The τ-Skyline for Uncertain Data

Haitao Wang* Wuzhou Zhang†

Abstract

In this paper, we introduce the notion of τ-*skyline* as an alternative representation for uncertain skylines. Given a parameter $\tau \in [0,1]$ and a set $\mathcal{P}$ of n uncertain points in the plane, where each uncertain point P_i is described by a discrete probability distribution defined over k locations, the τ-*skyline region* of $\mathcal{P}$ is the set of points q in the plane such that the probability of any P_i dominating q is at most τ, and the τ-*skyline* of $\mathcal{P}$ is the boundary of the τ-skyline region of $\mathcal{P}$. We present an $O(nk\log(nk))$ time algorithm for computing the τ-skyline of $\mathcal{P}$. The τ-*skyline probability* of each P_i of $\mathcal{P}$ is defined to be the probability of P_i lying inside the τ-skyline region of $\mathcal{P}$. We show that the τ-skyline probabilities of all P_i's can be computed in $O(nk\log(nk))$ time. Remarkably, our method is very simple and can be easily implemented, a huge potential interest for practice.

1 Introduction

It has long been observed that many real-world measurements are inherently accompanied with uncertainty. As a response, researchers have shown an increased interest in dealing with *uncertain* data, especially in the computational geometry community and the database community nowadays. Among a large number of problems casted under uncertainty, the skyline problem is one of the minions (see e.g., [1, 4, 8, 10]). In this paper, we introduce the notion of τ-*skyline* as an alternative representation for uncertain skylines, and show that τ-skyline can be computed very efficiently. Throughout the paper, we focus on the plane. We begin with some useful concepts and definitions.

xy-Monotone path. We call a path *xy-monotone* if (i) when we move along the path from its one end to the other, the x-coordinate is non-decreasing and the y-coordinate is non-increasing; (ii) it consists of either horizontal or vertical line segments. Note that if we move from left to right along a xy-monotone path, we move either rightwards or downwards.

Dominance relationship. For a point p, denote by

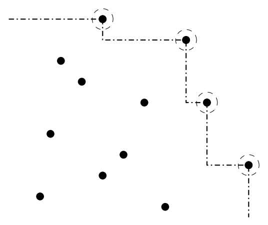

Figure 1: Illustrating the skyline points (the circled points) and the skyline (the dashed path) for a set of exact points. The skyline region is strictly above and to the right of the dashed path.

$x(p)$ and $y(p)$ its x- and y-coordinates, respectively. For two points p and q, we say that p *dominates* q, denoted by $p \succ q$, if $x(p) \geq x(q)$ and $y(p) \geq y(q)$. By this definition, a point dominates itself, which differs from some previous work, e.g., [1, 4] (we will explain later why we make this minor change).

Skyline. Given a set P of n (exact) points, a point $p \in P$ is a *skyline point* of P if p is not dominated by any other point of P. The *skyline region* of P consists of all the points that are not dominated by any point of P. The *skyline* of P is the boundary of the skyline region of P. It is easy to see that the skyline of P is a xy-monotone path connecting all the skyline points of P (e.g., see Fig. 1). All the skyline points (hence the skyline) of P can be computed in $O(n\log n)$ time [7].

1.1 Problem Statement

Next, we describe our model for uncertain points, followed by the definition of τ-skyline.

Uncertain points. Let $\mathcal{P} = \{P_1, P_2, \ldots, P_n\}$ be a set of n *uncertain points*, where each uncertain point P_i is described by a discrete probability distribution defined over k locations $p_{i1}, p_{i2}, \ldots, p_{ik}$, with k being an input parameter, such that the probability of P_i being at location p_{ij} is w_{ij}, where $0 < w_{ij} \leq 1$ for $1 \leq j \leq k$, and $\sum_{j=1}^{k} w_{ij} = 1$. We assume that the probability distributions of P_i's are independent. With a little abuse of notation, we also use P_i to denote the set of locations $\{p_{i1}, \ldots, p_{ik}\}$. For a location $p \in P_i$, we also use $w(p)$ to denote the probability of P_i being at p. Set $m = nk$.

