The Range Skyline Query

Theodoros Tzouramanis
University of the Aegean,
83200 Samos, Greece
ttzouram@aegean.gr

Eleftherios Tiakas
Aristotle University of Thessaloniki,
54124 Thessaloniki, Greece
tiakas@csd.auth.gr

Apostolos N. Papadopoulos
Aristotle University of Thessaloniki,
54124 Thessaloniki, Greece
papadopo@csd.auth.gr

Yannis Manolopoulos
Open University of Cyprus,
2220 Latsia, Cyprus
yannis.manolopoulos@ouc.ac.cy

ABSTRACT

The range skyline query retrieves the dynamic skyline for every individual query point in a range by generalizing the point-based dynamic skyline query. Its wide-ranging applications enable users to submit their preferences within an interval of 'ideally sought' values across every dimension, instead of being limited to submit their preference in relation to a single sought value. This paper considers the query as a hyper-rectangle iso-oriented towards the axes of the multi-dimensional space and proposes: (i) main-memory algorithmic strategies, which are simple to implement and (ii) secondary-memory pruning mechanisms for processing range skyline queries efficiently. The proposed approach is progressive and I/O optimal. A performance evaluation of the proposed technique demonstrates its robustness and practicability.

CCS CONCEPTS

•**Theory of computation** → **Theory and algorithms for application domains** → **Database theory** → Data structures and algorithms for data management.

KEYWORDS

Algorithms; progressive skyline search; index-based query processing; multi-dimensional data; experimentation.

ACM Reference format:
Theodoros Tzouramanis, Eleftherios Tiakas, Apostolos N. Papadopoulos, and Yannis Manolopoulos. 2018. The Range Skyline Query. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management, Torino, Italy, October 22-26, 2018 (CIKM 2018)*, 10 pages. https://doi.org/10.1145/3269206.3271693

1 INTRODUCTION

Its role in many real life applications, including multi-criteria decision-making, market analysis, and quantitative economics research, has led data management research to focus on the *skyline query* [2]. Given a dataset P of points in an n-dimensional (n-d) space, the skyline query returns all the points of P which are not *dominated* by another point of P. A point dominates another point if it is as good, or better, in all dimensions and strictly better in at least one dimension. Without loss of generality, smaller values are preferred, *i.e.* it is assumed that the *min* function is used to determine the *goodness* along every dimension.

Some applications require that the skyline of a dataset be dynamically generated on the basis of a user's predicate, rather than be *static*. A tourist is looking for cheap hotels close to the beach: the query point q (q_x, q_y) in Figure 1(a) might represent the preference expressed for a q_x euros priced hotel (x-axis) at a q_y meters distance from the beach (y-axis). For an 'ideal' hotel q, all interesting hotels not dominated by others in relation to (hereafter i.r.t.) q must be retrieved. A skyline analysis will provide recommendations for the best match: Every data point p corresponding to a hotel is projected onto a new space, in which the point's coordinate in every dimension equals the absolute difference between the data point p and the query point q. Figure 1(a) demonstrates the result. A *static* skyline in the projected space includes the points *e*, *i* and *l*.

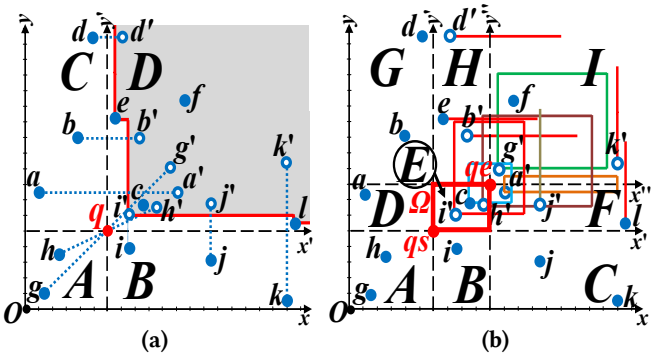


Figure 1: (a) The dynamic skyline set $SS_q(P) = \{e, i, l\}$ of a dataset P of points i.r.t. a single query point q. (b) On the computation of the range skyline of P i.r.t. a 2-d rectangle Ω.

Figure 1(a) illustrates a (*point-based*) *dynamic skyline* [16] i.r.t. the query point q. In the context of n-d databases, static and dynamic skyline queries have been examined in applications with precise [2, 4, 12], incomplete [10] and uncertain data [23], both on centralized; and parallel and distributed [5] databases. Methods for continuous skyline computation on streaming [11], moving objects [20] and other emerging applications have also been proposed.

Lin et al. [14] coined the term *range-based skyline query* for a query input which is not necessarily going to be a single point. Wang et al. [24] were the first to propose an algorithm for range-based dynamic skyline query processing i.r.t. all the infinite query points in a range but their method comes at a computational cost.

This paper focuses on a more general case, the *range skyline query*. Given an n-d dataset, the range skyline query retrieves the dynamic skyline of the dataset i.r.t. every individual point in an n-

CIKM'18, October 22-26, 2018, Torino, Italy.

ACM ISBN 978-1-4503-6014-2/18/10...$15.00
https://doi.org/10.1145/3269206.3271693

d query hyper-rectangle. The resulting set contains tuples of the form <*data object*, *sub-region*>, such that the '*data object*' is a dynamic skyline point of the dataset i.r.t. the corresponding '*sub-region*' of the given hyper-rectangle, and not necessarily i.r.t. the whole query hyper-rectangle, as was investigated by [24]. Consider a user who posts a query targeting an interval of 'ideal' values on every dimension, instead of a precise value, e.g. targeting the best hotels at between 50 and 100 euros per night and located between 200 and 500 meters from the beach. Any hotel belonging to the dynamic skyline i.r.t. any query point (and not necessarily i.r.t. all the query points) in the given 2-d range might be eligible. Consider a service-provider, offering customized fitness and health advice: in such a context, the user, asked to provide confidential information (e.g. blood pressure *etc.*), feels more comfortable submitting this personal information in an opaque manner. Submitting an interval of values, instead of precise values, in every dimension, would produce a *n*-d hyper-rectangle query pattern, rather than a point query.

This approach can also handle the *multi-source skyline query* [6] that can retrieve the points of a dataset not dominated by a set of query points, also known as the *spatial skyline query* [19], a special case of the (point-based) dynamic skyline query. Our method deals with the multi-source skyline query with a single R-tree traversal, after drawing a range that includes all the provided sources and by searching, in the results, for the data points that belong to the dynamic skyline of every one of these sources.

The proposed solution can be an alternative to the caching mechanisms (e.g. [18]) put forward for speeding up point-based dynamic skyline queries by using the information of other previously processed point-based dynamic skyline queries. Such queries, issued in relatively close time instants (e.g. in a read-heavy database setting), representing relatively close query points on the space, can be grouped and processed in batches, through issuing a single range skyline query, the range of which will be the *MBR* of all the query points that the users have submitted. The range skyline search results can then be refined in main memory to extract the results of the individual point-based dynamic skyline queries.

To our knowledge, this is the first effort to tackle (dynamic) range skyline query processing in spatial and multi-dimensional databases. The technical contributions of this paper are that:

(1) it introduces and formalizes the (dynamic) range skyline query, together with a new dominant method for evaluating multi-dimensional data using the range skyline operator.
(2) it proposes a progressive algorithm that exploits a number of interesting properties and pruning strategies to process the above query efficiently in $n \geq 2$ dimensions, and i.r.t. two forms of the predicate of the query, *i.e.* i.r.t. a line segment parallel to an axis and i.r.t. a hyper-rectangle iso-oriented towards the axes of the *n*-d space.
(3) it experimentally verifies that the proposed algorithm is I/O optimal and robust in practical terms.

Below, Section 2 provides some important definitions and notations; Section 3 develops an operational and optimal solution for range skyline computation; The CPU and I/O costs obtained experimentally are analyzed in Section 4; Section 5 discusses the advantages and limitations of related earlier work; and Section 6 summarizes and points to future directions for research.

2 FORMULATING THE PROBLEM

Let S be an n-d space and P be a dataset of points onto S. A point $p \in P$ can be represented as $p\,(p_1, p_2, ..., p_n)$, where $p_1, p_2, ..., p_n$ are its coordinate values. For the sake of simplicity we assume non-negative coordinate values.

