Algorithms for Range-Skyline Queries

Saladi Rahul
Dept. of Computer Science & Engg.
Univ. of Minnesota–Twin Cities
4-192 Keller Hall, 200 Union St. S.E.
Minneapolis, MN 55455, USA
sala0198@umn.edu

Ravi Janardan
Dept. of Computer Science & Engg.
Univ. of Minnesota–Twin Cities
4-192 Keller Hall, 200 Union St. S.E.
Minneapolis, MN 55455, USA
janardan@umn.edu

ABSTRACT

Let S be a set of n points in $\mathbb{R}^d$, where each point has $t \geq 1$ real-valued attributes called features. A range-skyline query on S takes as input a query box $q \in \mathbb{R}^d$ and returns the skyline of the points of $q \cap S$, computed w.r.t. their features (not their coordinates in $\mathbb{R}^d$). Efficient algorithms are given for computing range-skylines and a related hardness result is established.

Categories and Subject Descriptors

F.2.2 [**Nonnumerical Algorithms and Problems**]: Geometrical problems and computations; H.2.8 [**Database Applications**]: Spatial databases and GIS

General Terms

Algorithms

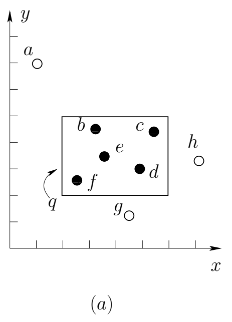
Keywords

Query processing, skyline, range search, geometric transformation, computational hardness, spatial decomposition

1. INTRODUCTION

Consider the following scenario: We are given a real-estate database that contains information on several thousand homes for sale in a large metropolitan area. Each house has a number of attributes associated with it: its location (e.g., x- and y-coordinates or lat-long), price, real-estate tax, age, distance to the high school, crime rate, etc. A prospective buyer would like to narrow her search by focusing on houses in a small neighborhood of interest (specified as a query region in the xy-plane). Among these houses, any house that has higher values for each of the remaining attributes (price, tax, age, etc.) than some other house in the region clearly need not be considered further. (We say that the former house is "dominated" by the latter.) Thus, the response to the buyer's query should be a list of houses lying in the query region such that no house in the list is dominated by another house in the query region. This list is called the "skyline" of the houses in the query region; it consists of the houses that are worth further investigation by the buyer. The advantage of working with the skyline is that its size is typically much smaller than the number of houses lying in the query region and, moreover, it contains no redundant information from the buyer's perspective (i.e., no dominated house); thus, it makes the buyer's search much easier.

Figure 1 illustrates the above discussion. In (a), a set of 8 houses in the xy-plane is shown. The points b, c, d, e, f (shown filled) lie in the query rectangle, q. The skyline of these points is to be computed w.r.t. the 'Age' and 'Price' information shown in the table. In (b), these skyline points, b, d, f, are shown circled.



	Price	Age
a	200	5
b	200	10
c	325	8
d	150	12
e	225	11
f	300	2
g	250	9
h	350	6

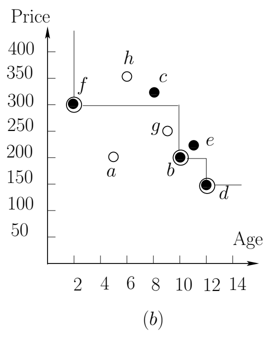


Figure 1: Skyline of points (i.e., houses) in q computed w.r.t. 'Age' (in years) and 'Price' ($\times$ 100K).

Problem statement: Let S be a set of n points in d-dimensional space, $\mathbb{R}^d$, which we call the *range space*. Each point $p \in S$ has d real-valued attributes $x_1(p), \ldots, x_d(p)$ that we call *range attributes*, i.e., its coordinates in range space. Furthermore, p has t additional real-valued attributes, called *features*, which we denote by $a_1(p), \ldots, a_t(p)$. We call $\mathbb{R}^t$ the *feature space*.

Throughout the paper, we will be interested in computing skylines in feature space (*not* range space). Accordingly, for points $p, r \in S$, we say that p *dominates* r iff $a_i(p) \leq a_i(r)$, $1 \leq i \leq t$. For simplicity, we assume that no two points have identical coordinates in all t dimensions of feature space; thus, implicitly, at least one of the inequalities above is strict. The extension to the more general case is straightforward but tedious. The *skyline* of S consists of those points of S that are not dominated by any other point of S.