τ-Skyline. Given a point q, the probability that P_i dominates q, denoted by $\delta_i(q)$, is the sum of the proba-

*Department of Computer Science, Utah State University, Logan, UT 84322, USA. E-mail: `haitao.wang@usu.edu`. This work was supported in part by NSF under Grant CCF-1317143.

†Department of Computer Science, Duke University, Durham, NC 27708, USA. E-mail: `wuzhou@cs.duke.edu`.

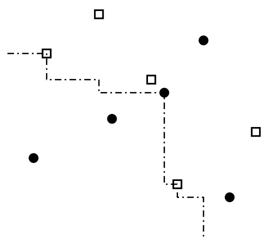

Figure 2: Illustrating a τ-skyline for two uncertain points: one appears in one of the five disk-style points with 0.2 probability each and the other appears in one of the five box-style points with 0.2 probability each. For $\tau = 0.3$, the τ-skyline consists of the dashed segments and the τ-skyline region is strictly above and to the right of the dashed path.

bilities of the locations of P_i that dominate q, i.e.,

$$\delta_i(q) = \sum_{p \in P_i, p \succ q} w(p).$$

For a fixed parameter $\tau \in [0,1]$, the *τ-skyline region* of $\mathcal{P}$, denoted by R, is the set of points q such that the probability of any P_i dominating q is at most τ, i.e., $R = \{q \mid \delta_i(q) \le \tau, \forall i \le n\}$. The *$\tau$-skyline* of $\mathcal{P}$, denoted by π, is the boundary of the τ-skyline region of $\mathcal{P}$. The *τ-skyline probability* of each P_i of $\mathcal{P}$ is defined to be the probability of P_i lying inside the τ-skyline region of $\mathcal{P}$.

We also define the *τ-skyline region* for each $P_i \in \mathcal{P}$, denote by R_i, as the set of points q such that the probability that P_i dominates q is at most τ, i.e., $R_i = \{q \mid \delta_i(q) \le \tau\}$. The boundary of R_i is called the *τ-skyline* of P_i, denoted by π_i. We show later that π_i is a xy-monotone path of size $O(k)$, and can be computed in $O(k \log k)$ time.

Clearly, τ-skyline generalizes the notion of skyline: if $\tau = 0$, the τ-skyline of $\mathcal{P}$ is simply the skyline of $\bigcup_{i=1}^{n} P_i$, and the τ-skyline of P_i is simply the skyline of P_i.

In this paper, we show that the τ-skyline π of $\mathcal{P}$ is a xy-monotone path of size $O(m)$ (e.g., see Fig. 2), and can be computed in $O(m \log m)$ time, where $m = nk$. Furthermore, the τ-skyline probabilities of all P_i's can be computed in $O(m \log m)$ time.

1.2 Related Work

Two models have been widely used for data uncertainty: the *existential model* (see e.g., [5, 6, 11, 13]) and the *locational model* (see e.g., [2, 3, 12]). In the existential model, each uncertain point has a fixed location but it only exists with some probability. In the locational model, each uncertain point always exists but its location follows a probability density function. In this paper, the model of our interest is the locational model.

The skyline problem has been investigated in the locational model, see e.g., [1, 4], where a point was not allowed to dominate itself. Given a point q, the probability that q lies on the skyline of $\mathcal{P}$, called the *skyline probability* of q, is defined to be $\alpha(q, \mathcal{P}) = \prod_{i=1}^{n}(1 - \delta_i(q))$. The probability of P_i of $\mathcal{P}$ being on the skyline of $\mathcal{P}$, called the *skyline probability* of P_i, is defined to be $\sum_{j=1}^{k} w_{ij} \alpha(p_{ij}, \mathcal{P}_{\neq i})$, where $\mathcal{P}_{\neq i} = \mathcal{P} - \{P_i\}$. For a parameter $\rho \in (0,1]$, the goal is to compute the *ρ-skyline* of $\mathcal{P}$, which consists of all the uncertain points of $\mathcal{P}$ whose skyline probabilities are at least ρ.

