_i. At o_i we compute the most-likely skyline of O_i subject to the constraint that o_i belongs to this skyline. We denote this optimal skyline by $\mathcal{S}(o_i)$. We initialize $\mathcal{S}(o_0) = \{o_0\}$ and report $\mathcal{S}(o_{n+1}) - \{o_0, o_{n+1}\}$ as $MLSky(O)$.

Let $F(\mathcal{S}(o_i))$ be the set of points of $O_i \setminus \mathcal{S}(o_i)$ that are not dominated by any points in $\mathcal{S}(o_i)$. Then, the probability of $\mathcal{S}(o_i)$ being the skyline of O_i is

$$PrSky(\mathcal{S}(o_i)) = \prod_{o_k \in \mathcal{S}(o_i)} p_k \times \prod_{o_k \in F(\mathcal{S}(o_i))} q_k.$$

Let o_j be the point in $\mathcal{S}(o_i)$ with largest x_2-coordinate smaller than $x_2(o_i)$ and let $R(o_i, o_j) = (x_1(o_i), x_1(o_j)) \times (x_2(o_j), \infty)$. (Figure 3.) Let $F_R(\mathcal{S}(o_i))$ be the subset of $F(\mathcal{S}(o_i))$ lying in $R(o_i, o_j)$. Then,

$$PrSky(\mathcal{S}(o_i)) = p_i \times \left(\prod_{o_k \in F_R(\mathcal{S}(o_i))} q_k \times \prod_{o_k \in (\mathcal{S}(o_i) \setminus \{o_i\})} p_k \times \prod_{o_k \in (F(\mathcal{S}(o_i)) \setminus F_R(\mathcal{S}(o_i)))} q_k \right).$$

Thus, we can write

$$PrSky(\mathcal{S}(o_i)) = p_i \times score_{o_i}(o_j),$$

where $score_{o_i}(o_j)$ is the expression inside the large parentheses in the above equation.

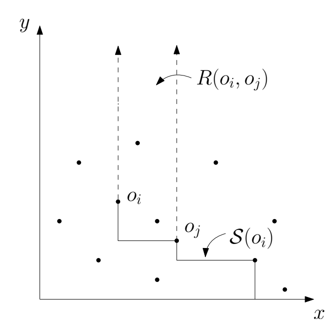


Figure 3: Illustrating $\mathcal{S}(o_i)$ and $R(o_i, o_j)$. (Dummy points o_0 and o_{n+1} are not shown.)

Since p_i is fixed for o_i and $PrSky(\mathcal{S}(o_i))$ is maximum, it follows that $score_{o_i}(o_j)$ must be maximum. For the given pair (o_i, o_j), the first term in $score_{o_i}(o_j)$, i.e., $(\prod_{o_k \in F_R(\mathcal{S}(o_i))} q_k)$, is fixed. Thus, for $score_{o_i}(o_j)$ to be maximum, the product of the second and third terms must be maximum. This product is nothing but the probability of the most-likely skyline of O_j subject to the constraint that o_j belongs to the skyline, i.e., $PrSky(\mathcal{S}(o_j))$. Hence, $score_{o_i}(o_j) = (\prod_{o_k \in F_R(\mathcal{S}(o_i))} q_k) \times PrSky(\mathcal{S}(o_j))$.

Thus the recurrence for $\mathcal{S}(o_i)$ is:

$$\mathcal{S}(o_i) = \begin{cases} \{o_i\}, & \text{if } i = 0, \\ \{o_i\} \cup \mathcal{S}(o_j), & \text{otherwise,} \end{cases}$$

where $o_j = \underset{\substack{o_j \in O_{i-1}; \\ x_2(o_i) > x_2(o_j)}}{\arg\max} \{score_{o_i}(o_j)\}$.

This recurrence leads naturally to a dynamic programming algorithm that can be implemented easily to run in $\mathcal{O}(n^2)$ time, where the run time is dominated by time to maintain the first term, i.e., $(\prod_{o_k \in F_R(\mathcal{S}(o_i))} q_k)$, in $score_{o_i}(o_j)$ for all relevant pairs (o_i, o_j). The run time can be improved to $\mathcal{O}(n \log n)$ through a more careful approach, as follows.

When the sweep is at o_i, we will, for the sake of brevity, refer to $(\prod_{o_k \in F_R(\mathcal{S}(o_i))} q_k)$, $score_{o_i}(o_j)$, and $PrSky(\mathcal{S}(o_j))$ as the *R-value*, *S-value*, and *P-value* of o_j, respectively. Note that the S-value of o_j is the product of the R-value and the P-value of o_j. Note also that the R-value and S-value of o_j depend on o_i, too, but since we refer to these when the sweep is at o_i, we omit the reference to o_i. (Similarly, if the sweep is later at some other point.)

