Directional Queries: Making Top-k Queries More Effective in Discovering Relevant Results

Directional Queries: Making Top-k Queries More Effective in Discovering Relevant Results

ABSTRACT

Top-k queries, in particular those based on a linear scoring function, are a common way to extract relevant results from large datasets. Their major advantage over alternative approaches, such as skyline queries (which return all the undominated objects in a dataset), is that the cardinality of the output can be easily controlled through the k parameter and user preferences can be accommodated by appropriately weighing the involved attributes.

In this paper we concentrate on two so-far neglected aspects of top-k queries: first, their general ability to return *all* the potentially interesting results, i.e., the tuples in the skyline; second, the difficulty that linear top-k queries might encounter in returning tuples with *balanced* attribute values that match user preferences more closely than tuples that are extremely good in one dimension but (very) poor in others. In order to quantify these undesirable effects we introduce four novel indicators for skyline tuples, which measure their robustness as well as the difficulty incurred by top-k queries to retrieve them.

After observing that real datasets usually contain many relevant results that are hardly retrievable by linear top-k queries, and with the aim of favoring balanced results, we extend the queries with a term that accounts for the *distance* of a tuple from the preference direction established by the attributes' weights. This novel query, which we call *directional query*, adds the flexibility needed to allow *each* skyline tuple to be ranked first for a proper choice of weights, with no extra burden on the user and, in the most adverse scenarios, only a minor computational overhead, as measured through an extensive experimental analysis on real and synthetic data.

CCS CONCEPTS

• **Information systems** → **Data management systems**; **Top-k retrieval in databases**.

KEYWORDS

scoring functions, skylines, user preferences, balanced results

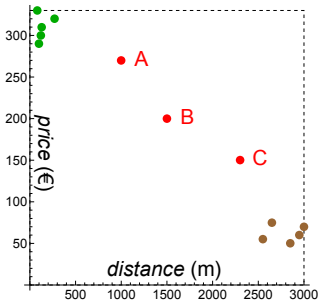
ACM Reference Format:

. 2018. Directional Queries: Making Top-k Queries More Effective in Discovering Relevant Results. In . ACM, New York, NY, USA, 14 pages. https://doi.org/XXXXXXX.XXXXXXX

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM

https://doi.org/XXXXXXX.XXXXXXX



(a) Dataset

		weights		
	w_d	0.3	0.5	0.7
	w_p	0.7	0.5	0.3
hotel ranks	A	11	10	6
	B	7	8	7
	C	6	13	9

(b) Ranks of A, B, and C

Figure 1: A hotel dataset and ranks of some hotels.

1 INTRODUCTION

Ranking queries based on linear scoring functions (i.e., using a weighted sum of the attribute values) are undoubtedly the most common way to obtain relevant results from large, multi-attribute datasets. This stems from a series of factors, including the ability to limit the cardinality of the result (hence the alternate name "top-k queries", where k is the output size), the possibility of incorporating user preferences (in the form of *weights*), and the availability of efficient indexing and processing methods [20]. In spite of this, linear top-k queries have several shortcomings, including the difficulty in specifying exact values for the weights, which is an especially hard task with many attributes [11, 25, 28].

In this paper, we focus on further limitations of linear top-k queries that have so far been neglected. First, the best result according to any linear top-k query is restricted to be an element of the *convex hull* of the dataset [6]. This can lead to missing relevant results, even when large values of k are used. Second, depending on data distribution, tuples with somehow *balanced* attribute values might be hard to retrieve. Example 1.1 illustrates both aspects.

Example 1.1. Ana is looking for a room and wants to minimize both the price and the distance from the city center. The hotels still available for reservation, shown in Figure 1a, mainly concentrate in two clusters: luxury hotels located at most a few hundred meters from the center, and budget hotels, further away from the center. Hotels A, B, and C are more balanced alternatives lying between these two extreme cases, with intermediate prices and not too far from the center – a combination that might indeed be an interesting trade-off for Ana. Assume that hotels are ranked by a linear function combining (normalized) price and distance, i.e., $w_d \cdot distance/3000 + w_p \cdot price/330$. No matter how Ana specifies the weights, the best result will be a hotel in one of the two clusters. Furthermore, for *any* choice of the weights, the top-5 set will never include any of A, B, or C, whose ranks are shown in Figure 1b for several combinations of w_d and w_p (the lower the rank, the better).

As the example highlights, a dataset might well contain interesting results that are hardly retrievable by any linear top-k query, thus

limiting the possibility of users to access relevant alternatives and reducing the effectiveness of preferences. The same problem also occurs with more sophisticated approaches, like *regret queries* [28] and the ORD and ORU operators in [25], since all of them stay within the realm of linear functions. Although one may be tempted to circumvent the problem by using non-linear (e.g., quadratic or cubic) scoring functions, the analysis we provide in Section 3.2 shows that, in practice, there are objects that can become top-1 only with functions of very high polynomial degree, which makes this approach hardly applicable. Furthermore, the higher the non-linearity of the function, the less user preferences are effective in influencing the ranking, as discussed in Section 4.