The ρ-skyline can be easily computed in $O(m^2)$ time, where $m = nk$. Atallah and Qi [4] devised the first sub-quadratic algorithm with time $O(m^{5/3}\text{poly}(\log n))$. Later, Afshani *et al.* [1] improved the running time to $O(m^{3/2})$, and they also constructed a set of uncertain points suggesting that this bound might be optimal. For the case $k \ll n$, the running time can be further improved to $O(mk \log m)$. Moreover, Afshani *et al.* [1] considered approximating the ρ-skyline. Pei *et al.* [10] proposed several heuristic algorithms for computing the ρ-skyline. Other variants of uncertain skylines have also been proposed, see e.g., [8, 14].

The ρ-skyline does not offer us a structure or a region. In view of this, we prefer calling it ρ-skyline (uncertain) points. One may similarly define the ρ-skyline region, though it is not obvious how to compute it efficiently. When ρ is sufficiently large or the skyline probabilities are small enough, the ρ-skyline may not exist. There is a connection between the τ-skyline of $\mathcal{P}$ and the ρ-skyline of $\mathcal{P}$: if a point q lies in the τ-skyline region of $\mathcal{P}$, then the skyline probability of q is at least $(1-\tau)^n$.

Our definition of the τ-skyline of P_i is closely related to the concept of K-skyband, proposed in [9]. Given a set P of n (exact) points and a parameter $K \le n$, the K-skyband of P asks for the set of points which are dominated by at most K points of P. 0-skyband corresponds to the conventional skyline. The τ-skyline of P_i can be used for answering the *weighted* skyband of P_i, where the weight of a point $p \in P_i$ is the probability of P_i being at p, for which we are not aware of any previous work. As a byproduct, we obtain the first algorithm for computing the weighted skyband of a set of weighted (exact) points (note that our proposed algorithm can be extended to arbitrary positive weights).

Unlike previous work [1, 4], we allow a point to dominate itself. We make this minor change only to avoid those tedious discussions later in describing our algorithm and proving its correctness. As will be clear later, if a point is not allowed to dominate itself, the τ-skyline π of $\mathcal{P}$ and our algorithm for computing π remain the same, while the τ-skyline region R of $\mathcal{P}$ may contain some turning points of the τ-skyline π of $\mathcal{P}$ (if a point can dominate itself, R and π are disjoint).

2 Our Algorithm

In this section, we describe our algorithm for computing the τ-skyline of $\mathcal{P}$ in $O(m \log m)$ time, where $m = nk$, for a fixed parameter $\tau \in [0,1]$.

To compute the τ-skyline π of $\mathcal{P}$, we first compute, for each P_i of $\mathcal{P}$, the τ-skyline π_i of P_i, in $O(k \log k)$ time. We then show that π is simply the upper envelope of $\pi_1, \ldots, \pi_n$, and can be computed in $O(m \log m)$ time, where $m = nk$. In the sequel, fix any uncertain point P_i of $\mathcal{P}$. For simplicity of discussion, we assume no two locations of P_i have the same x- or y-coordinate.

As a preprocessing step, we sort all the locations of P_i into two sorted lists L_x and L_y, with L_x by increasing x-coordinate and L_y by decreasing y-coordinate. We define a grid G, by drawing a vertical line and a horizontal line through each location of P_i. Note that G is only used for describing our algorithm, but not computed explicitly.

The following main step of our algorithm will find the τ-skyline π_i. Refer to Fig. 3 for an example. We scan the sorted list L_y to find the location in P_i, denoted by p_1 (e.g., see Fig. 3), such that the sum of the probabilities of the locations of P_i that are strictly above p_1 is at most τ, but the above probability sum plus $w(p_1)$ (i.e., the probability of p_1) is larger than τ.

From the position $(-\infty, y(p_1))$, we will move a point q along the grid G. More specifically, q will move either rightwards or downwards, depending on different scenarios. The movement of q will be discretized by scanning the two sorted lists L_x and L_y. An *event* happens if either $x(q) = x(p)$ for some $p \in L_x$, or $y(q) = y(p')$ for some $p' \in L_y$. Whenever an event happens, we will decide whether q should move rightwards or downwards. If q moves rightwards, then we keep scanning L_x; otherwise we keep scanning L_y. After the algorithm stops, the path along which q moves is the τ-skyline π_i. Below, we first present the algorithm and then argue its correctness (i.e., prove that the path of q is π_i). As will be seen later, the main step of the algorithm only takes $O(k)$ time since we can process each event in $O(1)$ time and there are $O(k)$ events (since $|L_x| + |L_y| = 2k$).