The problem we wish to solve is the following: *Preprocess S into a suitable data structure so that for any axes-parallel query box $q = \prod_{i=1}^{d}[u_i, v_i]$ in range space, the skyline of the set $q \cap S$, computed in feature space, can be reported efficiently.* We call this skyline the *range-skyline* of S w.r.t. q and denote the set of range-skyline points by $RangeSky(S, q)$.

Previous work: The work in the skyline literature closest to our work are algorithms for computing skylines of points that lie within a rectangular query range [1, 5]. Note that here the range restrictions and skyline are both defined on the same set of attributes, whereas in the range-skyline problem that we consider, they are defined on different sets of attributes (range attributes versus features). The method in [5] can be suitably adapted to answer our range-skyline query, but this approach provides no theoretical worst-case guarantees.

Flow of the paper: In Section 2, we give an algorithm for range-skyline queries on points with $t \geq 1$ features in 1-dimensional range space $\mathbb{R}^1$ (i.e., $d = 1$). In Section 3, we solve the range-skyline query problem for points with $t \geq 1$ features in range space $\mathbb{R}^d$, $d > 1$. Finally, in Section 4, we consider a closely-related problem, called *range-skyline counting* where the goal is to count the number of points in the range-skyline, rather than report them. We establish the computational difficulty of this problem for $d \geq 2$ and $t \geq 3$ by showing a linear-time reduction from the so-called *set intersection* problem which is conjectured to be difficult.

2. RANGE-SKYLINE IN 1-DIMENSIONAL RANGE SPACE

In this section, we take S to be a set of n points on the real line $\mathbb{R}^1$. Thus, a point p in S has $d = 1$ range attribute, $x_1(p)$, and $t \geq 1$ features, $a_1(p), \ldots, a_t(p)$. To simplify notation, throughout this section we will view the range attribute $x_1(p)$ to be equivalent to p and will generally write "p" instead of "$x_1(p)$". The query is an interval $q = [u_1, v_1]$ in range space. As we shall see, our solution takes advantage of the fact that there is a linear ordering of the points in the range space and $q \cap S$ is a set of points that are contiguous in this ordering. Therefore, our approach works in a more general setting, where the points of S lie on a curve in $\mathbb{R}^d$ ($d \geq 1$) and the query, q, is a subsegment of the curve; the curve is our range space, the ordering of the points along the curve is the desired linear ordering, and $q \cap S$ is a subset of contiguous points in this ordering. An instance of this setting, for $d = 2$, is where many houses are situated along a freeway and the buyer is interested in houses lying between two points on the freeway. However, for simplicity, we will describe our solution assuming that the range space is $\mathbb{R}^1$.

Key ideas: Assume that the points of S are given in increasing order along $\mathbb{R}^1$ as $p_1, p_2, \ldots, p_n$, and that there are two dummy points $p_0 = -\infty$ and $p_{n+1} = \infty$ for which all feature coordinates are set to $-\infty$. For each point $p_i \in S$, we define two other points, p_i^- and p_i^+, as follows: p_i^- is the point of $S \cup \{p_0, p_{n+1}\}$ with the largest index less than i that dominates p_i in the feature space. Symmetrically, p_i^+ is the point with the smallest index greater than i that dominates p_i in the feature space. Clearly, due to how we defined the dummy points, the points p_i^- and p_i^+ always exist.

The crucial observation is that $p_i \in RangeSky(S, q)$ iff the following conditions are met: (a) $p_i \in q$, (b) $p_i^- \notin q$, and (c) $p_i^+ \notin q$. Condition (a) is clear from the definition of $RangeSky(S, q)$. Conditions (b) and (c) follow from the definition of p_i^- and p_i^+.

Thus, $p_i \in RangeSky(S, q)$ iff $u_1 \leq p_i \leq v_1$ and $p_i^- < u_1$ and $p_i^+ > v_1$, i.e., iff $u_1 \in (p_i^-, p_i]$ and $v_1 \in [p_i, p_i^+)$, i.e., iff $(u_1, v_1) \in (p_i^-, p_i] \times [p_i, p_i^+)$.