Just after o_i is processed in the right-to-left sweep, let o be some point of O for which we wish to compute $\mathcal{S}(o)$. Let o_j be any point of O to the right of and below o_i. Then, if o is above o_j then o_i will be in the rectangle $R(o, o_j)$ and so q_i will need to be included in the R-value of o_j when the sweep reaches o. Thus, when we have finished processing o_i, we preemptively multiply by q_i the R-value of each o_j to the right of and below o_i. All such points o_j would have already been encountered in the sweep and their y-coordinates will lie in the range $[0, x_2(o_i))$, so the multiplication can be done for all o_j in this range efficiently by grouping them into a small number of sets. This observation along with a suitable data structure is the key to realizing the improved run time.

Let $\mathcal{D}$ be a data structure on O which supports the following operations when the sweepline is at o_i.

- *FindMax_S-value*(o_i): Returns the maximum of the S-values associated with the points whose y-coordinates are in the range $[0, x_2(o_i))$, along with the corresponding point o_j.
- *Set_P-value*(o_i, μ): Sets the P-value of o_i to μ. (In our algorithm, μ will be p_i times the S-value returned by *FindMax_S-value*(o_i), which is executed just before *Set_P-value*(o_i, μ).)
- *RangeMult_R-value*(o_i): Multiplies by q_i the R-values of the points whose y-coordinates are in the range $[0, x_2(o_i))$.

Note that the range $[0, x_2(o_i))$ used in *FindMax_S-value*(o_i) and *RangeMult_R-value*(o_i) may include points that are below and to the left of o_i, hence have not yet been seen in the sweep. However, $\mathcal{D}$ is set up so that the P-value (hence the S-value) of any point that has not yet been seen in the sweep is zero. Thus, such points are effectively ignored when the sweep is at o_i. This approach obviates the need to make $\mathcal{D}$ dynamic.

In Section 4.1 we show that $\mathcal{D}$ can be implemented so that it supports the above operations in $\mathcal{O}(\log n)$ time using $\mathcal{O}(n)$ space. Given this it should be clear that the algorithm runs in $\mathcal{O}(n \log n)$ time and $\mathcal{O}(n)$ space as it involves doing at each point of $o_i \in O$ (other than at o_0), one *FindMax_S-value*(o_i), one *Set_P-value*(o_i, μ), and one *RangeMult_R-value*(o_i) operation, in that order. (After each *FindMax_S-value*(o_i), $\mathcal{S}(o_i)$ is updated by including o_i in $\mathcal{S}(o_j)$.) This leads to the following conclusion.

Theorem 6 *The most-likely skyline of a set of n stochastic points in $\mathbb{R}^2$ can be computed in $\mathcal{O}(n \log n)$ time using $\mathcal{O}(n)$ space.*

4.1 Implementing the data structure $\mathcal{D}$

We implement $\mathcal{D}$ as a 1-dimensional range tree, i.e., a balanced binary search tree where the x_2-coordinates of the points of O are stored at the leaves, in increasing order from left to right. We maintain several fields at the nodes of $\mathcal{D}$, whose meanings are defined relative to the position of the sweep at the current point o_i.

During the sweep, we need to keep track of the R-value of each point o_j that is to the right of and below o_i. Rather than doing this explicitly for each such o_j, which would be expensive, we accumulate the R-value for o_j as the product of certain real numbers stored at the leaf containing o_j and the ancestors of this leaf. Specifically, let $prod(v)$ be a real-valued field at any node v. Then the R-value of o_j (relative to the current position of the sweep at o_i) is the product of the $prod(\cdot)$ fields at the leaf containing o_j and its ancestors. (Note that when the sweep is at o_i, the R-value is irrelevant for points that are to the right of and above o_i, and undefined for points that are to the left of o_i.) We initialize $prod(v)$ to 1 for all nodes of $\mathcal{D}$.

At each leaf w, besides $prod(w)$, we store three additional fields $pt(w)$, $pval(w)$, and $val(w)$. Here $pt(w)$ is the point whose x_2-coordinate is stored at w, $pval(w)$ is the P-value of the point if it has already been seen in the sweep and zero otherwise, and $val(w)$ is $prod(w) \times pval(w)$. We initialize $pval(w)$ to 1 for the leaf w corresponding to o_0 and to zero for all other leaves.

At each non-leaf v, besides $prod(v)$, we store two additional fields $val(v)$ and $pt(v)$, whose meanings are as follows: Let $\mathcal{D}(v)$ be the subtree of $\mathcal{D}$ rooted at v. Among the leaves of $\mathcal{D}(v)$, let w be the one for which the product of $pval(w)$ and the $prod(\cdot)$ fields at w and its ancestors, up to and including v, is maximum. Then $val(v)$ stores this maximum product and $pt(v)$ equals $pt(w)$. Thus the maximum S-value among the leaves in $\mathcal{D}(v)$ is the product of $val(v)$ and the $prod(\cdot)$ fields at the proper ancestors of v. (Note that any leaf in $\mathcal{D}(v)$ corresponding to a point that has yet to be seen in the sweep cannot realize the maximum product as its $pval(\cdot)$ field is zero.)