For two points q_1 and q_2, we say that q_1 *strictly dominates* q_2 if $x(q_1) > x(q_2)$ and $y(q_1) > y(q_2)$. For a point p, denote by S_p the set of locations of P_i that dominate p and by S_p^+ the set of locations of P_i that strictly dominate p. For any subset P_i' of P_i, let $w(P_i') = \sum_{p \in P_i'} w(p)$. Clearly, $w(S_p) = \delta_i(p)$.

When q is moving, our algorithm will maintain the following two *invariants*: $w(S_q) > \tau$ and $w(S_q^+) \leq \tau$. Besides, the two values $w(S_q)$ and $w(S_q^+)$ will be maintained during the algorithm (they will be updated when an event happens).

We claim that both invariants hold when q is at the position $(-\infty, y(p_1))$. To see this, S_q consists of p_1 and all the locations of P_i that are strictly above p_1, and S_q^+ consists of all the locations of P_i that are strictly above p_1. It follows from our definition of p_1 that $w(S_q) > \tau$ and $w(S_q^+) \leq \tau$. See Fig. 3 for an example.

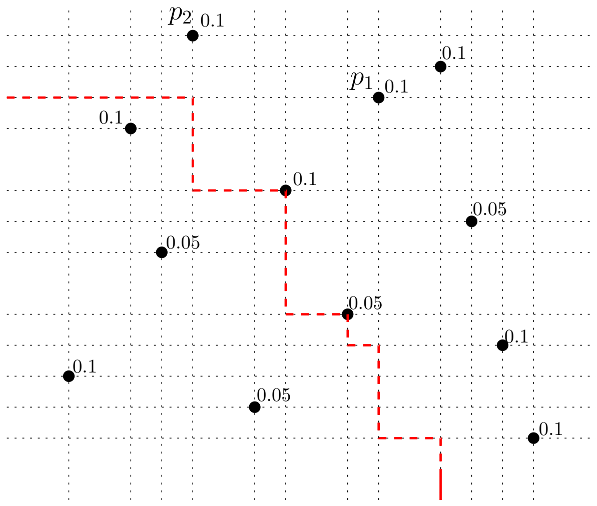

Figure 3: Illustrating our algorithm for computing π_i. One uncertain point P_i consists of 12 locations with corresponding probabilities labeled. For $\tau = 0.27$, the τ-skyline π_i of P_i is shown with (red) dashed line segments.

Rightwards movements. We move q rightwards, by scanning the sorted list L_x until an event happens, i.e., $x(q) = x(p)$ for some $p \in L_x$. We refer to the position $(x(p), y(p_1))$ as the *event point*. Depending on different scenarios, our algorithm takes different actions and updates the two values $w(S_q)$ and $w(S_q^+)$ accordingly. For any action our algorithm takes, we will argue that the two invariants hold. Based on the values of $y(p)$ and $y(q)$, we classify three cases.

1. If $y(p) < y(q)$, then p does not dominate q when q is at the event point. Hence, both S_q and S_q^+ remain the same as before, implying that both invariants hold. No action is needed in this case.

   We continue moving q rightwards. Note that before the next event happens, both S_q and S_q^+ continue remaining the same, and both invariants hold.

2. If $y(p) > y(q)$, then p strictly dominates q before the event. And when q arrives at the event point, p does not strictly dominates q anymore but p still dominates q. Hence, for processing the event, we remove p from S_q^+ and decrease $w(S_q^+)$ by $w(p)$, but S_q does not change. Clearly, both invariants hold when q is at the event point.

   Note that since $y(p_1) = y(q)$ and $y(p) > y(q)$, p_1 is not p. Based on whether $w(S_q^+) + w(p_1) \leq \tau$, we further classify two cases.

   (a) If $w(S_q^+) + w(p_1) > \tau$, then we continue moving q rightwards ($x(q) > x(p)$). Consider any point q after the current event point and before the next event point. Note that the point p does not dominate q any more. Hence, we remove p from S_q and decrease $w(S_q)$ by $w(p)$.