Definition 1 – Point-based Dynamic Dominance. *Given a dataset P of points in an n-d space S and a reference point $q\,(q_1, q_2, ..., q_n) \in S$, a data point $p\,(p_1, p_2, ..., p_n) \in P$ dynamically dominates another data point $r\,(r_1, r_2, ..., r_n) \in P$ i.r.t. q, denoted as $p \prec_q r$, if and only if $\forall\ i \in \{1, 2, ..., n\}$ we have $|q_i - p_i| \leq |q_i - r_i|$ and $\exists\ j \in \{1, 2, ..., n\}$: $|q_j - p_j| < |q_j - r_j|$.*

Definition 2 – Dynamic Dominance (Generalization of Definition 1). *Given a dataset P of points in an n-d space S and a reference hyper-region $Q \in S$, a data point $p \in P$ dynamically dominates another data point $r \in P$ i.r.t. Q, denoted as $p \prec_Q r$, if and only if $\forall\ q \in Q$ we have $p \prec_q r$.*

Definition 3 – Dynamic Skyline Query. *Given a dataset P of points in an n-d space S and a reference hyper-region $Q \in S$, the* dynamic skyline query *of P i.r.t. Q retrieves the set of data points $SS_Q(P) \subseteq P$ which are not dynamically dominated by any other data point in P i.r.t. Q, that is, $SS_Q(P) = \{p \in P \mid \nexists\ r \in P: r \prec_Q p\}$. The points in $SS_Q(P)$ are called* dynamic skyline points *of P i.r.t. Q.*

Q is called the *query region*. When Q shrinks to a single point, the query represents a dynamic skyline query i.r.t. a single query point, the computation of which is studied in [16]. The dynamic skyline of P i.r.t. a query point $q\,(q_1, ..., q_n)$ can be computed as the static skyline query of P after projecting all data points $p\,(p_1, ..., p_n) \in P$ of the original space onto a new dynamic data space, in which q is the origin-point. The Euclidean distances to q on every dimension are used as mapping functions for the data points. Therefore, every data point p of the original space will be projected onto point $p'\,(|q_1 - p_1|, ..., |q_n - p_n|)$.

Figure 1(a) shows that the dynamic skyline of the dataset P i.r.t. a query point q consists of the set of points $SS_q(P) = \{e, i, l\}$. Point a, dynamically dominated by point i, is not part of $SS_q(P)$.

The computation of the dynamic skyline of P i.r.t. a region Q is much more complicated than the computation of the point-based dynamic skyline because it demands that every data point be projected i.r.t. every one of the - infinite in number - query points $q \in Q$. Figure 1(b) illustrates such an example in the 2-d space, in which Q is a rectangle Ω, i.r.t. which the user might ask for the dynamic skyline of a dataset P, *i.e.* the set $SS_\Omega(P)$ of the data points which are not dynamically dominated by any other data point in P i.r.t. every query point q in Ω.

Definition 4 – Partial Dynamic Dominance. *Given a dataset P of points in an n-d space S and a reference hyper-region $Q \in S$, a point $p \in P$ partially dynamically dominates another point $r \in P$ i.r.t. Q, denoted as $p \prec^P_Q r$, if and only if $\exists\ q \in Q$: $p \prec_q r$.*

Definition 5 – (Dynamic) Range Skyline Query. *Given a dataset P of points in an n-d space S and a reference hyper-rectangle $\Omega \in S$ iso-oriented towards the axes of S, the range skyline query of P i.r.t. Ω retrieves the dynamic skylines of P i.r.t. every reference point in Ω. The answer to the query is a set $RSS_\Omega(P)$ of tuples of the form <p, Qp>, where $p \in P$ is a data point and $Qp \in \Omega$ is a hyper-region, i.r.t. which the point p is a dynamic skyline point of P, that is, $RSS_\Omega(P)$ =*

{<*p*, *Qp*>: *Qp* ∈ Ω *and* $p \in SS_{Qp}(P)$}. *The data points in* $RSS_{\Omega}(P)$ *are called range skyline points of P i.r.t.* Ω.

The query predicate Ω can only be a hyper-rectangle iso-oriented towards the axes of the *n*-d space, since it expresses the user's preference, which consists of an interval of values on every dimension. On the basis of Figure 1(b), one tuple in $RSS_{\Omega}(P)$ is the <*e*, Ω − [(7, 5), (10, 8)]>[1], which means (see below) that $e \in SS_{\Omega - [(7, 5), (10, 8)]}(P)$, where the range [(7, 5), (10, 8)] represents a rectangle (boundaries inclusive), with bottom-left-most vertex (7, 5) and top-right-most vertex (10, 8).

3 RANGE SKYLINE SEARCH

As discussed, the computation of the dynamic skyline of a dataset *P* i.r.t. a query point *q* on the data space determines that a new projection space needs to be created, for which the point *q* will be the origin-point. The dynamic projection of every data point onto this new space is defined on the basis of the following theorem:

THEOREM 1. *For the computation of the dynamic skyline of a dataset P of points i.r.t. a reference query point q* $(q_1, q_2, ..., q_n)$, *every data point* $p\,(p_1, p_2, ..., p_n) \in P$ *needs to be projected onto a point p' with coordinate values*:

$$p'_i = \begin{cases} 2q_i - p_i, & \text{if } p_i \le q_i \\ p_i, & \text{elsewhere} \end{cases}; \quad \forall\, i \in \{1, 2, ..., n\} \qquad (1)$$

PROOF. The initial focus is on the 2-d space, with a generalization subsequently to multiple dimensions.

As Figure 1(a) shows, the new 2-d projection space *O'x'y'*, with *O'* ≡ *q*, splits the original space into four regions, *A*, *B*, *C* and *D*. Then, with regard to its original position, for the computation of the dynamic skyline of *P* i.r.t. *q*, every point $p\,(p_x, p_y) \in P$ will need to be projected onto point:

$$\begin{cases} p'(2q_x - p_x, 2q_y - p_y), & \text{if } p_x \le q_x \text{ and } p_y \le q_y \\ p'(p_x, 2q_y - p_y), & \text{if } q_x \le p_x \text{ and } p_y \le q_y \\ p'(2q_x - p_x, p_y), & \text{if } p_x \le q_x \text{ and } q_y \le p_y \\ p'(p_x, p_y), & \text{if } q_x \le p_x \text{ and } q_y \le p_y \end{cases}$$

The above set of equations shows that if $p_x \le q_x$ then $p'_x = 2q_x - p_x$, otherwise $p'_x = p_x$.[2] Also, if $p_y \le q_y$ then $p'_y = 2q_y - p_y$, otherwise $p'_y = p_y$. Intuitively, every data point that does not lie in region *D* is projected onto its symmetrical point inside *D* i.r.t. the contour of *D* (*i.e.* using the *O'x'y'* axes).

Hence, Equation (1) truly holds for every dimension of the *n*-d space. This can also be proved by mathematical induction based on the 2^d partitions into which *q* divides the original space, by considering that the coordinate value, which Equation (1) computes for every dimension, is independent from the computed coordinate values in the other dimensions. □

The algorithm for processing the dynamic skyline $SS_q(P)$ of a dataset *P* i.r.t. a query point *q* was introduced an analyzed in [16]. According to Definition 3, a data point $e \in SS_q(P)$ if $\nexists\ p \in SS_q(P)$: $p \prec_q e$. The algorithm for this dominance examination uses the projected versions of *p* and *e* i.r.t. *q*, the coordinates of which are calculated using Equation (1).

3.1 Two-dimensional Space

3.1.1 Single-dimensional Query Intervals

The basis of our approach is the case of 1-d query intervals. The user's preference are assumed to consists of a precise value on the *y*-axis and of an interval of values on the *x*-axis; the created query predicate is a line segment aligned with the *x*-axis (*idem* for a line segment aligned with the *y*-axis). On the basis of Figure 1, Figure 2(a) illustrates such a line segment *Lx* [*qs*, *qe*] with end-points *qs* (qs_x, qs_y) and *qe* (qe_x, qe_y), where $qs_x \le qe_x$, and $qs_y = qe_y$.

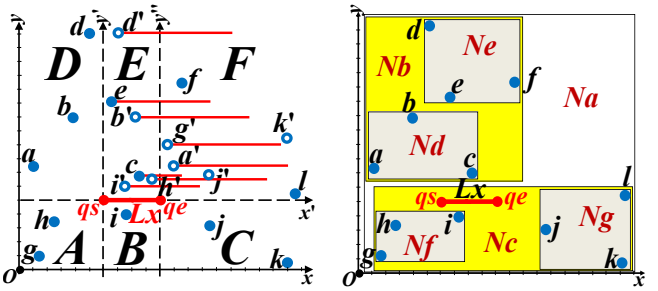


Figure 2: (a) On the computation of the (dynamic) range skyline of a dataset *P* i.r.t. a line segment *Lx* aligned with the *x*-axis. (b)The R-tree built on top of the example dataset *P*.

Figure 2(a) illustrates the division of the original space into six regions *A*, *B*, *C*, *D*, *E* and *F* on the basis of the location of *Lx* on the workspace: the following properties should be kept in mind:

PROPERTY 1. *Assuming a data point p in the region A / region D, its projected point p' i.r.t. every different query point q* ∈ *Lx will belong on a line segment aligned to the x-axis that will be double the length of Lx.*

PROPERTY 2. *Assuming a data point p in the region C / region F, its projected point p' i.r.t. every different q* ∈ *Lx will have a fixed position, independently of the position of q on Lx.*

PROPERTY 3. *Assuming a data point p* (p_x, p_y) *in the region B / region E, the projected point p' i.r.t. every different q* ∈ *Lx* [*qs*, *qe*] *will belong on a line segment which will be aligned with the x-axis and it will have a length that is equal to* $2(qe_x - p_x)$.