Thus, in $\mathbb{R}^2$, the point $q' = (u_1, v_1)$, defined by the endpoints of the query interval $q = [u_1, v_1]$, must lie in the axes-parallel rectangle $R_i = (p_i^-, p_i] \times [p_i, p_i^+)$ defined by p_i, p_i^-, and p_i^+. Thus, our problem reduces to determining all rectangles R_i, corresponding to points $p_i \in S$, that are stabbed by the query point $q' = (u_1, v_1)$. This is a standard intersection searching problem on n rectangles and can be solved by employing a data structure due to Chazelle [2] that uses $O(n)$ space and has a query time of $O(k + \log n)$, where k is the number of rectangles stabbed (hence the number of points in $RangeSky(S, q)$). Complete details will appear in the full paper.

THEOREM 2.1. *Let S be a set of n points on the real line $\mathbb{R}^1$, where each point has t features. S can be preprocessed in $O(n \log^t n)$ time into a data structure of size $O(n)$ such that for any query interval $q = [u_1, v_1]$, the skyline of the points of $q \cap S$, computed w.r.t. their features, can be reported in time $O(k + \log n)$, where k is the size of the skyline. This is optimal, w.r.t. space and query time, in the decision tree model. Furthermore, the same bounds hold even if the points of S lie on a curve, C, in $\mathbb{R}^d$ ($d \geq 1$) and the query is specified as a subsegment of C.*

3. RANGE-SKYLINE QUERIES IN HIGHER-DIMENSIONAL RANGE SPACE

In this section we give an efficient algorithm to answer range-skyline queries in d-dimensional range space ($d > 1$), for $t \geq 1$ features. Our solution has the added advantage of being efficiently dynamizable, i.e., points can be inserted into and deleted from the set efficiently.

Key ideas: We take S to be a set of n points in $\mathbb{R}^d$, $d > 1$. Thus, a point p in S has $d > 1$ range attributes, $x_1(p), \ldots, x_d(p)$, and $t \geq 1$ features, $a_1(p), \ldots, a_t(p)$. The query is an axes-parallel box $q = \prod_{i=1}^{d}[u_i, v_i]$ in range space. The goal is to compute $RangeSky(S, q)$.

Here is a high-level description of our approach. Throughout this discussion, our focus will be on the points of S that lie in q. The idea is to report the range-skyline of these points using their a_t coordinates. The initial search space is the entire feature space, $\mathbb{R}^t$, and we find here the point $p \in q \cap S$ with the least a_t coordinate; p is guaranteed to be in the range-skyline. In searching for the next range-skyline point of $q \cap S$, we can ignore a certain subset of $\mathbb{R}^t$, determined by p, that cannot possibly contain that range-skyline point. This subset will consist of the part of $\mathbb{R}^t$ that lies below p, w.r.t. the a_t coordinate, or is in the first octant of p. (A more formal description will be given below.) As we will see, the remainder of $\mathbb{R}^t$ can be partitioned into $t-1$ disjoint, axes-parallel boxes. We search these boxes in a certain order (to be specified later) to find the next range-skyline point. The box from which the reported point came can then be partitioned similarly and the search is continued as above. The search stops when no box containing points of $q \cap S$ remains.

More formally, among the points of $q \cap S$, let p be the point with the smallest a_t coordinate. If there is a tie between two or more points, then we consider their $a_{t-1}, \ldots, a_1$ values in turn to break the tie, favoring smaller coordinate values over larger. (This will succeed because, by assumption, no two points have identical coordinates in all t dimensions of feature space.) We report p as the first range-skyline point.

Next, we define the subset of feature space that can be ignored when searching for the second range-skyline point of $q \cap S$. Towards that end, let $w = (a_1, \ldots, a_t)$ be a generic point in the feature space $\mathbb{R}^t$. Let

$$I_p = \{w \mid a_t \leq a_t(p)\} \cup \{w \mid (a_1 \geq a_1(p)) \wedge \cdots \wedge (a_t \geq a_t(p))\}.$$

Here the first term in the union is the closed halfspace of the feature space below p (w.r.t. the a_t coordinate) and the second term is the first octant of p in feature space. Our choice of p, as described above, ensures that any other point $r \in q \cap S$ that lies in I_p is dominated by p. Thus, the set I_p can be ignored in our search for the second range-skyline point of $q \cap S$ and we focus our search on its complement $\bar{I}_p$, defined as follows.