We claim that both invariants hold. Note that S_q^+ does not change, hence $w(S_q^+) \leq \tau$. On the other hand, according to the definition of q, no location of P_i has the same y-coordinate as q. Hence, $S_q = S_q^+ \cup \{p_1\}$. Due to $w(S_q^+) + w(p_1) > \tau$, we have $w(S_q) > \tau$. Therefore, both invariants hold.

(b) If $w(S_q^+) + w(p_1) \leq \tau$, then for later reference we use p_2 to refer to p (e.g., see Fig. 3).

Now we move q downwards, by scanning the sorted list L_y ($y(q) < y(p_1)$). Consider any point q after the current event point and before the next event point. Since $y(q) < y(p_1)$, p_1 now strictly dominates q, we add p_1 to S_q^+ and increase $w(S_q^+)$ by $w(p_1)$. Since $w(S_q^+) + w(p_1) \leq \tau$ for the previous $w(S_q^+)$, the second invariant $w(S_q^+) \leq \tau$ holds for the new S_q^+. On the other hand, the invariant $w(S_q) > \tau$ still holds since S_q does not change. Therefore, both invariants hold.

3. If $y(p) = y(q)$, then $y(p) = y(p_1)$ since $y(q) = y(p_1)$. Due to our general position assumption that no two locations of P_i have the same x- or y-coordinate, p_1 must be p. In other words, $p_1 = p$ is the event point. Clearly, the two sets S_q and S_q^+ do not change when q arrives at p_1. Therefore, both invariants hold when q is at the event point.

   Next we move q downwards, by scanning the sorted list L_y. For later reference, we use p_2 to refer to p. Consider any point q after the current event point and before the next event point. It is no hard to see that the the two sets S_q and S_q do not change. Therefore, both invariants still hold.

According to the above discussion, after an event point during rightwards movements, q will either keep moving rightwards or change the moving direction and move downwards. In the former case, we process the rightwards movement in the same way as described above. In the latter case, we have used p_2 to refer to p, with $x(q) = x(p_2)$ and $y(p_2) > y(q)$, and q moves downwards by scanning the sorted list L_y until the next event happens, i.e., $y(q) = y(p')$ for some $p' \in L_y$. In the sequel, we discuss our algorithm for processing downwards movements, which is slightly different from that for rightwards movements.

Now the position $(x(p_2), y(p'))$ is the event point.

Downwards movements. Depending on different scenarios, our algorithm takes different actions and updates $w(S_q)$ and $w(S_q^+)$ accordingly. For any action our algorithm takes, we will argue that both invariants hold. As discussed above, right before the event happens, both invariants hold and $w(S_q)$ and $w(S_q^+)$ are properly maintained.

1. If $x(p') < x(q)$, then p' does not dominate q when q is at the event point. Both S_q and S_q^+ remain the same as before, and both invariants hold when q is at the event point.

   We continue moving q downwards. Before the next event happens, both S_q and S_q^+ remain the same and both invariants hold.

2. If $x(p') > x(q)$, then before q arrives at the event point, p' does not dominates q. When q is at the event point, p' dominates q but does not strictly dominate q (since $y(q) = y(p')$). We increase $w(S_q)$ by $w(p')$, and $w(S_q^+)$ does not change. Clearly, both invariants hold. Based on whether $w(S_q^+)+w(p') \leq \tau$, we further classify two cases.

   (a) If $w(S_q^+) + w(p') \leq \tau$, then we continue moving q downwards. Consider any q after the current event point and before the next event point. The point p' now strictly dominates q. Then, we increase $w(S_q^+)$ by $w(p')$. Since $w(S_q^+) + w(p') \leq \tau$ for the previous S_q^+, we have $w(S_q^+) \leq \tau$ for the new S_q^+. On the other hand, the invariant $w(S_q) > \tau$ still holds since S_q does not change.

   (b) If $w(S_q^+) + w(p') > \tau$, then for later reference, we use p_3 to refer to p'.