Due to space limitation, the proofs of the above properties are not included. Figure 2(a) illustrates the projections (as red line segments parallel to the *x*-axis) of all the points in the example dataset *P* i.r.t. every individual query point *q* ∈ *Lx*.

3.1.1.1 Main-Memory Dynamic Dominance Computation

Assuming two points *p*, *r* ∈ *P*, to find whether $p \prec^{P}_{Lx} r$, and recalling Equation (1), does ∃ *q* ∈ *Lx*: $p \prec_q r$? The answer lies in:

THEOREM 2. *Given a dataset P of points in a 2-d space S, two points* $p(p_x,p_y)$, $r(r_x,r_y) \in P$ *and a reference line segment* $Lx[qs, qe] \in S$ *aligned with the x-axis, where* $qs(qs_x,qs_y)$ *is the left-most end-point and* $qe(qe_x,qe_y)$ *is the right-most end-point of Lx, Table* 1 *specifies the range of coordinate values, per axis of the space, of all points q* ∈ *Lx i.r.t. which* $p \prec_q r$ *holds.*

[1] For the sake of simplicity, the expression "Ω − Ω1" represents the subregion of Ω, which does not include the rectangle Ω1.

[2] Evidently, if $p_x = q_x$ then the transformed version p'_x of p_x can be calculated on the basis of either the $p'_x = 2q_x - p_x$, or the $p'_x = p_x$ formulas, since they both provide the same result.

Table 1: A table specifying the range of the coordinate values on the *i*-axis of all the points $q \in Lx$ i.r.t. which $p \prec_q r$ holds.

Case id	Condition	Additional condition	Range [*min*, *max*] of the coordinates on the *i*-axis of the points $q \in Lx$ for which we have $p \prec_q r$
$p_i < r_i$			
1	$p_i < r_i \le qs_i \le qe_i$		$\varnothing$
2	$p_i \le qs_i \le r_i \le qe_i$	if $qs_i \le \frac{p_i+r_i}{2}$ then:	$[qs_i, \frac{p_i+r_i}{2}]$
		else:	$\varnothing$
3	$p_i \le qs_i \le qe_i \le r_i$	if $qs_i \le \frac{p_i+r_i}{2} \le qe_i$ then:	$[qs_i, \frac{p_i+r_i}{2}]$
		else if $qe_i \le \frac{p_i+r_i}{2}$ then:	$[qs_i, qe_i]$
		else:	$\varnothing$
4	$qs_i \le p_i \le qe_i \le r_i$	if $\frac{p_i+r_i}{2} \le qe_i$ then:	$[qs_i, \frac{p_i+r_i}{2}]$
		else:	$[qs_i, qe_i]$
5	$qs_i \le p_i < r_i \le qe_i$		$[qs_i, \frac{p_i+r_i}{2}]$
6	$qs_i \le qe_i \le p_i < r_i$		$[qs_i, qe_i]$
$r_i < p_i$			
7	$r_i < p_i \le qs_i \le qe_i$		$[qs_i, qe_i]$
8	$r_i \le qs_i \le p_i \le qe_i$	if $qs_i \le \frac{p_i+r_i}{2}$ then:	$[\frac{p_i+r_i}{2}, qe_i]$
		else:	$[qs_i, qe_i]$
9	$r_i \le qs_i \le qe_i \le p_i$	if $qe_i < \frac{p_i+r_i}{2}$ then:	$\varnothing$
		else if $qs_i \le \frac{p_i+r_i}{2} \le qe_i$ then:	$[\frac{p_i+r_i}{2}, qe_i]$
		else:	$[qs_i, qe_i]$
10	$qs_i \le r_i \le qe_i \le p_i$	if $\frac{p_i+r_i}{2} \le qe_i$ then:	$[\frac{p_i+r_i}{2}, qe_i]$
		else:	$\varnothing$
11	$qs_i \le r_i < p_i \le qe_i$		$[\frac{p_i+r_i}{2}, qe_i]$
12	$qs_i \le qe_i \le r_i < p_i$		$\varnothing$
$p_i = r_i$			
13	$p_i = r_i$		$[qs_i, qe_i]$

PROOF. The dynamic dominance investigation between the data points p (p_x, p_y) and r (r_x, r_y) i.r.t. a query point q (q_x, q_y) $\in$ Lx [qs (qs_x, qs_y), qe (qe_x, qe_y)] involves the following inequalities:

$$p \prec_q r \Leftrightarrow \left[\begin{pmatrix} p'_x < r'_x \\ p'_y \le r'_y \end{pmatrix} \quad or \quad \begin{pmatrix} p'_x \le r'_x \\ p'_y < r'_y \end{pmatrix}\right] \Leftrightarrow \left[\begin{pmatrix} p'_x <\!/\!\le r'_x \\ p'_y \le\!/\!< r'_y \end{pmatrix}\right]$$

in which "</≤" means "< or ≤" and $\forall\ i \in \{x, y\}$ the parameters p'_i and r'_i are the dynamic coordinate values on the x- and y- axes of p and r, respectively, i.r.t. q, as they are calculated using Equation (1). We shall examine the inequalities for one of the axes, which we shall name the i-axis, where $i \in \{x, y\}$, which means that, by considering the symbols "x" or "y" instead of "i", we can have the coordinates on every axis of the space.

Therefore, in relation to the i-axis, if we assume that $p_i < r_i$, then, according to Figure 2(a), six cases exist in respect of the spatial order between the points p, r, qs and qe on the workspace, as this order is projected onto their coordinate values on the i-axis, *i.e.* it holds that either $p_i < r_i \le qs_i$ or $p_i \le qs_i \le r_i \le qe_i$ or $p_i \le qs_i \le qe_i \le r_i$ or $qs_i \le p_i \le qe_i \le r_i$ or $qs_i \le p_i < r_i \le qe_i$, or $qe_i \le p_i < r_i$. If, however, we assume that $r_i < p_i$, then, according to Figure 2(a), six cases exist in relation to the spatial order between the points p, r, qs and qe, as this order is projected onto their coordinate values on the i-axis, *i.e.* it holds that either $r_i < p_i \le qs_i$ or $r_i \le qs_i \le p_i \le qe_i$ or $r_i \le qs_i \le qe_i \le p_i$ or $qs_i \le r_i \le qe_i \le p_i$ or $qs_i \le r_i < p_i \le qe_i$, or $qe_i \le r_i < p_i$. If, finally, we assume that $p_i = r_i$, then one additional case exists.

Due to space limitation, we shall prove only one of these cases.

Case 1 (*i.e.* $p_i < r_i \le qs_i$)*:* In this case, the inequality $p'_i <\!/\!\le r'_i$, according to Equation (1), becomes:

$p'_i <\!/\!\le r'_i \Leftrightarrow 2q_i - p_i <\!/\!\le 2q_i - r_i \Leftrightarrow p_i >\!/\!\ge r_i$

therefore p cannot dynamically dominate r i.r.t. the whole line segment Lx since by assumption we have $p_i < r_i$. □

Theorem 2 is about reducing the cost of examining the partial dynamic dominance relation between two data points i.r.t. a line segment, and about incurring, instead, the markedly lower cost of a search into a look-up table that can be kept into main memory. In particular, if we assume two data points $p, r \in P$, then, according to Theorem 2, if Table 1 indicates that, as concerns the coordinates on the y-axis, the data point p can dynamically dominate the data point r, it will follow that Table 1, again, will define the coordinates on the x-axis of the left-most and of the right-most end-points of the line sub-segment (if any such line sub-segment exists) $Lp \in Lx$, i.r.t. which $p \prec_{Lp} r$ holds. For example, on the basis of Figure 2(a), to determine whether there is any line sub-segment $Lp \in Lx$ i.r.t. which h (2.5, 3.5) $\prec_{Lp} g$ (1.5, 1) holds, we consult Table 1. Thus, since $g_x < h_x < qs_x$ and $g_y < h_y < qs_y$, Table 1 specifies that as concerns every axis separately, h can dynamically dominate g i.r.t. every possible coordinate value per axis that the query point $q \in Lx$ can take, therefore, finally, $h \prec_{Lx} g$ holds.

Lemma 1 follows from Theorem 2. Due to space limitation, the proofs of the lemmas are not included.

LEMMA 1: *Given a dataset P of points, two points $p, r \in P$ and a line segment Lx aligned with the x-axis, for which $p \prec_{Lx} r$ holds, then p dynamically dominates r i.r.t. only one single sub-segment $Lp \in Lx$, which also includes the left-most or/and the right-most end-points of Lx.*

3.1.1.2 *Secondary-Memory Range Skyline Computation*

The R-tree [8] will serve as the backbone spatial index to process the range skyline query. The proposed algorithm employs a branch-and-bound technique to prune the search space, as does the algorithm for the point-based dynamic skyline query [16]. Lemma 2 below speeds-up the algorithm, which will be described later.