Relevant points like A, B, or C in Example 1.1 could be obtained by computing the *skyline* of the dataset [3] – a point t belongs to the skyline iff t is *undominated* (s dominates t if it is no worse than t on all attributes and strictly better for at least one). However, skyline queries have major shortcomings, as they cannot accommodate user preferences and are unable to limit the cardinality of the result.

To provide a user-independent and objective way to determine whether a point is relevant, we identify the relevant points with those in the skyline. The rationale for this choice is that all other points are dominated, thus less interesting. Clearly, the relevance of different skyline points might change depending on the specified user preferences. In Section 3, we introduce a set of 4 *indicators*, aiming to quantify both the *difficulty* of retrieving skyline points with top-k queries and their *interestingness*. The findings we obtain from both real and synthetic datasets lead us to conclude that the existence of hard-to-retrieve yet interesting results is the rule rather than the exception.

To overcome the limits of linear top-k queries, while avoiding additional complexity on the user side, in Section 4 we introduce an original type of scoring functions, which give rise to what we call *directional queries*. The rationale for these queries is that, besides considering a weighted mean of the attribute values, as linear ranking queries do, they also include a term accounting for how much a point is *balanced* with respect to the stated user preferences. In the basic case in which all weights are equal, this would favor points close to the main diagonal of the data space, whereas in the general case the distance with respect to a *preference line* (easily derived from the weights) can be used. By referring to Example 1.1, one could indeed argue that B is the best compromise if one aims to balance the price and distance criteria, yet B is one of the worst hotels if one runs a linear ranking query with equal weights! Directional queries can easily solve problems like this and, generally, achieve a higher result quality than linear top-k queries, without incurring extra computational overhead.

Summarizing, our contributions are: *i)* we characterize the limits of linear top-k queries in retrieving relevant results by introducing a set of novel indicators, and we experimentally show that interesting results that are hard to retrieve are very common; *ii)* to overcome the above limits, we introduce the novel notion of directional query, which combines the classical mean-based component of linear top-k queries with another one based on the distance from the preference line, a novel concept exploiting the notion of balanced results; *iii)* we show that, with directional queries (and unlike linear queries), any skyline point can be the best result; *iv)* we provide an extensive experimental analysis that shows that directional queries are more effective than linear top-k queries, as well as other methods including non-linear scoring functions, in terms of quality of results without requiring any significant overhead. Proofs are omitted or only sketched in the interest of space.

2 BACKGROUND

Consider a relational schema $R(A_1, \ldots, A_d)$, with $d \geq 1$ numeric attributes whose domains are, respectively, $D_1, \ldots, D_d$. A *tuple* $t = \langle v_1, \ldots, v_d \rangle$ over R is an element of $\mathcal{D} = D_1 \times \ldots \times D_n$; each v_i is denoted by $t[i]$. Without loss of generality, in the following we assume $\mathcal{D} = [0, 1]^d$, unless otherwise specified. An *instance* over R is a set of tuples over R; in the following, we refer to an instance r over R with $|r| = N$ tuples.

Definition 2.1 (Dominance and skyline). Let s, t be tuples over R. Then, t *dominates* s, written $t \prec s$, if *(i)* $\forall i.\, 1 \leq i \leq d \rightarrow t[i] \leq s[i]$, and *(ii)* $\exists j.\, 1 \leq j \leq d \wedge t[j] < s[j]$. The *skyline* of r, denoted by $\textsc{Sky}(r)$, is defined as $\textsc{Sky}(r) = \{t \in r \mid \nexists s \in r.\, s \prec t\}$.

A *scoring function* f is a function $f : \mathcal{D} \rightarrow \mathbb{R}^+$. For a tuple $t = \langle v_1, \ldots, v_d \rangle$ over R, the value $f(v_1, \ldots, v_d)$ is called the *score* of t, also written $f(t)$. We conventionally consider lower attribute values to be better than higher ones, as would be appropriate for attributes representing "cost" and similar characteristics (but of course the opposite convention would also be possible); consequently, lower score values are also preferred over higher ones.

The *rank* $\mathsf{rank}(t; r, f)$ of a tuple $t \in r$ according to scoring function f is 1 plus the number of tuples with a better score than $f(t)$, i.e., $\mathsf{rank}(t; r, f) = 1 + |\{s \in r \mid f(s) < f(t)\}|$. We write $\textsc{Top}_k(r; f)$ to indicate a set of top-k ranked tuples ($k \geq 1$) in r according to f.[1]

We generally refer to a family $\mathcal{L}_p$ of scoring functions with a *weight vector* $w = \langle w_1, \ldots, w_d \rangle$ whose components are normalized, i.e., $\sum_{i=1}^{d} w_i = 1 \wedge \forall i.\, w_i \in [0, 1]$:

$$\mathcal{L}_p = \left\{ f \;\middle|\; f(t) = \left(\sum_{i=1}^{d} w_i t[i]^p \right)^{\frac{1}{p}} \right\}, \quad p \geq 1 \tag{1}$$

The vast majority of cases considered in the literature refer to top-k queries using a scoring function in $\mathcal{L}_1$, simply called *linear* top-k queries in the following.