   Next we move q rightwards ($x(p_2) < x(q)$). Consider any point q after the current event point and before the next event point. Notice that p_2 does not dominate q any more. Therefore, we decrease $w(S_q)$ by $w(p_2)$. However, S_q^+ is the same as before and we do not need to change $w(S_q^+)$. We claim that both invariants hold. Indeed, according to our definition, it is not difficult to see that $S_q = S_q^+ \cup \{p'\}$. Hence, $w(S_q) = w(S_q^+) + w(p') > \tau$. On the other hand, since S_q^+ does not change, $w(S_q^+) \leq \tau$ still holds. Therefore, both invariants hold.

3. If $x(p') = x(q)$, then $x(p_2) = x(p')$ since $x(q) = x(p_2)$. We argue that this case cannot happen. As $y(p') > y(p_2)$ and both p_2 and p' are in P_i, it implies that p_2 and p' are two different locations in P_i and have the same x-coordinate. This contradicts with our general position assumption that no two locations of P_i have the same x- or y-coordinate.

According to the above discussion, after an event point during downwards movements, q will either keep moving downwards or change the direction and move rightwards. In the former case, we process the next event in the same way as for downwards movements. In the latter case, we process the next event in the same way as for rightwards movements. The algorithm stops

Algorithm 1: Compute the τ-skyline π_i of P_i

```
scan $L_y$ to find the point $p_1$;
compute $w(S_q)$ and $w(S_q^+)$;
move $q$ rightwards and set $j = 1$;
while $L_x \neq \emptyset$ and $L_y \neq \emptyset$ do
    if $q$ is moving rightwards then
        scan $L_x$ for the next event, i.e., find a location $p \in L_x$ with $x(q) = x(p)$;
        if $y(p) < y(q)$ then
            move $q$ rightwards;
        if $y(p) > y(q)$ then
            $w(S_q^+) = w(S_q^+) - w(p)$;
            if $w(S_q^+) + w(p_j) > \tau$ then
                move $q$ rightwards;
                $w(S_q) = w(S_q) - w(p)$;
            else
                move $q$ downwards;
                $w(S_q^+) = w(S_q^+) + w(p_j)$;
                $j = j + 1$ and set $p_j = p$;
        if $y(p) = y(q)$ then
            move $q$ downwards;
            $j = j + 1$ and set $p_j = p$;
    if $q$ is moving downwards then
        scan $L_y$ for the next event, i.e., find a location $p' \in L_y$ with $y(q) = y(p')$;
        if $x(p') < x(q)$ then
            move $q$ downwards;
        if $x(p') > x(q)$ then
            $w(S_q) = w(S_q) + w(p')$;
            if $w(S_q^+) + w(p') \leq \tau$ then
                move $q$ downwards;
                $w(S_q^+) = w(S_q^+) + w(p')$;
            else
                move $q$ rightwards;
                $w(S_q) = w(S_q) - w(p_j)$;
                $j = j + 1$ and set $p_j = p'$;
move $q$ to infinity along the last moving direction;
return the path of $q$ as $\pi_i$;
```

when either L_x or L_y becomes empty (q will go to infinity along the last moving direction). The pseudocode in Algorithm 1 summarizes the main step of our algorithm.

Let $\tilde{\pi}_i$ denote the path of q above. Notice that during the algorithm, q moves either rightwards or downwards, and its x- (resp. y-) coordinate is non-decreasing (resp. non-increasing). Hence, $\tilde{\pi}_i$ is an xy-monotone path. Since q makes turns only if an event happens and there are $O(k)$ events ($|L_x| + |L_y| = 2k$), the number of turns of $\tilde{\pi}_i$ is $O(k)$. Hence $\tilde{\pi}_i$ has $O(k)$ complexity. To analyze the time complexity of our algorithm, after L_x and L_y are obtained in $O(k \log k)$ time in the preprocessing step, the main step of our algorithm takes $O(k)$ time since each event is processed in $O(1)$ time.

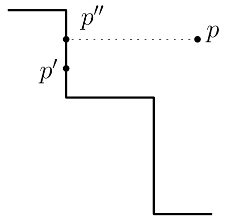

Figure 4: Illustrating the second case of the proof for Lemma 1: the solid path is $\tilde{\pi}_i$.

Correctness. Next we prove that $\tilde{\pi}_i$ is the τ-skyline π_i of P_i. To this end, we prove the following lemma, which essentially shows that the τ-skyline region R_i of P_i is strictly above and to the right of $\tilde{\pi}_i$.