LEMMA 2. *Given a dataset P of points, a point $p \in P$ and a line segment Lx aligned with the x-axis, if p is a range skyline point of P i.r.t. Lx then only one single sub-segment $Lp \in Lx$ exists for which $<p, Lp> \in RSS_{Lx}(P)$ i.e. only one single sub-segment $Lp \in Lx$ exists, i.r.t. which the point p is a dynamic skyline point of P.*

Algorithm 1 sets out the *pseudo*-code of the algorithm for processing the range skyline query i.r.t. a reference line segment Lx aligned with the x-axis. Based on Theorem 2 and Lemmas 1-2, it uses two priority queues H and H' with pairs of the form $<e, Le\,[qes, qee]>$, where e is an R-tree entry (a point or an *MBR*) and Le is the sub-segment of Lx, i.r.t. which e has not yet been found to be dynamically dominated by any data point in RSS_{Lx}. All entries in H have the same left-most end-point qes of their Le and are sorted in ascending order of the *MINDIST* minimum distance of e to qes.

In the beginning, the algorithm in Line 2 pushes into H all the entries e of the R-tree root in the form $<e, Lx>$. Then, in Lines 5-6 the algorithm removes the top entry $<e, Le>$ of H and, using Theo-

Algorithm 1: The Range Skyline $RSS_{Lx}(P)$
Input: The dataset P indexed using an R-tree R and the query line segment $Lx[qs, qe]$.

```
1.  RSS_Lx = ∅; H = H' = ∅;
2.  Insert into H a pair of the form <e, Lx>
      for every entry e in the root of R;
3.  REPEAT
4.    WHILE H ≠ ∅ DO
5.      Remove top entry <e, Le [qes, qee]> of H;
6.      IF ∃ Lee[qees,qeee] ∈ Le: ∄ <p,Lp> ∈ RSS_Lx
          for which Lee ∈ Lp and p ≺_Lee e THEN
7.        IF qees ≠ qes THEN Insert <e,Lee> in H';
8.        ELSE
9.          IF e is an MBR THEN
10.           FOR every child ee of e DO
11.             IF ∃ Leee [qeees, qeeee] ∈ Lee:
                  ∄ <p, Lp> ∈ RSS_Lx for which
                  Leee ∈ Lp and p ≺_Leee ee THEN
12.               IF qeees ≠ qees THEN Insert <ee,
                    Leee> into H';
13.               ELSE Insert <ee, Leee> into H;
14.         ELSE
15.           FOR every pair <p, Lp> ∈ RSS_Lx for
                which Lee ∩ Lp ≠ ∅ DO
16.             IF ∃ Lpp ∈ (Lp ∩ Lee): e ≺_Lpp p THEN
17.               Update <p, Lp − Lpp> into RSS_Lx;
18.           Insert <e, Lee> into RSS_Lx;
19.   Move into H the pairs <e, Le[qes,qee]> ∈ H'
        with the point qes that is closest to qs;
20. UNTIL H' = ∅;
21. Return RSS_Lx;
```

Algorithm 1: Computation of the range skyline of a dataset P i.r.t. a query line segment $Lx[qs, qe]$ aligned with the x-axis.

rem 2, checks whether a sub-segment $Lee \in Le$ exists, i.r.t. which e is not dynamically dominated by any data point in RSS_{Lx}. If such a sub-segment Lee exists, then e belongs to a branch of the R-tree which can contain eligible points for the range skyline. Therefore, if Lee is not the left-most sub-segment of Le, then in Line 7 the entry $<e, Lee>$ is inserted into the auxiliary heap H' for future processing; otherwise, in Line 9 the algorithm checks whether e is an intermediate R-tree node entry or a data point.

In particular, if e is an intermediate R-tree node entry, then its *MBR* is expanded and, in Lines 10-11, for every entry ee, the algorithm checks (using Theorem 2) whether a sub-segment $Leee \in Lee$ exists, i.r.t. which ee is not dynamically dominated by any point in the range skyline set RSS_{Lx}. If such a sub-segment $Leee$ exists, then if $Leee$ is not the left-most sub-segment of Lee, in Line 12 the entry $<ee, Leee>$ is inserted into the auxiliary heap H' for future processing; otherwise in Line 13 it is inserted into heap H.

Else, if in Line 9 it is discovered that e is a data point, then e is definitely a dynamic skyline point i.r.t. the segment Lee, or i.r.t. its left-most sub-segment, the length of which will be determined later, in the next steps of the algorithm. However, if the RSS_{Lx} set is not empty, then before inserting into it the entry $<e, Lee>$, in Lines 15-17 for every entry $<p, Lp>$ in RSS_{Lx} for which $Lp \cap Lee \neq \emptyset$ it is checked (using Theorem 2, again) whether e dynamically dominates p i.r.t. the right-most sub-segment Lpp of Lp[3]. If such a domination exists, then the entry of p is updated in RSS_{Lx} from $<p, Lp>$ to $<p, Lp–Lpp>$ by removing the sub-segment Lpp from Lp.

The steps in Lines 4-18 are repeated for every subsequent top entry e of H. When H empties for the first time, then RSS_{Lx} stores all the data points which are dynamic skyline points i.r.t. at least the left-most end-point qs of Lx. Upon the termination of this process, all R-tree entries, which are dynamically dominated by data points in RSS_{Lx} i.r.t. qs, but which might be not dynamically dominated i.r.t. the whole line segment Lx, have been moved to the auxiliary heap H' for re-examination. For this reason, in Line 19, one batch of the entries in H' is moved back into H: these entries are the pairs $<e, Le [qes,qee]>$ with the left-most end-point qes that is closest to qs. Then, the loop in Lines 4-18 is executed again to process the new content of H, as the corresponding algorithm for the point-based dynamic skyline does [16]: this time, the loop is executed i.r.t. the new query point qes selected earlier, in Line 19.

When in Line 20 it is discovered that both the H and H' heaps are empty, then the algorithm terminates and the RSS_{Lx} set will contain entries of the form $<p, Lp>$, in which p is a dynamic skyline point of P i.r.t. the sub-segment $Lp \in Lx$; therefore, p is a range skyline point of P i.r.t. Lx.

LEMMA 3: *Algorithm* 1 *computes its output results progressively.*

The following lemmas jointly help to verify the correctness of the proposed algorithm, *i.e.* that no false hits and misses occur:

LEMMA 4. *Every data point of P that is added into the RSS_{Lx} set during the execution of Algorithm* 1 *is guaranteed to be a final range skyline point of P i.r.t. Lx.*

LEMMA 5. *If upon the termination of the execution of Algorithm* 1 *we have $<p, Lp> \in RSS_{Lx}(P)$, then $\nexists\ q \in Lp$: $\exists\ e \in P$ for which $e \prec_q p$ holds.*

LEMMA 6. *There is no other range skyline point of P i.r.t. Lx other than those in the RSS_{Lx} set.*

Below, Algorithm 1 is proved to be I/O optimal, meaning that: (i) it visits only the R-tree nodes that demonstrably can contain range skyline points, and (ii) it does not retrieve the same R-tree node twice from the secondary memory.

THEOREM 3. *The number of R-tree nodes accessed by Algorithm* 1 *is optimal.*

PROOF. To prove that Algorithm 1 expands only the R-tree intermediate entries that may contain range skyline points, let's assume that it also expands an R-tree intermediate entry e which cannot contain range skyline points. Without loss of generality, we consider a range skyline point p for which $p \prec_{Lx} e$ holds; therefore, e cannot contain any range skyline point. Then, in line with Definition 1, we understand that the distance from p to every point $q \in Lx$ is shorter than the corresponding *MINDIST* distance from e to q. Hence, p will be processed by Algorithm 1 before e. Therefore, in Line 6 of the algorithm, the entry e will have been pruned by p, contradicting the fact that e is visited and expanded.

[3] According to Lemma 4 below, since p is already in the RSS_{Lx} set, the data point e cannot dynamically dominate p i.r.t. the left-most sub-segment of Lp. Therefore, by taking into consideration Lemma 1, if e dynamically dominates p, this can be done only i.r.t. the right-most sub-segment of Lp.

Finally, we need to establish that an R-tree entry is not retrieved multiple times from the secondary memory. This is straightforward because the R-tree entries are retrieved from the secondary memory to be inserted into heap H (and expanded) one time at the most. Entries of heap H that represent branches of the R-tree, the processing of which is postponed for a later execution of the loop in Lines 3-20, are moved temporally into the auxiliary heap H'. □

We now follow the execution of Algorithm 1 for the running example of the dataset in Figure 2(a) for the computation of the range skyline i.r.t. a line segment Lx [qs, qe] aligned with the x-axis, with end-points qs (6, 5) and qe (10, 5). We assume that the dataset is indexed using the R-tree that is demonstrated in Figure 2(b). The contents of heaps H and H', as well as the contents of the RSS_{Lx} set during the execution of the algorithm, are illustrated in Table 2.