By interpreting the tuples of a relation r as points in a d-dimensional space, we can define the *convex hull* of r as the intersection of all convex sets containing r. As is well known [6], a prominent limitation of linear top-k queries is that they can never rank as first any skyline tuple not in the convex hull of r. Following this geometric view, using a scoring function in $\mathcal{L}_1$ corresponds to moving from the origin and sweeping the data space with a hyperplane orthogonal to the weight vector [33];[2] the order in which the points are intercepted determines the ranking of the corresponding tuples. Similar considerations apply to $\mathcal{L}_p$, in which case we have "curved" fronts (one such example will be shown in Section 3 in Figure 2b for $\mathcal{L}_2$).

[1] The set is unique when no ties occur.
[2] Clearly, hyperplanes are planes in 3D and lines in 2D.

3 INDICATORS

Several previous research attempts, including [5, 8, 22–24, 28, 29, 31, 34–38], have tried to assess the "strength" of skyline tuples so as to be able to rank such tuples and thereby control the cardinality of the skyline – indeed, unlike top-k queries, skyline queries suffer from the inability to impose an output size. Inspired by these efforts, in this section, we discuss four indicators (three are novel contributions of this work and one generalizes a previous proposal) that we use for measuring properties of skyline tuples.

The first two indicators we introduce, namely *best rank* (Section 3.1) and *concavity degree* (Section 3.2) aim to quantify the "difficulty" of retrieving a skyline tuple t. We also consider two other indicators for measuring the "robustness" of a skyline tuple t, namely the *exclusive volume* (Section 3.3) and the *grid resistance* (Section 3.4).

We shall use our indicators to address an important research question: are there natural occurrences of skyline tuples that are highly interesting (i.e., they distinguish themselves neatly from other skyline tuples and are thus particularly robust according to our indicators) but difficult to obtain with a linear top-k query? We address this question (and answer it in the affirmative) in Section 3.6, where we also discuss our experiments on indicators, an excerpt of which is shown in Table 2 in that section as aggregate values over various datasets.

Besides the purposes for which the indicators are used in the present work, they are interesting in their own right, and might equally well be adopted in all contexts in which skyline tuples need to be compared (including skyline cardinality control). To this end, we extend to our indicators two properties that have been studied [28] for assessing variants of top-k and skyline queries. An indicator is *scale-invariant* if it ranks skyline tuples in the same way even if we multiply all values in some attribute by a constant. An indicator is *stable* if it does not depend on dominated tuples (i.e., adding or removing "uninteresting" tuples does not change the ranking of skyline tuples). Table 1 offers an overview of the extent and properties of our indicators.

3.1 Best rank

The first indicator we introduce considers how well a skyline tuple can perform when a set of possible scoring functions $\mathcal{F}$ is in use; these functions represent the possible criteria to be used for ranking tuples. The *best rank* of t is the minimum of $\mathsf{rank}(t; r, f)$ when all functions in $\mathcal{F}$ are considered:

$$\mathsf{brank}(t; r, \mathcal{F}) = \min_{f \in \mathcal{F}} \mathsf{rank}(t; r, f) \tag{2}$$

For example, $\mathsf{brank}(t; r, \mathcal{L}_1)$ returns the best rank that t can have with any linear top-k query, independently of the weights. Larger values thus indicate a higher difficulty in retrieving t. For ease of notation, we shall write $\mathsf{brank}_p(t; r)$ to indicate $\mathsf{brank}(t; r, \mathcal{L}_p)$.[3]

Indicator	Symbol	Measures	Stable	Scale-inv.
Best rank	brank	Difficulty	No	Yes
Concavity degree	cdeg	Difficulty	Yes	Yes
Exclusive volume	evol	Robustness	Yes	Yes
Grid resistance	gres	Robustness	Yes	Yes

Table 1: Overview of the indicators

Example 3.1. Figure 2a shows a dataset r whose skyline consists of t_1, t_2, and t_3. We have $\mathsf{brank}_1(t_1; r) = 1$ (as can be seen, e.g., by increasing the values obtained via a linear scoring function of the form $f(t) = \frac{2}{3}t[1] + \frac{1}{3}t[2]$, i.e., scanning the dataset from the origin with lines perpendicular to the weight vector $\langle \frac{2}{3}, \frac{1}{3} \rangle$, as the blue line, which meets t_1 first). Similarly, $\mathsf{brank}_1(t_3; r) = 1$ (following the purple line). Yet, t_2 never ranks first with any linear top-k query: its best rank is $\mathsf{brank}_1(t_2; r) = 3$ (following, e.g., the blue arrow, one meets t_2 after t_1 and t_7). Changing the direction does not improve t_2's rank: t_2 is met, e.g., after t_1 and t_3 if one follows the green arrow or after t_3 and t_4 when moving along the purple arrow.

As is apparent in the previous example, the best rank of a skyline tuple may be affected by the presence of non-skyline tuples (for instance, without t_7, the best rank of t_2 would be 2 instead of 3), so brank is not stable. However, brank is scale-invariant in $\mathcal{L}_p$, since any multiplicative factor applied to a dimension can be countered by dividing by the same factor the weight for that dimension in the scoring function (and then re-normalizing the weights).