Lemma 1 *For any point p, if p is strictly above and to the right of the path $\tilde{\pi}_i$, then $\delta_i(p) \leq \tau$; otherwise $\delta_i(p) > \tau$.*

Proof. Consider any point p. Below, we classify two cases, depending on whether p is strictly above and to the right of $\tilde{\pi}_i$. Our proof relies heavily on the two invariants: for any point $q \in \tilde{\pi}_i$, $w(S_q) > \tau$ (i.e., $\delta_i(q) > \tau$) and $w(S_q^+) \leq \tau$.

1. p is not strictly above or to the right of $\tilde{\pi}_i$, i.e., $p \in \tilde{\pi}_i$ or p is to the left of/below $\tilde{\pi}_i$. We show that $\delta_i(p) > \tau$.

   If $p \in \tilde{\pi}_i$, by the first invariant, $\delta_i(p) > \tau$.

   If p is to the left of/below $\tilde{\pi}_i$, we shoot a ray from p to the right and let p' be the first point on $\tilde{\pi}_i$ that is hit by the ray. Clearly, p' dominates p. By the first invariant, $\delta_i(p') > \tau$. Since p' dominates p, the locations of P_i that dominate p' must also dominate p, i.e., $S_{p'} \subseteq S_p$ and $w(S_{p'}) \leq w(S_p)$. Note that $\delta_i(p) = w(S_p)$ and $\delta_i(p') = w(S_{p'})$. Consequently, due to $\delta_i(p') > \tau$, we obtain $\delta_i(p) > \tau$.

2. p is strictly above and to the right of $\tilde{\pi}_i$. We show that $\delta_i(p) \leq \tau$.

   Suppose we shoot a ray from p to the left and let p'' be the first point on $\tilde{\pi}_i$ that is hit by the ray. Since p is strictly to the right of $\tilde{\pi}_i$, $p \neq p''$ and $x(p'') < x(p)$. According to the definition of p'', after the point q passes p'' in the algorithm, q must move downwards. In other words, there must be a point p' strictly below p'' and on the vertical line through p'' such that the vertical line segment $\overline{p'p''}$ is on $\tilde{\pi}_i$ (e.g., see Fig. 4).

   Since $p' \in \tilde{\pi}_i$, by the second invariant, $w(S_{p'}^+) \leq \tau$.

We claim that $S_p \subseteq S_{p'}^+$. Indeed, consider any point $p^* \in S_p$. It is sufficient to show that p^* strictly dominates p', i.e., $x(p^*) > x(p')$ and $y(p^*) > y(p')$. On one hand, since $x(p) > x(p'')$, $x(p^*) \geq x(p) > x(p'') = x(p')$. On the other hand, since p' is strictly below p'', $y(p'') > y(p')$. Therefore, $y(p^*) \geq y(p) = y(p'') > y(p')$. The above proves that $x(p^*) > x(p')$ and $y(p^*) > y(p')$. Hence, $S_p \subseteq S_{p'}^+$.

It follows from $w(S_{p'}^+) \leq \tau$ and $S_p \subseteq S_{p'}^+$ that $w(S_p) \leq w(S_{p'}^+) \leq \tau$ and $\delta_i(p) = w(S_p) \leq \tau$.

This concludes the proof of the lemma. □

We obtain the following theorem.

Theorem 2 *For any uncertain point P_i defined over k locations, the τ-skyline π_i of P_i has $O(k)$ complexity and can be computed in $O(k \log k)$ time. Moreover, π_i can be computed in $O(k)$ time if the locations of P_i are given as two sorted lists by their x-coordinates and y-coordinates, respectively.*

Remarks. Note that no point of the τ-skyline π_i of P_i belongs to the τ-skyline region R_i of P_i. However, if a point is not allowed to dominate itself, as in [1, 4], then it is not difficult to see that if a turning point q of π_i co-locates with a location of P_i, then q belongs to R_i.