Table 2: Steps of an example execution of Algorithm 1.

Step	Action	H content	H' content	RSS_{Lx} content
1	Access R-tree root	<*Nc*, [*qs*, *qe*]>, <*Nb*, [*qs*, *qe*]>	∅	∅
2	Expand *Nc*	<*Nf*, [*qs*, *qe*]>, <*Nb*, [*qs*, *qe*]>, <*Ng*, [*qs*, *qe*]>	∅	∅
3	Expand *Nf*	<*Nb*, [*qs*, *qe*]>, <*i*, [*qs*, *qe*]>, <*h*, [*qs*, *qe*]>, <*g*, [*qs*, *qe*]>, <*Ng*, [*qs*, *qe*]>	∅	∅
4	Expand *Nb*	<*Nd*, [*qs*, *qe*]>, <*i*, [*qs*, *qe*]>, <*h*, [*qs*, *qe*]>, <*g*, [*qs*, *qe*]>, <*Ne*, [*qs*, *qe*]>, <*Ng*, [*qs*, *qe*]>	∅	∅
5	Expand *Nd*	<*i*, [*qs*, *qe*]>, <*c*, [*qs*, *qe*]>, <*h*, [*qs*, *qe*]>, <*b*, [*qs*, *qe*]>, <*a*, [*qs*, *qe*]>, <*g*, [*qs*, *qe*]>, <*Ne*, [*qs*, *qe*]>, <*Ng*, [*qs*, *qe*]>	∅	∅
6	Examine *i*	<*c*, [*qs*, *qe*]>, <*h*, [*qs*, *qe*]>, <*b*, [*qs*, *qe*]>, <*a*, [*qs*, *qe*]>, <*g*, [*qs*, *qe*]>, <*Ne*, [*qs*, *qe*]>, <*Ng*, [*qs*, *qe*]>	∅	<*i*, [*qs*, *qe*]>
7	Examine *c*	<*h*, [*qs*, *qe*]>, <*b*, [*qs*, *qe*]>, <*a*, [*qs*, *qe*]>, <*g*, [*qs*, *qe*]>, <*Ne*, [*qs*, *qe*]>, <*Ng*, [*qs*, *qe*]>	<*c*, ((8, 5), *qe*]>	<*i*, [*qs*, *qe*]>
...	...	...	...	...
19	Examine *k*	∅	<*c*, ((8 ,5), *qe*]>, <*f*, ((9.5,5), *qe*]>	<*i*, [*qs*, *qe*]>, <*e*, [*qs*, (7,5))>, <*l*, [*qs*, *qe*]>
20	Move *c* from *H'* into *H*	<*c*, ((8, 5), *qe*]>	<*f*, ((9.5,5), *qe*]>	<*i*, [*qs*, *qe*]>, <*e*, [*qs*, (7,5))>, <*l*, [*qs*, *qe*]>
21	Examine *c*	∅	...	...
...	...	...	...	...

The algorithm begins with the R-tree root and inserts its *MBRs* *Nc* and *Nb* into heap H, in the form <*Nc*, [*qs*, *qe*]>, <*Nb*, [*qs*, *qe*]>, sorted according to the *MINDIST* distance of the *MBRs* to *qs*. Then, the entry *Nc* is expanded. This process removes the entry from H and inserts the *MBRs* of its entries *Nf* and *Ng* into H. The next entry to be expanded is *Nf*, and its data points *i*, *h* and *g* are inserted into H. The projected versions of points *i*, *h* and *g* i.r.t. *qs* are displayed in Figure 2(a). The next *MBRs* to be expanded are *Nb* and *Nd*.

The first entry of H that corresponds to a data point is the nearest neighbour point *i* of *qs*, about which it is known from the literature that it belongs to the dynamic skyline of P i.r.t. *qs* [12]. Therefore; *i* is inserted into RSS_{Lx} in the form <*i*, [*qs*, *qe*]>, *i.e.* with an initial estimation that *i* is a dynamic skyline point of the dataset i.r.t. *Lx*. The examination of the remainder of the data points during the algorithm's execution might lead to an update of this initial [*qs*, *qe*] interval estimation as shown by Line 17 of the algorithm.

When processing the second data entry <*c*, [*qs*, *qe*]> of H, in Line 6, Table 1 will suggest that a line sub-segment Lc ((8, 5), *qe*] ∈ Lx exists[4], i.r.t. which *i* does not dynamically dominate *c*. Therefore, in Line 7, the entry <*c*, ((8, 5), *qe*]> is inserted into H'.

The algorithm continues by examining the next entries of H. By the time H empties, the algorithm has discovered all the dynamic skyline points i.r.t. at least the query point *qs*. Then, according to Line 19 of the algorithm, the entry <*c*, ((8, 5), *qe*]> is moved from H' back into H and the loop in Lines 3-20 is executed again.

When all data points are processed, the final content of RSS_{Lx} is {<*i*, [(6, 5), (10, 5)]>, <*e*, [(6, 5), (7, 5))>, <*l*, [(6, 5), (10, 5)]>, <*c*, ((8, 5), (10, 5)]>}, which shows that while the query point 'shifts' along the line segment Lx from left to right, then, i.r.t. the line sub-segment $L1x$ [(6, 5), (7, 5)) ∈ Lx, the dynamic skyline of P contains the data points *i*, *e* and *l*; i.r.t. the sub-segment $L2x$ [(7, 5), (8, 5)] ∈ Lx, the dynamic skyline contains the data points *i* and *l*; and i.r.t. the sub-segment $L3x$ ((8, 5), (10, 5)] ∈ Lx the dynamic skyline contains the points *i*, *l* and *c*. Therefore, with a single R-tree traversal, the algorithm can report the whole range skyline of the dataset i.r.t. the line segment Lx *i.e.* it can report the dynamic skyline of the dataset i.r.t. every query point on Lx.

3.1.2 *Two-dimensional Query Intervals*

Figure 1(b) illustrates the case of the 2-d rectangle: firstly, it allows a better grasp of how the algorithm works on the 2-d space, and the subsequent extension of the concept to the n-d case; secondly, there are many applications that may exclusively involve 2-d spaces ([6, 9, 14, 20], *etc.*), whence the attraction of this case.

Next, it is assumed that the user's preference consists of an interval of values on both the x- and the y- axes: the created query predicate is a rectangle Ω iso-oriented towards the axes of the 2-d space. Figure 1(b) illustrates a 2-d reference rectangle Ω (boundary inclusive), i.r.t. every individual point of which our user requests the dynamic skyline.

To compute the dynamic skyline of the dataset i.r.t. every single query point $q \in \Omega$, every data point p is projected onto another point p' with coordinate values that are computed using Equation (1). However, we can anticipate that, as the query point q rolls on the surface of the rectangle Ω from South-West to North-East, the coordinate values of the projected point p' of p might increase or might not change at all. Therefore, the point p' either rolls on the workspace in the same direction as q, *i.e.* from South-West to North-East, or remains in the same position.

Properties 1, 2 and 3 hold for each (x and y) dimension separately, dividing the dataset into 3^n different types of data points, where $n = 2$. For example, if we consider the case of the point a (a_x, a_y) in the region A of Figure 3(a), with $a_x \le qs_x$ and $a_y \le qs_y$, Property 1 holds in both dimensions and the projected point i.r.t. every point $q \in \Omega$ belongs in a rectangle with bottom-left vertex a' ($2qs_x - a_x$, $2qs_y - a_y$) and top-right vertex a'' ($2qe_x - a_x$, $2qe_y - a_y$). In accordance with Thales's Intercept Theorem [15], the length of the side of this rectangle in every dimension is twice the length of the corresponding side of rectangle Ω. Figure 1(b) illustrates the projection of all the data points of the running example.

[4] The left-most boundary end-point (8, 5) is not included in the line sub-segment Lc because, as Table 1 indicates, $i \prec_{(8,5)} c$ holds.