Proposition 3.2. brank_p *is scale-invariant for every* $p \in \mathbb{N}$.

We observe that with linear functions the set of possible top-1 results is the intersection of $\textsc{Sky}(r)$ and the vertices of the convex hull of r. Consequently, we say that a skyline tuple $t \in \textsc{Sky}(r)$ is *convex* if $\mathsf{brank}_1(t; r) = 1$, *concave* otherwise.

3.2 Concavity degree

The second indicator we introduce to characterize the difficulty encountered to retrieve a skyline tuple is called cdeg (for *concavity degree*). Unlike brank, cdeg has the advantage of being stable, thus insensitive to the presence of dominated tuples. The intuition about cdeg is that it provides a way to measure the amount of non-linearity needed so that also a *concave* skyline tuple can become top-1 for some scoring function (as observed, concave skyline tuples are never the top-1 result for any linear top-k query).

Example 3.3. Figure 2b shows that t_2 is concave and any linear top-k query would rank as first either t_1 or t_3 (which are convex).

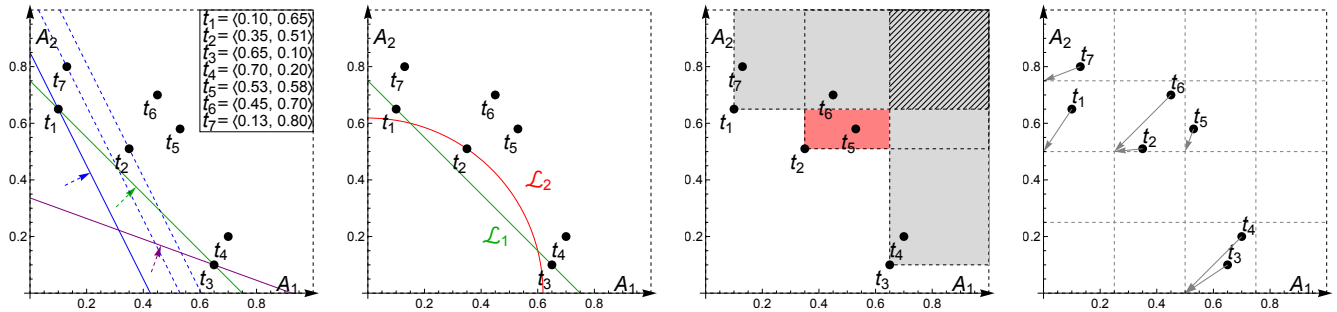
As a major result, useful for defining cdeg, we can prove that *every* skyline tuple is the top-1 result for some function in $\mathcal{L}_p$, for a sufficiently large p.

Theorem 3.4. *If* $t \in \textsc{Sky}(r)$ *then* $\exists p^*$ *s. t., for* $p \geq p^*$, $\mathsf{brank}_p(t; r) = 1$.

Proof sketch. To provide an intuition on this result, we consider the notion of convex $\mathcal{L}_p$-combination [11], which generalizes the standard notion of convex combination:

Definition 3.5. For a family $\mathcal{L}_p$, a (virtual) tuple v is a *convex* $\mathcal{L}_p$*-combination* of $t_1, \ldots, t_n$, $n > 1$, if $\exists \alpha_1, \ldots, \alpha_n$ such that $\alpha_j \in [0, 1]$ for j in $1..n$, $\sum_{j=1}^{n} \alpha_j = 1$, and $v[i]^p = \sum_{j=1}^{n} \alpha_j t_j[i]^p$, for i in $1..d$.

[3] Note that brank_1 coincides with the *MaxRank* operator of [27]. However, we prefer to qualify it as "best rank", since one is indeed looking for the *minimum* (i.e., best) possible rank of a tuple.



(a) Best rank brank_1 **(b) Concavity degree** cdeg **(c) Exclusive volume** evol **(d) Grid resistance** gres

Figure 2: Illustration of the indicators.

Theorem 4.7 in [11] proves that a skyline tuple t is top-1 for some function in $\mathcal{L}_p$ iff no convex $\mathcal{L}_p$-combination dominates t. Since the α_j's sum up to 1, there exists $\alpha_{j^*} \geq 1/n$. We observe that, since t is a skyline tuple, the relationship $\frac{1}{n} t_{j^*}[i_{j^*}]^p > t[i_{j^*}]^p$ must hold for some p and some attribute i_{j^*}; this, in turn, implies $\alpha_{j^*} t_{j^*}[i_{j^*}]^p > t[i_{j^*}]^p$, hence non-dominance. □

We call the smallest integer p such that $\text{brank}_p(t;r) = 1$ the *concavity degree* of t and denote it as $\text{cdeg}(t;r)$:

$$\text{cdeg}(t;r) = \min_{p \in \mathbb{N}} \{p \mid \text{brank}_p(t;r) = 1\} \tag{3}$$

As a consequence of Theorem 3.4, we have that for $\bar{p} = \max_{t \in \text{Sky}(r)} \{\text{cdeg}(t;r)\}$, *all* skyline tuples become possible top-1 results.