We are ready to compute the τ-skyline π of $\mathcal{P}$. An observation is that the τ-skyline region R of $\mathcal{P}$ is the common intersection of the τ-skyline regions $R_1, \ldots, R_n$, i.e., $R = \bigcap_{i=1}^{n} R_i$. Moreover, the τ-skyline π is the upper envelope of the τ-skylines $\pi_1, \ldots, \pi_n$. Equivalently, π is the (conventional) skyline of the turning points of all the π_i's, implying that π is a xy-monotone path. By Theorem 2, each π_i has $O(k)$ turning points and can be computed in $O(k \log k)$ time. Hence all the π_i's have $O(m)$ turning points and can be computed in total $O(m \log k)$ time, implying that π has $O(m)$ complexity and can be further computed in $O(m \log m)$ time [7], where $m = nk$. We conclude the following.

Theorem 3 *Given a set $\mathcal{P}$ of n uncertain points, each of which is defined over k locations, the τ-skyline π of $\mathcal{P}$ can be computed in $O(m \log m)$ time. Further, π is a xy-monotone path of size $O(m)$, where $m = nk$.*

Finally, if one is interested in returning a subset of good candidates as skyline (uncertain) points, as in [1, 4], we can compute all the τ-skyline probabilities and use a threshold to select. After obtaining the τ-skyline of $\mathcal{P}$, one can easily compute, in total $O(m \log m)$ time, that for each P_i of $\mathcal{P}$, the probability of P_i lying inside the τ-skyline of $\mathcal{P}$. Therefore, we obtain the following.

Theorem 4 *Given a set $\mathcal{P}$ of n uncertain points, each of which is defined over k locations, all the τ-skyline probabilities can be computed in total $O(m \log m)$ time, where $m = nk$.*

References

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

[2] P.K. Agarwal, B. Aronov, S. Har-Peled, J.M. Phillips, K. Yi, and W. Zhang. Nearest neighbor searching under uncertainty II. In *Proc. of the 32nd Symposium on Principles of Database Systems*, pages 115–126, 2013.

[3] P.K. Agarwal, S.-W. Cheng, Y. Tao, and K. Yi. Indexing uncertain data. In *Proc. of the 28th Symposium on Principles of Database Systems*, pages 137–146, 2009.

[4] M.J. Atallah and Y. Qi. Computing all skyline probabilities for uncertain data. In *Proc. of the 28th Symposium on Principles of Database Systems*, pages 279–287, 2011.

[5] P. Kamousi, T.M. Chan, and S. Suri. Closest pair and the post office problem for stochastic points. In *Proc. of the 12nd Workshop on Algorithms and Data Structures*, pages 548–559, 2011.

[6] P. Kamousi, T.M. Chan, and S. Suri. Stochastic minimum spanning trees in Euclidean spaces. In *Proc. of the 27th Annual Symposium on Computational Geometry*, pages 65–74, 2011.

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

[8] X. Lian and L. Chen. Monochromatic and bichromatic reverse skyline search over uncertain databases. In *Proc. of the 2008 ACM SIGMOD International Conference on Management of Data*, pages 213–226, 2008.

[9] D. Papadias, Y. Tao, G. Fu, and B. Seeger. Progressive skyline computation in database systems. *ACM Trans. Database Syst.*, 30(1):41–82, 2005.

[10] J. Pei, B. Jiang, X. Lin, and Y. Yuan. Probabilistic skylines on uncertain data. In *Proc. of the 33rd International Conference on Very Large Data Dases*, pages 15–26, 2007.

[11] S. Suri, K. Verbeek, and H. Yıldız. On the most likely convex hull of uncertain points. In *Proc. of the 21st Annual European Symposium on Algorithms*, pages 791–802, 2013.

[12] Y. Tao, X. Xiao, and R. Cheng. Range search on multidimensional uncertain data. *ACM Transactions on Database Systems*, 32, 2007.

[13] M.L. Yiu, N. Mamoulis, X. Dai, Y. Tao, and M. Vaitis. Efficient evaluation of probabilistic advanced spatial queries on existentially uncertain data. *IEEE Transactions on Knowledge and Data Engineering*, 21:108–122, 2009.

[14] W. Zhang, X. Lin, Y. Zhang, W. Wang, G. Zhu, and J.X. Yu. Probabilistic skyline operator over sliding windows. *Information Systems*, 38:1212–1233, 2013.