$$\begin{aligned} \bar{I}_p &= \{w \mid a_t > a_t(p)\} \cap \{w \mid (a_1 < a_1(p)) \vee \cdots \vee \\ &\qquad (a_t < a_t(p))\} \\ &= \{w \mid a_t > a_t(p)\} \cap \{w \mid (a_1 < a_1(p)) \vee \cdots \vee \\ &\qquad (a_{t-1} < a_{t-1}(p))\} \\ &= H \cap (A_1 \cup \cdots \cup A_{t-1}), \end{aligned}$$

where $H = \{w \mid a_t > a_t(p)\}$ and $A_i = \{w \mid a_i < a_i(p)\}$, $1 \leq i \leq t-1$.

It is easy to see that

$$\begin{aligned} A_1 \cup \cdots \cup A_{t-1} &= \{w \mid a_1 < a_1(p)\} \cup \\ &\quad \{w \mid (a_1 \geq a_1(p)) \wedge (a_2 < a_2(p)\} \cup \cdots \cup \\ &\quad \{w \mid (a_1 \geq a_1(p)) \wedge \cdots \wedge \\ &\qquad (a_{t-2} \geq a_{t-2}(p)) \wedge \\ &\qquad (a_{t-1} < a_{t-1}(p))\} \\ &= B_1 \cup \cdots \cup B_{t-1}, \end{aligned} \tag{1}$$

where $B_i = \{w \mid (a_1 \geq a_1(p)) \wedge \cdots \wedge (a_{i-1} \geq a_{i-1}(p)) \wedge (a_i < a_i(p))\}$, $1 \leq i \leq t-1$.

Clearly $B_1 \cup \cdots \cup B_{t-1}$ is a union of $t-1$ disjoint, axes-parallel boxes. Therefore, so is $\bar{I}_p = (H \cap B_1) \cup \cdots \cup (H \cap B_{t-1})$, where, for $1 \leq i \leq t-1$,

$$\begin{aligned} H \cap B_i = B'_i = \\ \{w \mid a_t > a_t(p)\} \cap \{w \mid (a_1 \geq a_1(p)) \wedge \cdots \wedge \\ (a_{i-1} \geq a_{i-1}(p)) \wedge (a_i < a_i(p))\}. \end{aligned} \tag{2}$$

Figure 2 illustrates the partition of $\bar{I}_p$.

Searching the boxes: Thus, once the first range-skyline point, p, is found, the search space can be partitioned into $t-1$ disjoint, axes-parallel boxes $B'_i = H \cap B_i$. We now show that there is a certain order in which the boxes B'_i can be searched for subsequent range-skyline points of $q \cap S$ that makes the search simple and efficient.

We argue that the second range-skyline point of $q \cap S$ must belong to B'_1, unless B'_1 contains no points of $q \cap S$. To see this, consider the point in B'_1 with the smallest a_t coordinate (with ties broken as described earlier). This point cannot be dominated by any other point in B'_1. Moreover, it cannot be dominated by any point in a B'_j, $j > 1$, either. This follows from the fact that for any point $w \in B'_j$, we have $a_1 \geq a_1(p)$, whereas for any point $w \in B'_1$, we have $a_1 < a_1(p)$. Thus, the point in B'_1 with the smallest a_t coordinate is the second range-skyline point. The search for the third range-skyline point proceeds recursively in the descendant boxes of B'_1. If B'_1 contains no point of $q \cap S$, then the search proceeds to B'_2 and the point in B'_2 with the smallest a_t coordinate is the second range-skyline point. And so on.

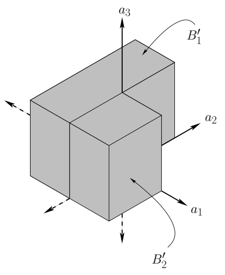


Figure 2: Partition of $\bar{I}_p$, in 3-dimensional feature space, into two disjoint, axes-parallel boxes B'_1 and B'_2. Point p (not shown) is the origin of the coordinate system.

Generalizing the above discussion, the search for the next range-skyline point will always favor searching a box B'_i and, recursively, its descendants, before searching a sibling B'_j of B'_i, $j > i$.