As we will see below, when searching downwards in $\mathcal{D}$ during any of the aforementioned operations, if we are at a non-leaf node v then we will multiply the $prod(\cdot)$ field and the $val(\cdot)$ field of each child of v by $prod(v)$ and then reset $prod(v)$ to 1. This ensures that (a) at any node on the search path, the maximum S-value among the leaves in its subtree is equal to the node's $val(\cdot)$, and (b) the value that was originally in $prod(v)$ will continue to be applied to the points in the subtree of each of v's children. This as-needed, lazy approach to propagating the values in the $prod(\cdot)$ fields allows us to implement the operations efficiently.

We now describe how to do the operations on $\mathcal{D}$.

- *FindMax_S-value*(o_i): We search downwards in $\mathcal{D}$ with $x_2(o_i)$ and identify a set, $\mathcal{C}$, of *canonical nodes*, as follows: Whenever the search at a node v goes to the right child, we include the left child v' in $\mathcal{C}$. Thus, the leaves of the $\mathcal{D}(v')$'s yield a grouping of the points of O lying in the range $[0, x_2(o_i))$ into $\mathcal{O}(\log n)$ subsets.

  During the search down $\mathcal{D}$, when we are at a non-leaf node v, we update $prod(u)$ to $prod(u) \times prod(v)$ (resp. $val(u)$ to $val(u) \times prod(v)$) for each child u of v, and then reset $prod(v)$ to 1. Finally, we return the maximum of the $val(v')$'s, taken over all nodes v' in $\mathcal{C}$.

- *Set_P-value*(o_i, μ): We search downwards in $\mathcal{D}$ with $x_2(o_i)$ to find the leaf w containing o_i. At each non-leaf node v in the search, we update $prod(u)$ to $prod(u) \times prod(v)$ (resp. $val(u)$ to $val(u) \times prod(v)$) for each child u of v, and then reset $prod(v)$ to 1.

  At w, we set $pt(w)$ to o_i, $prod(w)$ to 1, and both $pval(w)$ and $val(w)$ to μ. We then walk back up $\mathcal{D}$ towards the root and, at each node v visited, we update $val(v)$ to the larger of the $val(\cdot)$ of its children and update $pt(v)$ accordingly.

- *RangeMult_R-value*(o_i): We search downwards in $\mathcal{D}$ with $x_2(o_i)$ and identify the set $\mathcal{C}$ of canonical nodes, as we did in *FindMax_S-value*(o_i). At each non-leaf node v in the search, we update $prod(u)$ to $prod(u) \times prod(v)$ (resp. $val(u)$ to $val(u) \times prod(v)$) for each child u of v, and then reset $prod(v)$ to 1. Next, for each node v' in $\mathcal{C}$, we update $prod(v')$ to $prod(v') \times q_i$. Finally, starting at the lowest node in $\mathcal{C}$ we walk back up $\mathcal{D}$ towards the root and, at each node v visited, we update $val(v)$ to the larger of the $val(\cdot)$ of its children and update $pt(v)$ accordingly.

It should be evident from the preceding discussion that $\mathcal{D}$ implements the operations correctly in $\mathcal{O}(\log n)$ time apiece and uses $\mathcal{O}(n)$ space.

5 Conclusion

Given a set of points in $\mathbb{R}^d$, where each point has a fixed probability of existence, we have considered the problem of computing the skyline that has the greatest probability of existing, i.e., the most-likely skyline. For $d > 2$, we have shown that the problem is NP-hard and, moreover, cannot even be well-approximated unless $P = NP$. For $d = 2$, we have given an optimal algorithm which runs in $\mathcal{O}(n \log n)$ time and uses $\mathcal{O}(n)$ space.

Acknowledgement

The research of the first author was supported, in part, by a Doctoral Dissertation Fellowship from the Graduate School of the University of Minnesota.

References

[1] P. Afshani, P. Agarwal, L. Arge, K. Larsen, and J. Phillips. (Approximate) uncertain skylines. In *Proc. 14th Intl. Conf. on Database Theory*, pages 186–196, 2011.

[2] M. Atallah, Y. Qi, and H. Yuan. Asymptotically efficient algorithms for skyline probabilities of uncertain data. *ACM Trans. on Database Sys.*, 36(2):1–28, 2011.

[3] S. Börzsöny, D. Kossmann, and K. Stocker. The skyline operator. In *Proc. 17th Intl. Conf. on Data Engineering*, pages 421–430, 2001.

[4] V. Koltun and C. H. Papadimitriou. Approximately dominating representatives. *Theoretical Computer Science*, 371(3):148–154, 2007.

[5] 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.

[6] C. Papadimitriou and M. Yannakakis. Multiobjective query optimization. In *Proc. 20th ACM Symp. on Principles of Database Sys.*, pages 52–59, 2001.

[7] J. Pei, B. Jiang, X. Lin, and Y. Yuan. Probabilistic skylines on uncertain data. In *Proc. 33rd Intl. Conf. on Very Large Data Bases*, pages 15–26, 2007.

[8] S. Suri, K. Verbeek, and H. Yıldız. On the most likely convex hull of uncertain points. In *Proc. 21st European Symposium on Algorithms*, pages 791–802. 2013. (Full version at: https://goo.gl/fFlFbS).

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