Theorem 2, as well as Lemmas 1 and 2, also hold in the case of a query predicate that is a 2-d rectangle Ω, given the following appropriate adjustments:

THEOREM 4 (EXTENSION OF THEOREM 2). *Given a dataset P of points in a 2-d space S, two points $p\,(p_x, p_y)$, $r\,(r_x, r_y) \in P$ and a reference rectangle Ω $[qs, qe]$ iso-oriented towards the axes of S, where $qs\,(qs_x, qs_y)$ is the bottom-left-most and $qe\,(qe_x, qe_y)$ is the top-right-most vertex points of Ω, Table 1 specifies the range of coordinate values, per axis of the space, of all points $q \in \Omega$ i.r.t. which $p \prec_q r$ holds.*

PROOF. If we assume two data points $p, r \in P$, the theorem determines that if (i) Table 1 indicates that, for the range of coordinate values $[qs_x, qe_x]$ of Ω on the x-axis, the data point p can dynamically dominate the data point r i.r.t. the sub-range of values $[qps_x, qpe_x] \in [qs_x, qe_x]$ and, if (ii) Table 1 indicates that, for the range of coordinate values $[qs_y, qe_y]$ of Ω on the y-axis, the data point p can dynamically dominate the data point r i.r.t. the sub-range of values $[qps_y, qpe_y] \in [qs_y, qe_y]$, then we need to combine the independent processes (i) and (ii) to discover that p dynamically dominates r i.r.t. all the query points q in the sub-region of Ω created by these two sub-segments, *i.e.* i.r.t. the sub-rectangle $\Omega p\ [(qps_x,qps_y), (qpe_x,qpe_y)] \equiv \Omega p\ [qps, qpe] \in \Omega$, where $qps\,(qps_x, qps_y)$ and $qpe\,(qpe_x, qpe_y)$ are the bottom-left-most and top-right-most vertices of Ωp, if such a sub-rectangle of Ω truly exists. □

LEMMA 7 (EXTENSION OF LEMMA 1). *Given a point dataset P, two points $p, r \in P$ and a rectangle Ω iso-oriented towards the axes of the space, for which $p \prec^P_\Omega r$ holds, then p dynamically dominates r i.r.t. only one single sub-rectangle $\Omega p \in \Omega$, which will also include at least one vertex of Ω.*

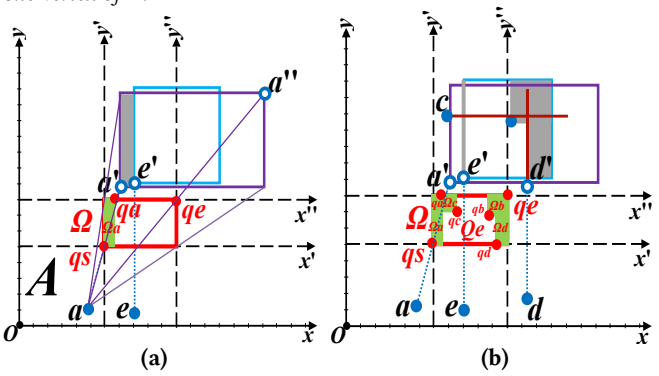


Figure 3: (a) The sub-rectangle Ωa (in green) of the query window Ω (in red), i.r.t. which the point a dynamically dominates the point e. (b) The combined picture of the dynamic domination of a point e by the points a, b, c, d, i.r.t. a rectangle Ω.

Figure 3(a) assumes a rectangle Ω iso-oriented towards the axes of the space and two data points a and e. When the query point q rolls on the surface of rectangle Ω, the projected versions a' and e' of the data points a and e roll in the same direction on the surface of the violet and blue rectangles. Therefore, a dynamically dominates e when a' is located in the grey part of the violet rectangle, which corresponds to the green sub-rectangle Ωa $[qs, qa]$ of Ω.

LEMMA 8 (EXTENSION OF LEMMA 2). *Given a dataset P of points, a point $p \in P$ and a rectangle Ω iso-oriented towards the axes of the space, if p is a range skyline point of P i.r.t. Ω then at most one single line sub-segment may exist in every one of the four sides of Ω, i.r.t. which p is a dynamic skyline point of P.*

Similarly to the example of Figure 3(a), Figure 3(b) illustrates the combined picture of the sub-regions (in green) of the query rectangle Ω, i.r.t. which a data point e is assumed to be dynamically dominated by four other data points a, b, c, and d. The point e is assumed to not be dynamically dominated by any data point i.r.t. the remainder of the sub-region Qe (in white) of Ω, and thus it is considered to be a dynamic skyline point of the illustrated dataset $P = \{a, b, c, d, e\}$ i.r.t. Qe.

The *pseudo*-code of the algorithm for processing the range skyline query i.r.t. a 2-d reference iso-oriented rectangle Ω is described in Algorithm 2, which is an extension of Algorithm 1 for 2-d query rectangles Ω. The algorithm uses two priority queues H and H' with pairs <e, Qe>, where e is an R-tree entry and Qe is the sub-region of Ω, i.r.t. which e has not yet been found to be dynamically dominated by any other data point. The entries in H' are sorted in the ascending order of the *MINDIST* distance of Qe to qs. The entries in H are sorted in the ascending order of the *MINDIST* of e to the query point $qes \in Qe$, located at the minimum distance from qs.

Algorithm 2: The Range Skyline $RSS_\Omega(P)$
Input: A dataset P indexed using an R-tree R and the query rectangle Ω $[qs, qe]$.

```
1.  RSS_Ω = ∅; H = H' = ∅;
2.  Insert into H a pair of the form <e, Ω> for
      every entry e in the root of R;
3.  REPEAT
4.    WHILE H ≠ ∅ DO
5.      Remove top entry pair <e, Qe> of H;
6.      IF ∃ Qee ∈ Qe: ∄ <p, Qp> ∈ RSS_Ω for which
          Qee ∈ Qp and p ≺_Qee e THEN
7.        IF (the point qees of Qee with the
            minimum distance to qs) ≠ (the point
            qes of Qe with the minimum distance to
            qs) THEN Insert <e, Qee> into H';
8.        ELSE
9.          IF e is an MBR THEN
10.           FOR every child ee of e DO
11.             IF ∃ Qeee ∈ Qee: ∄<p,Qp>∈ RSS_Ω for
                  which Qeee ∈ Qp and p ≺_Qeee e THEN
12.               IF (the point qeees of Qeee with
                    the minimum distance to qs) ≠
                    (the point qees of Qee with the
                    minimum distance to qs) THEN In-
                    sert <ee, Qeee> into H';
13.               ELSE Insert <ee, Qeee> into H;
14.         ELSE
15.           FOR every pair <p, Qp> in RSS_Ω for
                which Qee ∩ Qp ≠ ∅ DO
16.             IF there is a sub-rectangle Ωpp ∈
                  ∈ (Qp ∩ Qee): e ≺_Ωpp p THEN
17.               Update <p, Qp - Ωpp> into RSS_Ω;
18.           Insert <e, Qee> into RSS_Ω;
19.   Move into H the pairs <e, Qe> ∈ H' having
        the regions Qe with the MINDIST to qs;
20. UNTIL H' = ∅;
21. Return RSS_Ω;
```

Algorithm 2: Computation of the range skyline of a dataset i.r.t. a 2-d query iso-oriented rectangle Ω.

The first execution of the loop in Lines 3-20 examines whether there are data points that are dynamic skyline points i.r.t. a sub-region of Ω which includes the bottom-left-most end-point qs of Ω. When the heap H empties for the first time, then the auxiliary heap H' will contain all the R-tree entries which are dynamically dominated by data points in RSS_Ω i.r.t. qs but they might not be dynamically dominated i.r.t. every point in Ω. In Line 19, the algorithm selects and moves from H' to H all the pairs $<e, Qe> \in H'$, the region Qe of which is located at the *MINDIST* minimum distance from qs. If multiple regions Qe exist at the same *MINDIST* distance from qs but with a different query point $qes \in Qe$, which is located at the minimum distance from qs, then only the batch of entries of H' sharing the same qes are moved into H and the remaining entries of H' are processed in a future execution of the loop in Lines 3-20.

When it is discovered in Line 20 that heaps H and H' are empty, the algorithm terminates and the RSS_Ω set contains entries of the form $<p, Qp>$, in which p is a dynamic skyline point of P i.r.t. the sub-region $Qp \in \Omega$; thus, p is a range skyline point of P i.r.t. Ω.

The progressive behaviour and the correction of Algorithm 2 can be verified by taking an approach similar to Algorithm 1. The extension of Lemmas 3-6 and Theorem 3 is straightforward in order to also hold for query predicates that are 2-d rectangles.

Table 3 demonstrates the execution of Algorithm 2 in respect of the example in Figure 1(b) for the computation of the range skyline i.r.t. a query predicate that is a rectangle Ω [qs, qe] iso-oriented towards the axes of the 2-d space, where qs (6, 5) is the bottom-left-most vertex and qe (10, 8) is the top-right-most vertex point of Ω. We again assume that the dataset is indexed using the R-tree of Figure 2(b).

Up to the point of the processing of the entry $<c, \Omega$ [qs, qe]$>$ of H, the steps in Table 3 replicate the steps shown in Table 2. At this point we check whether a sub-rectangle of Ω can be found, i.r.t. which the data point i dynamically dominates the point c. Table 1 indicates that i dynamically dominates c i.r.t. the range of values [6, 8] on the x-axis and i.r.t. the range of values [5, 5.375] on the y- axis which, according to Lemma 7, create the sub-rectangle $\Omega 1ic$ [(6, 5), (8, 5.375)] $\in \Omega$. Thus, c is not dynamically dominated i.r.t. the sub-region $\Omega - \Omega 1ic$. Therefore, in Line 12 of the algorithm, the entry $<c, \Omega - \Omega 1ic>$[5] is inserted into heap H' since, in this first execution of the loop in Lines 3-20, the reference query point in Ω is its bottom-left-most vertex point qs and we have $qs \notin \Omega - \Omega 1ic$.