Example 3.6. Continuing Example 3.3, Figure 2b shows that no straight front (like the green line, corresponding to the locus of points with the same score as t_1 and t_3 according to a scoring function $f_1(t) = 0.5t[1] + 0.5t[2]$ in $\mathcal{L}_1$) can meet t_2 before t_1 and t_3. However, this is possible with a curved front, such as the red arc, which corresponds to the points with the same score as t_2 according to a scoring function $f_2(t) = \sqrt{0.5t[1]^2 + 0.5t[2]^2}$ in $\mathcal{L}_2$. Therefore, $\text{cdeg}(t_2;r) = 2$, whereas $\text{cdeg}(t_1;r) = \text{cdeg}(t_3;r) = 1$.

As anticipates, unlike brank, the concavity degree of a tuple t is insensitive to dominated tuples. Moreover, cdeg is scale-invariant, since convex combinations can even out any scaling factor.

Proposition 3.7. cdeg *is both stable and scale-invariant.*

3.3 Exclusive volume

When assessing the interestingness of a skyline tuple t one may want to assess its very ability to dominate other, non-skyline tuples. A first, rough attempt might consist in counting the number of tuples dominated by t (indeed, this has been done in the relevant literature [29]). Yet, such a counting is, by definition, dependent on the presence of dominated tuples (i.e., uninteresting or less interesting solutions). A more targeted approach would consist in considering the number of dominated tuples that only t (and no other skyline tuple) dominates, but dependence on dominated tuples would of course still be an issue. What we propose in this section is an indicator that captures the potential of a skyline tuple to dominate other tuples with respect to the rest of the skyline. To this end, let $\text{DR}(t) = \{s \in \mathcal{D} \mid t \prec s\}$ denote the *dominance region* of tuple t, i.e., the set of all points in the domain $\mathcal{D}$ dominated by t. Analogously, for a set of tuples S, we extend this notion as $\text{DR}(S) = \bigcup_{t \in S} \text{DR}(t)$. Our proposal is then to consider the *exclusive dominance region* $\text{XR}(t;r)$ of a skyline tuple t, i.e., that part of the dominance region of the overall skyline that would not be there without t:

$$\text{XR}(t;r) = \text{DR}(\text{Sky}(r)) \setminus \text{DR}(\text{Sky}(r) \setminus \{t\}) \tag{4}$$

With this, we define our indicator, called *exclusive volume*, as the measure $\text{evol}(t;r)$ of the exclusive dominance region:

$$\text{evol}(t;r) = \text{Vol}(\text{XR}(t;r)), \tag{5}$$

where $\text{Vol}(\cdot)$ indicates the (hyper-)volume of a region. Intuitively, a high value of $\text{evol}(t;r)$ indicates that t is the only one to "cover" a large part of the domain, and thus neatly distinguishes itself from other skyline tuples.

Example 3.8. Figure 2c shows the dominance regions of the three skyline tuples. The gray regions are dominated by either t_1 or t_3, the hatched region is dominated by both and by t_2, the white part is non-dominated, while the pink region is $\text{XR}(t_2;r)$, dominated only by t_2. Its area is the exclusive volume of t_2 (here, $\text{evol}(t_2;r) = 0.042$, computed as $(t_3[1] - t_2[1]) \cdot (t_1[2] - t_2[2])$).

We observe that the exclusive volume of a tuple t is insensitive to dominated tuples, as immediately follows from Equations (4) and (5). As the previous indicators, evol is also scale-invariant.

Proposition 3.9. evol *is both stable and scale-invariant.*

3.4 Grid resistance

While evol considers a volume measure in its entirety, the indicator we introduce here accounts for the robustness of a skyline tuple on the single attributes, i.e., whether it would remain in the skyline after a slight perturbation of its attribute values. A practical way to measure this is via *value quantization*. Quantizing tuple values (i.e., snapping tuples to a grid divided in cells of equal size) may indeed affect dominance between tuples [16]. Typically, as the quantization step size grows, more tuple values tend to collapse, causing some skyline tuples that were non dominated by just a small margin to become dominated. The more a skyline tuple resists to larger quantization steps, the more it is robust.

We consider grids obtained by dividing each dimension in the same number (g) of equal-size intervals; each such grid has g^d cells and is henceforth referred to as a *g-grid*. The *grid-projection* $\text{gproj}(t, g)$ of a tuple t on a g-grid is defined as follows:

$$\text{gproj}(t, g) = \left\langle \frac{\lfloor t[1] \cdot g \rfloor}{g}, \ldots, \frac{\lfloor t[d] \cdot g \rfloor}{g} \right\rangle, \tag{6}$$

and corresponds to the lowest-value corner (in two dimensions: the lower-left corner) of the cell that contains t. We extend this notation to a relation r in the obvious way: $\text{gproj}(r, g) = \{\text{gproj}(t, g) \mid t \in r\}$. When assimilating tuples with their grid-projections, some new dominance relationships may occur.

Example 3.10. When using a 4-grid, as shown in Figure 2d, the grid-projection of t_2 (a skyline tuple, thus non-dominated) is dominated by that of t_1.