The data structure: The points of S are organized in a δ-dimensional range tree T, where $\delta = d + t$. The first d levels of T are built on the range attributes, $x_1, \ldots, x_d$, and the next t levels are built on the features $a_1, \ldots, a_t$. At the last level (i.e., level δ), each node v stores a field, $min(v)$, which contains the point with the minimum a_t value among those points of S stored in v's subtree. The structure T occupies $O(n \log^{\delta-1} n)$ space and can be constructed in $O(n \log^{\delta-1} n)$ [4, Ch. 5].

The query algorithm: We initialize a stack, Z, to contain the t-dimensional range $(-\infty, \infty) \times \cdots \times (-\infty, \infty)$ in feature space. At any point in the execution of the algorithm, the stack contains zero or more t-dimensional ranges representing boxes to be searched for range-skyline points. While Z is non-empty, we pop the range, R, at the top of Z and search in T with the range $q \times R$. This identifies a set, C, of $O(\log^\delta n)$ canonical nodes in T. (See [4, Ch. 5] for more details on this approach.) Among the points in $\{min(v) \mid v \in C\}$, we report the one with the smallest a_t value as the next range-skyline point p. We then form the boxes $B'_1, \ldots, B'_{t-1}$ and push these onto Z, in the order $B'_{t-1}, \ldots, B'_1$, so that B'_1 is on the top of the stack. (In forming the B'_i, care must be taken to ensure that they are t-dimensional. For instance, in Figure 2, $B'_2 = H \cap B_2$ is represented as the 3-dimensional range $[a_1(p), \infty) \times (-\infty, a_2(p)) \times (a_3(p), \infty)$.) The query algorithm terminates when Z becomes empty.

The query time is $O((k+1) \log^\delta n)$. Updates can also be efficiently performed on it without affecting the space and the query bounds. Details will appear in the full paper.

Theorem 3.1. *Let S be a set of n points in d-dimensional range space, $\mathbb{R}^d$, where each point has t features. S can be preprocessed into a data structure of size $O(n \log^{\delta-1} n)$ so*

Algorithm 1: Range-Skyline-Query(T, q)

```
Input: A $(d+t)$-dimensional range tree, $T$, storing a
       set, $S$, of $n$ points, each with $d$ range attributes
       and $t$ features, and a $d$-dimensional,
       axes-parallel query box $q$.
Output: The set, $RangeSky(S,q)$ of range-skyline
        points of $q \cap S$.
begin
    Initialize a stack, $Z$, to contain the $t$-dimensional
    range $(-\infty,\infty) \times \cdots \times (-\infty,\infty)$
    while $Z \neq \emptyset$ do
        $R \longleftarrow Pop(Z)$
        Query $T$ with $q \times R$ and find the point, $p$, in $T$
        with minimum $a_t$ value (as described in the text)
        if $p$ exists then
            Report $p$ as a range-skyline point
            Compute the ranges $B'_1, \ldots, B'_{t-1}$ (Eq. 2)
            Push $B'_{t-1}, \ldots, B'_1$ (in that order) onto $Z$
                                // $B'_1$ is on the top of $Z$
```

that given any axes-parallel, d-dimensional query box $q = \prod_{i=1}^{d}[u_i, v_i]$*, the skyline of the points of* $q \cap S$*, computed w.r.t. their features, can be reported in time* $O((k+1)\log^{\delta} n)$*, where* $\delta = d + t$ *and* $k \geq 1$ *is the size of the skyline. The data structure can be built in* $O(n \log^{\delta-1} n)$ *time and can be extended to support updates in* $O(\log^{\delta} n)$ *amortized time.*

4. HARDNESS OF RANGE-SKYLINE COUNTING

We consider a problem that is closely related to the range-skyline query problem. This problem, called *range-skyline counting*, is to count the number of points in $RangeSky(S,q)$. We provide evidence for the likely computational difficulty of this problem by showing a linear-time reduction from the *set intersection* problem, which is conjectured to be difficult.

Set Intersection Problem: Given sets $S_1, S_2, \ldots, S_m$ of positive reals, where $\sum_{i=1}^{m}|S_i| = n$, decide if S_i and S_j are disjoint, for query indices i and j, $i < j$.