The algorithm then proceeds by examining the next entries of heap H, until it empties. This first execution of the loop in Lines 3-20 finds the dynamic skyline points i.r.t. qs. At this stage, the entry $<c, \Omega - \Omega 1ic> \in H'$, for which the sub-region $\Omega - \Omega 1ic$ has the smallest *MINDIST* to qs of all the entries in H', moves from H' to H and the loop in Lines 3-20 is executed again.

The final content of RSS_Ω is {$<i, \Omega -$ [(8, 5.375), (10, 8)]$>$, $<e, \Omega -$ [(7, 5), (10, 8)]$>$, $<l, \Omega -$ [(6, 6.125), (10, 8)]$>$, $<c, \Omega -$ [(6, 5), (8, 5.375)]$>$, $<a, \Omega -$ [(6, 5), (10, 7.125)]$>$, $<b, \Omega -$ [(6, 5), (10, 7)] – [(6.125, 5), (10, 8)]$>$}. Thus, with a single R-tree traversal, we discover the dynamic skyline i.r.t. every single query point in Ω.

[5] The sub-region "$\Omega - \Omega 1ic$" of Ω can also be represented in the heaps H and H' using the rectangles $Q1c$ ((6, 5.375), (8, 8)], $Q2c$ ((8, 5), (10, 5.375)] and $Q3c$ [(8, 5.375), (10, 8)], which construct it. From an implementation viewpoint, some extra bits can be used *per* rectangle to determine whether some of its edges or vertices belong to this sub-region.

Table 3: Steps of an example execution of Algorithm 2.

Step	Action	H content	H' content	RSS_Ω content
1	Access R-tree root	$<Nc, \Omega>$, $<Nb, \Omega>$	∅	∅
...	...	...	...	...
7	Examine c	$<h, \Omega>$, $<b, \Omega>$, $<a, \Omega>$, $<g, \Omega>$, $<Ne, \Omega>$, $<Ng, \Omega>$	$<c, \Omega -$ [(6, 5), (8, 5.375)]$>$	$<i, \Omega>$
...	...	...	...	...
19	Examine k	∅	$<c, \Omega -$ [(6, 5), (8, 5.375)]$>$, $<b, \Omega -$ [(6, 5), (10, 7)]$>$, $<a, \Omega -$ [(6, 5), (10, 5.75)]$>$, $<f, \Omega -$ [(6, 5), (9.5, 8)]$>$	$<i, \Omega>$, $<e, \Omega -$ [(7, 5), (10, 8)]$>$, $<l, \Omega -$ [(6, 6.125), (10, 8)]$>$
20	Move c from H' to H	$<c, \Omega -$ [(6, 5), (8, 5.375)]$>$	$<b, \Omega -$ [(6, 5), (10, 7)]$>$, $<a, \Omega -$ [(6, 5), (10, 5.75)]$>$, $<f, \Omega -$ [(6, 5), (9.5, 8)]$>$	$<i, \Omega>$, $<e, \Omega -$ [(7, 5), (10, 8)]$>$, $<l, \Omega -$ [(6, 6.125), (10, 8)]$>$, $<c, \Omega -$ [(6, 5), (8, 5.375)]$>$
21	Examine c	∅	...	...
...	...	...	...	...

3.2 Multi-dimensional Space

The discussion and the algorithm for processing the range skyline i.r.t. a rectangle iso-oriented towards the axes of the 2-d workspace can be straightforwardly extended to the case of the n-d space for $n > 2$. Indicatively, we simply reformulate Theorem 4:

THEOREM 5 (EXTENSION OF THEOREM 4). *Given a dataset P of points in an n-d space S, two points p, $r \in P$ and a reference hyper-rectangle Ω [qs, qe] iso-oriented towards the axes of S, where qs (qs_1, ..., qs_n) is the vertex with the lowest coordinate values of Ω on every axis, and where qe (qe_1, ..., qe_n) is the vertex with the highest coordinate values of Ω on every axis, Table 1 specifies the range of coordinate values, per axis, of all points $q \in \Omega$ i.r.t. which $p \prec_q r$ holds.*

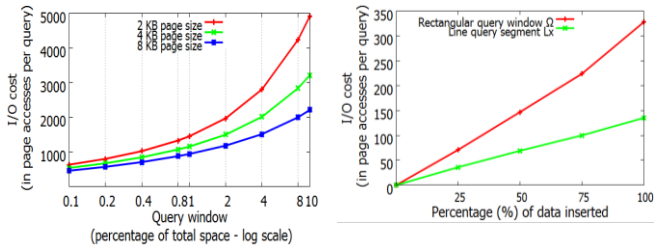
PROOF: It is a straightforward extension of the proof of Theorem 4 in n dimensions. □

4 PERFORMANCE EVALUATION

Experiments were conducted using two synthetic datasets composed of 10M data points on a 4-d workspace of 10,000 units in every dimension. Following [2], for the first dataset all the attribute values were generated independently using a uniform distribution; for the second dataset they were generated as anti-correlated values (points good in one dimension are bad in other dimensions). The datasets were indexed using the R*-tree [1] with a node size equal to the page size of the file system. The workstation was equipped with Intel I7 16GB RAM running the Windows 8.1 Pro 64-bit OS.

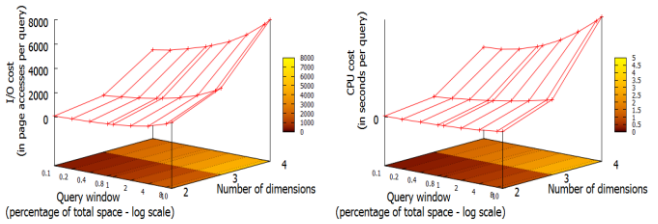
Every experiment was repeated ten times and, at every run, a different randomly-located query range was chosen. Due to length restrictions only a selection of the findings of the performance investigation of the proposed algorithm are presented.

Experiment 1: Figure 4(a) demonstrates the impact of the page size of the file system on the I/O cost (*i.e.* number of disk accesses) for answering the range skyline query i.r.t. nine different values for



(a) (b)

Figure 4: (a) Query window size *vs.* I/O cost for three page sizes. (b) Dataset size *vs.* I/O cost, for two types of query predicates.



(a) (b)

Figure 5: (a) Dimensionality *vs.* I/O cost, and (b) Dimensionality *vs.* CPU time cost, in both cases for several query window sizes.

the size of the rectangular query window, using a 3-d dataset containing 1M data points of independent attributes. 2K, 4K, and 8K disk page sizes are studied. As expected, the number of pages accessed by the algorithm decreases when the page size increases, and as the query window increases, so does the number of accessed disk pages. This is explained by the fact that, when the query window increases, both the area to be searched and the expected number of range skyline points increase.

Experiment 2: Figure 4(b) presents the impact of the dataset size on the I/O cost for executing the proposed algorithm using a 2-d dataset with 10M data points of independent attributes and a 4K page size. The query is executed at every 25% intervals of the data inserted, every time i.r.t. two different types of query predicates: (i) a rectangle Ω iso-oriented towards the axes of the space, covering 1% of the area of the space, and (ii) a line segment Lx parallel to the x-axis with a length equal to the length of the side of the above rectangle Ω on the x-axis. Although the window Ω covers a much larger surface of the workspace than the line segment Lx covers, the figure shows that the I/O cost for answering the query i.r.t. Ω is only about 2.5 times the corresponding cost i.r.t. Lx. This result is due to the fact that the R-tree nodes that overlap the searched area i.r.t. Lx also overlap the area searched i.r.t. Ω. Then in the case of Ω, the algorithm accesses only a few more (adjacent on the space surface) R-tree nodes than it would access in the case of Lx.

Experiment 3: Figure 5(a) illustrates the impact of the space dimensionality on the I/O cost to satisfy the range skyline query i.r.t. query windows of nine different sizes, using a 3-d dataset with 1M data points of anti-correlated attributes. Three different dimensionalities are studied *i.e.* 2, 3, and 4. The figure points to the rapid increase of the I/O cost with the increase of dimensionality, due to the fact that an increase in the number of dimensions makes it less possible for a data point to be dynamically dominated. Thus, the number of range skyline points increases (more dimensions imply more dynamic dominance checks and thus more range skyline points). The I/O increase is also due to the gradual degradation of the performance of R*-trees as the number of dimensions increases.

Same experiment: Figure 5(b) measures the impact of the number of dimensions on the CPU time cost (in seconds). The I/O cost presented in Figure 5(a) and the CPU time cost presented in Figure 5(b) indicate that if the query range is sufficiently small or the available main memory sufficiently large (or if an acceptable combination of these two parameters exists) for the entire priority queues to be stored in main-memory, then most of the time cost for the execution of the algorithm is spent on accessing R-tree nodes to collect eligible range skyline points. On the other hand, if the priority queues are too long to fit in main-memory, a disk-based structure (e.g. a B^+-tree) needs to be adopted. However, it would be counterproductive for a user to express a large range of preferences on every axis, since this would produce a large number of range skyline points, and would be unhelpful for decision-making.