Intuitively, as the grid step increases (i.e., the lower the value of g), more distinct values tend to collapse to the only values available on the grid, which may make some skyline tuples leave the skyline. We define the *grid resistance* $\text{gres}(t; r)$ of a skyline tuple t as the smallest value of g^{-1} for which t is no longer in the skyline when considering the grid-projections (notice that, indeed, g^{-1} corresponds, in the case of $[0, 1]$-normalized domains, to the width of the intervals):

$$\text{gres}(t; r) = \min_{g \in \mathbb{N}} \{g^{-1} \mid \text{gproj}(t, g) \notin \text{Sky}(\text{gproj}(r, g))\} \tag{7}$$

Conventionally, we set $\text{gres}(t; r) = 1$ if t never exits the skyline.[4]

Example 3.11. As observed, tuple t_2 in Figure 2d has $\text{gres}(t_2; r) \leq 1/4$. More precisely, here $\text{gres}(t_2; r) = 1/6$, since $\text{gproj}(t_2, 6) = \langle \frac{1}{3}, \frac{1}{2} \rangle$ and $\text{gproj}(t_1, 6) = \langle 0, \frac{1}{2} \rangle$ and for no $g > 6$ do the grid projections of t_2 and t_1 have the same value on the A_1 axis (nor will any grid-projection of t_3 ever dominate that of t_2 for $g > 6$).

We observe that the grid resistance of a tuple t is insensitive to dominated tuples and scale-invariant.

Proposition 3.12. gres *is both stable and scale-invariant.*

3.5 Computing the indicators

Algorithms for computing brank_1 were given in [27], with a complexity that is essentially exponential in the number of dimensions d, but heavily affected by pruning and indexing. Computing brank_p, with $p > 1$, can be done in the same way on a transformed dataset r^p obtained by mapping each tuple $t \in r$ into a corresponding tuple t^p such that $t^p[i] = t[i]^p$, for $1 \leq i \leq d$.

A simple algorithm for computing $\text{cdeg}(t; r)$ for a skyline tuple t can proceed by first identifying an upper bound p^* for which Theorem 3.4 holds, and then starting a binary search for the minimum degree p such that t can be a top-1 result. To check this, due to stability, non-skyline tuples can be omitted. We then transform each tuple $s \in \text{Sky}(r)$ into a corresponding tuple s^p, as described before, thus obtaining a transformed skyline $\text{Sky}(r)^p$. With this, checking whether t is a possible top-1 result for a function in $\mathcal{L}_p$ amounts to checking its convexity in the transformed space, i.e., whether t^p belongs to the convex hull of $\text{Sky}(r)^p$. An optimal convex hull algorithm in any fixed number of dimensions d has a complexity of $O(n \log n + n^{d/2})$, where n is the dataset size, see [7]. In our case, the complexity can then be expressed as $O(Sd + \log(p^*)(S \log p^* + S \log S + S^{d/2}))$, where $S = |\text{Sky}(r)|$, $O(Sd)$ is the asymptotic complexity to find p^* (whose value is dataset-dependent and, a priori, unbounded), $\log p^*$ refers to the iterations of binary search, and $S \log p^*$ is the time required to transform the space. If p^* can be assimilated to a constant, the previous expression simplifies to $O(Sd + S \log S + S^{d/2})$.

Computing evol when $d = 2$ amounts to computing the area of a rectangle. This can be done for all skyline tuples by simply sorting them on A_1 and then computing $\text{evol}(t_i; r) = (t_{i+1}[1] - t_i[1]) \cdot (t_{i-1}[2] - t_i[2])$, where t_i is the i^{th} tuple in the sorted dataset (we conventionally set $t_0[2] = t_{N+1}[1] = 1$). The overall complexity for computing evol for all skyline tuples is thus $O(S \log S)$, where $S = |\text{Sky}(r)|$. When $d > 2$, the problem of computing evol (which amounts to the volume of the union of hyper-rectangles) can be cast as an instance of the *hypervolume contribution* problem, which is #P-hard when solved exactly, and NP-hard to approximate [4, 17]. A more pragmatic approach consists in using estimates via Monte Carlo sampling, with a convergence of $O(1/\sqrt{L})$, where L is the number of samples.

Finding gres requires recomputing dominance on grid-projected datasets for various values of g (grid intervals). Thanks to stability, only tuples in $\text{Sky}(r)$ need to be considered. Let ℓ be the absolute value of the smallest non-zero difference between any two tuples on the same attribute, i.e., $\ell = \min\{|t[i] - s[i]| \mid t, s \in \text{Sky}(r), t[i] \neq s[i], 1 \leq i \leq d\}$. Then, no new dominance relationship may occur when $g^{-1} < \ell$. With this, we can set an upper bound $\bar{g} = \lfloor \ell^{-1} \rfloor$ for the number of grid intervals and, for each $g \in \bar{g}..2$, compute the skyline and test membership. The complexity of computing gres for a given tuple is therefore $O(\bar{g} S^2)$, where $S = |\text{Sky}(r)|$ and $\bar{g}$ is, again, dataset-dependent and, a priori, unbounded.