It is conjectured that this problem is "hard". Specifically, in the so-called cell-probe model without the floor function and where the maximum cardinality of the sets is polylogarithmic in m, any algorithm to answer set intersection queries in $\tilde{O}(\alpha)$ time requires $\tilde{\Omega}((n/\alpha)^2)$ space, for $1 \leq \alpha \leq n$ [3]. (The "tilde" notation is used to suppress polylogarithmic factors.)

Idea behind the reduction: Given sets $S_1, S_2, \ldots, S_m$, we map their elements to a point-set, T, with $d = 2$ range attributes and $t = 3$ features. This mapping has the property that S_i and S_j are disjoint iff $|RangeSky(T, q_{ij})| = |S_i| + |S_j|$, where q_{ij} is a certain query rectangle determined by S_i and S_j. (We define q_{ij} below and establish the stated property in Lemma 4.1.)

Specifically, let $L_1 : y = x + n$ and $L_2 : y = x - n$ be two lines in the plane (the range space). We map each element of each set S_i to two points, one on L_1 and one on L_2, as follows: Let $n_0 = 0$ and $n_i = n_{i-1} + |S_i|$, for $1 \leq i \leq m$. Let s_{ik} be the kth element of S_i. (S_i is unordered and s_{ik} is simply the kth element in an arbitrary ordering of the elements of S_i.) The mapping of s_{ik} to points on L_1 and L_2 is given as follows: $s_{ik} \mapsto s'_{ik} = (-(k + n_{i-1}), -(k + n_{i-1}) + n)$, on L_1 and $s_{ik} \mapsto s''_{ik} = ((k + n_{i-1}), (k + n_{i-1}) - n)$, on L_2.

The resulting set, T, consists of $2n$ points and the above coordinates are the range attributes of the points of T. Observe that the elements of each S_i map to sets of contiguous points on L_1 (in the second quadrant) and on L_2 (in the fourth quadrant). Moreover, it is easy to see that for any query indices i and j, where $i < j$, there is a rectangle q_{ij} that contains exactly all the points s'_{ik} that S_i maps to on L_1 and all the points s''_{jk} that S_j maps to on L_2. Specifically, q_{ij} is defined uniquely by the first and last mapped points of S_i on L_1 and the first and last mapped points of S_j on L_2. By storing with each set the first and last mapped point of that set on L_1, and similarly on L_2, the rectangle q_{ij} can be computed in $O(1)$ time for any pair of query indices i and j, $i < j$.

Next, each point of T is assigned $t = 3$ features. Specifically, for each set S_i, the points s'_{ik} and s''_{ik} are both assigned the feature vector $(i, s_{ik}, -s_{ik})$. This constitutes the entire reduction and it is clear that it takes $O(n)$ time. The following lemma, whose proof is omitted due to space limitation, captures the main property of the reduction.

LEMMA 4.1. *Sets* S_i *and* S_j $(i < j)$ *are disjoint iff* $|RangeSky(T, q_{ij})| = |S_i| + |S_j|$.

Clearly, deciding if $|RangeSky(T, q_{ij})| = |S_i| + |S_j|$ is no more difficult than range-skyline counting for $d = 2$ and $t = 3$. Furthermore, the latter is no more difficult than range-skyline counting for $d > 2$ and $t > 3$. This, together with Lemma 4.1, establishes the following theorem.

THEOREM 4.1. *The range-skyline counting problem (for* $d \geq 2$ *and* $t \geq 3$*) is at least as hard the set intersection problem. Specifically, answering a range-skyline counting query in* $\tilde{O}(\alpha)$ *time requires* $\tilde{\Omega}((n/\alpha)^2)$ *space, for* $1 \leq \alpha \leq n$ *in the cell-probe model, without the floor function.*

5. REFERENCES

[1] G. S. Brodal and K. Tsakalidis. Dynamic planar range maxima queries. In *Proc. 38th Intl. Conf. on Automata, Languages, and Programming*, pages 256–267, 2011.

[2] B. Chazelle. Filtering search: A new approach to query-answering. *SIAM J. Computing*, 15(3):703–724, 1986.

[3] P. Davoodi, M. Smid, and F. van Walderveen. Two-dimensional range diameter queries. In *Proc. Latin American Theoretical Informatics Symposium*, pages 219–230, 2012.

[4] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry: Algorithms and applications*. Springer-Verlag, 2nd edition, 2000.

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