5 RELATED WORK

Since the introduction of the skyline operator in 2001 [2], two categories of algorithms have stood out for the 'static' and for the dynamic skyline queries i.r.t. a query point: non-index algorithms that scan the whole dataset to provide the results (e.g. [2, 4]) and index-based algorithms (e.g. [12, 16]), which include the state-of-the-art index-based Branch-and-Bound Skyline algorithm [16], and which inspired the present work. Venturing beyond data management domains and applications, this paper examines the range skyline search as a preference-oriented type of query, to which none of the existing point-based algorithms is applicable, given the infinite number of query points in an n-d range.

Huang et al. [9] propose the continuous monitoring of the skyline, whereby both the query point and the data points continuously shift along a line at a constant speed in every dimension. Fu et al. [7] model the approximate location of the moving query on the 2-d road network as a range, instead of as a point, and compute the skyline after defining the domination relation between two data points on the basis of both their distance to the query and the monotonic order in any other available attribute. Their approach only considers the 'static' skyline and only in the spatial networks domain. Variants of continuous skyline queries in road networks (e.g. [20]) are not all applicable to the more generic scenario proposed here.

Since the confidentiality of a user's preferences and location became an important issue, range-based queries have been incorporated into skyline algorithms. Papadias et al. [16] introduced the *constrained skyline query* whereby the results of the 'static' skyline query include only the most interesting data points with coordinate values within a range. Rahul and Janardan [17] proposed a method for retrieving the local 'static' skyline of all the data points lying within a small neighbourhood of interest, specified as a query region on the xy-plane. Lin et al. [14] coined the term *range-based skyline query* in which the 'static' skyline for the non-spatial attributes of the data points is refined on the basis of the distance of these points to a rectangular query predicate. A recent approach considering the 'static' skyline is that of Lai et al. [13] who focused on the distributed domain over mobile wireless sensor networks.

Given a query range, with the algorithm proposed in [24], the data points which are dynamically dominated by other data points i.r.t. the whole range can be pruned. However i) it produces false hits as the method is not able to prune all the data that do not belong to the dynamic skyline i.r.t. the given range; ii) the final result can only be retrieved using a non-index skyline processing algorithm (e.g. [4]); iii) this approach cannot fully take advantage of an index-based skyline processing algorithm to reduce the computation cost; and iv) it also differs from our work in respect of the definition of the range skyline set, since, for [24], a data point can be in the range skyline only if it is not dynamically dominated by any other data point i.r.t. every query point in the given range (see our Definition 3). Our approach considers a data point which can be in the range skyline if it is not dynamically dominated by any other data point i.r.t. any individual query point in the range *i.e.* not necessarily i.r.t. all the query points in the range (see our Definition 5). Our solution also defines the sub-region of the given *n*-d range, i.r.t. which a data point can belong to the range skyline set.

Finally, some studies consider several reference query points simultaneously co-existing on the workspace [6, 19] and some other studies consider caching mechanisms (e.g. [18]) to speed up point-based dynamic skyline queries issued in close time instants using query points quite close to each other. Our approach deals with such problems with a single R-tree traversal, after drawing a range that includes all the sources (*i.e.* query points) that the users have submitted and by searching, in the results, for the data points belonging in the dynamic skyline of every individual source.

6 CONCLUSIONS AND FUTURE RESEARCH

This paper extends the concept of the point-based dynamic skyline query and proposes the range skyline query for applications which allow users to submit preferences using an interval of values on every dimension of the *n*-d space, rather than a precise value. This approach computes the dynamic skyline query i.r.t. the infinite set of query points in the given range. An algorithmic solution is presented and several geometric properties and heuristics are considered to avoid having to compute the dynamic skyline i.r.t. every single point in the range. The proposed algorithm is progressive and it provides I/O optimality, and no false hits and misses. Its efficiency and robustness is tested under empirical conditions with encouraging results.

Looking into the future, a comprehensive theoretical analysis of the lower and upper bounds of the I/O cost performance behaviour of the proposed algorithm needs to be carried out on foundation work on the point-based skyline query (for example the range skyline cardinality can be estimated by building upon the work of [3, 22] and others). Prior theoretical work in the area of the traditional range query [21] cannot be easily extended and applied to the range skyline query problem since the surface on the workspace that needs to be searched is not known beforehand in this more complex case of cost estimation.

Other lines of pursuit regard i) the use of the algorithm with very large data sets and in high-dimensional spaces, where the priority queues may have to be stored in a disk instead of in main-memory and ii) the efficient support of the continuous on-line maintenance of the range skyline set in dynamic environments where data points can move or the query range may change continuously.

The range skyline query is expected to benefit numerous data management applications and to apply in several other emerging domains requiring preference-based computation such as the privacy preservation domain, which is facing the increasingly urgent challenge of preserving confidentiality, thus moving from the era of user preferences stated with precise values to the era of preferences stated in the more opaque terms of a blurred range of values.

REFERENCES

[1] N. Beckmann, H. P. Kriegel, R. Schneider, and B. Seeger. 1990. The R*-tree: an efficient and robust access method for points and rectangles. In *SIGMOD Proceedings*, 322–331.

[2] S. Borzsonyi, D. Kossmann, and K. Stocker. 2001. The skyline operator. In 17th *ICDE Proceeding,* 421–430.

[3] C. Buchta. 1989. On the average number of maxima in a set of vectors. *Information Processing Letters*, 33(2):63-65.

[4] J. Chomicki, P. Godfrey, J. Gryz, and D. Liang. 2003. Skyline with presorting. In 19th *ICDE Proceedings*, 717–719.

[5] B. Cui, H. Lu, Q. Xu, L. Chen, Y. Dai, and Y. Zhou. 2008. Parallel distributed processing of constrained skyline queries by filtering. In 24th *ICDE Proceedings*, 546–555.

[6] K. Deng, X. Zhou, and H. T. Shen. 2007. Multi-source skyline query processing in road networks. In 23rd *ICDE Proceedings*, 796-805.

[7] X. Fu, X. Miao, J. Xu, and Y. Gao. 2017. Continuous range-based skyline queries in road networks. *World Wide Web*, 20(6):1443–1467.

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

[9] Z. Huang, H. Lu, B. C. Ooi, and A. K. H. Tung. 2006. Continuous skyline queries for moving objects. *IEEE TKDE*, 18(12):1645–1658.

[10] M. E. Khalefa, M. F. Mokbel, and J. J. Levandoski. 2008. Skyline query processing for incomplete data. In 24th *ICDE Proceedings*, 556-565.

[11] M. Kontaki, A. N. Papadopoulos, and Y. Manolopoulos. 2008. Continuous *k*-dominant skyline computation on multidimensional data streams. In 23rd *SAC Proceedings*, 956–960.

[12] D. Kossmann, F. Ramsak, and S. Rost. 2002. Shooting stars in the sky: An online algorithm for skyline queries. In 28th *VLDB Proceedings*, 275–286.

[13] C. C. Lai, Z. F. Akbar, and C. M. Liu. 2016. A cooperative method for processing range-skyline queries in mobile wireless sensor networks. In 6th *EDB Proceedings*, 1–8.

[14] X. Lin, J. Xu, and H. Hu. 2013. Range-based skyline queries in mobile environments. *IEEE TKDE*, *25*(4):835–849.

[15] A. Ostermann, and G. Wanner. 2012. *Geometry by its history*. Springer Science & Business Media.

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

[17] S. Rahul, and R.Janardan. 2012. Algorithms for range-skyline queries. In 20th *SIGSPATIAL Proceedings*, 526-529.

[18] D. Sacharidis, P. Bouros, and T. K. Sellis. 2008. Caching Dynamic skyline queries. In 20th *SSDBM Proceedings*, 455–472.

[19] M. Sharifzadeh, and C. Shahabi. 2006. The Spatial skyline queries. In 32nd *VLDB Proceedings*, 751–762.

[20] C. Shi, X. Qin, and L. Wang. 2015. Continuous skyline queries for moving objects in road networks. *Journal of Software*, 10(2):190–200.

[21] Y. Theodoridis, and T. Sellis. 1996. A model for the prediction of R-tree performance. In 15th *PODS Proceedings*, 161–171.

[22] E. Tiakas, A. N. Papadopoulos, and Y. Manolopoulos. 2013. On estimating the maximum domination value and the skyline cardinality of multidimensional data sets. *International Journal of Knowledge-Based Organizations*, 3(4):61–83.

[23] Y. Wang, X. Li, X. Li, and Y. Wang. 2013. A survey of queries over uncertain data. *Knowledge & Information Systems*, 37(3):485–530.

[24] W. C. Wang, E. T. Wang, and A. L. Chen. 2011. Dynamic skylines considering range queries. In 16th *DASFAA Proceedings,* 235–250.