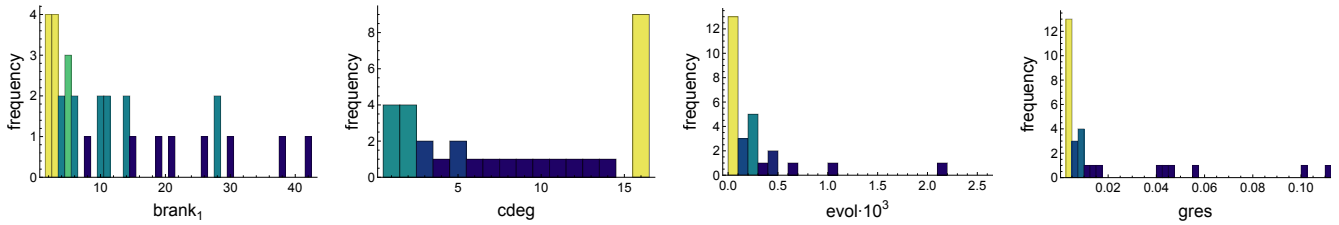
3.6 Using the indicators in practice

	N	d	convex fraction	brank_1	cdeg	evol $\cdot 10^4$	gres
$\text{ANT}_{2;5K}$	5K	2	4/31	11.3, 41	12.3, >15	23, 462	0.03, 0.20
NBA	4832	2	2/14	9.1, 27	8.6, >15	51, 269	0.09, 0.50
$\text{ANT}_{3;1M}$	1M	3	42/1022	154.8, >1K	>15, >15	0.2, 7	0.01, 0.14
$\text{ANT}_{6;1M}$	1M	6	195/99181	768.1, >1K	>15, >15	0.004, 2	0.02, 0.50
SEN	~2M	7	29/1496	352.2, >1K	>15, >15	0.1, 23	0.03, 0.50
RES	3.5M	6	22/8789	840.8, >1K	>15, >15	0.02, 54	0.01, 0.10

Table 2: Fraction of convex skyline tuples and indicator values (average and maximum) on datasets of various sizes (N) and dimensions (d)

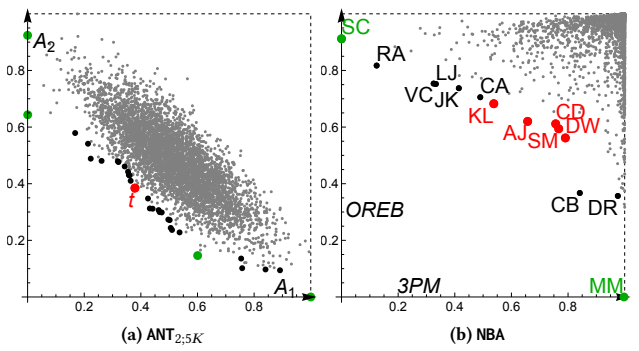
We start by observing that brank and cdeg refer to two distinct ways for tuning a top-k query so as to retrieve a given skyline tuple t: $\text{brank}(t; r)$ indicates the minimum result cardinality (k), while $\text{cdeg}(t; r)$ gives the minimum degree of non-linearity required for t to become top-1. Although the two measures are correlated (in particular, both must equal 1 for convex tuples), neither is redundant, and there are frequent occurrences of tuples with low brank and high cdeg or vice versa. Similarly, both evol and gres convey

[4] Indeed, the lowest value for g is 1 and, in that case, all grid-projections collapse to the origin and are thus non-dominated.



(a) **Histogram of** brank_1 (b) **Histogram of** cdeg (c) **Histogram of** evol (d) **Histogram of** gres

Figure 3: Indicator values for the skyline tuples of the $\mathsf{ANT}_{2;5K}$ **dataset.**



(a) $\mathsf{ANT}_{2;5K}$ (b) NBA

Figure 4: Example datasets and their skyline (larger dots).

a rough intuition about the "strength" of a skyline tuple, and are thus correlated, although, again, not redundant.

In order to root our discussion on practical cases, we have computed the indicators on 20 real and synthetic datasets of various sizes and distributions. In the interest of space, Table 2 shows just a small excerpt of our experimental study, but the observations that follow are common to all considered scenarios (datasets are fully described in Section 5). In particular, the table reports aggregate information about the indicators along with the incidence of convex skyline tuples. We remark that there is always a significant number of non-convex skyline tuples, with extremely high brank_1 values, i.e., impossible to find with a linear top-k query with a reasonable k (e.g., the average value of brank_1 in the RES dataset is > 800). Note that this is common to all datasets in Table 2 with large cardinality N and dimensionality d (last 4 rows), in each of which more than 95% of skyline tuples are concave, with very high values of brank_1 and cdeg, and many of them are robust.

We complement the numbers in Table 2 with an inspection of the two smallest 2D datasets, i.e., the synthetic $\mathsf{ANT}_{2;5K}$ and the real NBA, both easy to visualize along with their skylines. Figure 3 shows histograms for the value distributions of the four indicators when applied to the 31 skyline tuples of $\mathsf{ANT}_{2;5K}$. There are indeed several tuples that are "hard" to retrieve (for instance, $\mathsf{brank}_1 > 10$ and $\mathsf{cdeg} > 5$) but significantly robust. Figure 4a illustrates this on the plot of the entire dataset, where the dominated tuples are shown in gray, while the skyline tuples are shown as larger, non-gray dots. There are only 4 convex tuples (shown in green) out of 31 skyline tuples. However, several other tuples in the skyline might be interesting from a user's perspective and yet not be retrievable with any linear top-1 (or even top-10) query. For instance, the red tuple t in the figure has $\mathsf{brank}_1(t) = 14$ and $\mathsf{cdeg}(t) = 3$, so it is a hard point for top-k queries; however, it has the sixth best evol and the 8^{th} best gres values in the entire skyline. Moreover, t is a good compromise between the two attributes, much more balanced than the four available convex tuples.

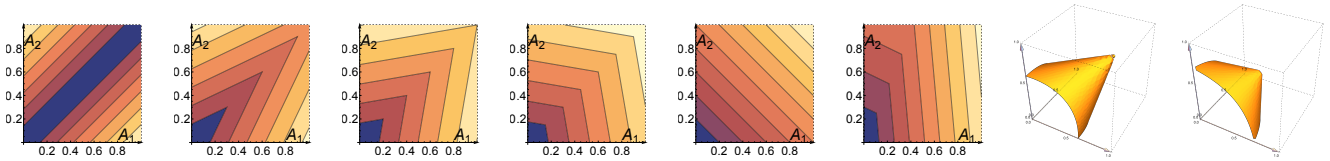
For an example of a real dataset with a small skyline, Figure 4b shows NBA,[5] which, due to its particular distribution and the limited number of distinct values on the two axes, has only 14 skyline tuples (2 of which convex). Yet, it also contains several harder but more balanced tuples. All tuples shown in red have $\mathsf{brank}_1 > 10$ and $\mathsf{cdeg} \geq 3$, but non-negligible evol and gres values. In particular, tuple AJ has the fourth largest gres and evol values in the skyline.

4 DIRECTIONAL QUERIES

From the results in the previous section we know that many skyline tuples are hard to retrieve with linear top-k queries (high values of the brank_1 indicator), yet they are potentially very interesting, as measured with our evol and gres indicators. A possible way around this shortcoming would be to use a different family of functions than $\mathcal{L}_1$, thereby moving to non-linear queries. As was pointed out in Theorem 3.4, a suitable value of p always exists such that any skyline tuple is also the top-1 result for some function in $\mathcal{L}_p$. However, the cdeg values shown in Table 2, and the brank_p values (not shown in the table) that are obtained with low values of p (e.g., brank_2 and brank_3 are both > 700, on average, in RES) show that much higher values of p would be required to easily retrieve many concave skyline tuples. Besides adding an excessive cognitive burden for the users, who would also have to come up with a suitable value for p, high values of p can lead to two undesirable side-effects. First, the higher p is, the less preferences can influence the result. This directly follows from the observation that, when p tends to infinity, any $\mathcal{L}_p$ function $f(t)$ tends to $\max\{t[1], \ldots, t[d]\}$, *regardless* of the weight vector. Second, ranking of tuples might be in contrast with weights' values, as the following example shows.

Example 4.1. Consider a dataset with 3 tuples $t_1 = \langle 0.6, 0.2\rangle$, $t_2 = \langle 0.2, 0.8\rangle$, $t_3 = \langle 0.5, 0.75\rangle$, and the weight vector $w = \langle 0.85, 0.15\rangle$. With a linear function, t_2 would be a clear winner, because of the high value of the weight on the first attribute and of the best value

[5] Normalized stats for "3 point field goals made" (*3PM*) and "offensive rebounds" (*OREB*). Players in the skyline: S. Curry, R. Allen, V. Carter, L. James, J. Kidd, C. Anthony, K. Love, A. Jamison, C. Drexler, S. Marion, D. Wilkins, C. Barkley, D. Rodman, M. Malone.



(a) $0; \langle \frac{1}{2}, \frac{1}{2} \rangle$ (b) $\frac{1}{3}; \langle \frac{1}{2}, \frac{1}{2} \rangle$ (c) $\frac{1}{2}; \langle \frac{1}{2}, \frac{1}{2} \rangle$ (d) $\frac{2}{3}; \langle \frac{1}{2}, \frac{1}{2} \rangle$ (e) $1; \langle \frac{1}{2}, \frac{1}{2} \rangle$ (f) $\frac{2}{3}; \langle \frac{7}{10}, \frac{3}{10} \rangle$ (g) $\frac{1}{2}; \langle \frac{1}{3}, \frac{1}{3}, \frac{1}{3} \rangle$ (h) $\frac{2}{3}; \langle \frac{1}{3}, \frac{1}{3}, \frac{1}{3} \rangle$

Figure 5: Shape of the directional query in 2D and 3D for several combinations of $\beta; \langle w_1, \ldots, w_d \rangle$.

that t_2 has on it. However, for all $p \geq 7$, t_2 would be the *worst* tuple, which is quite counter-intuitive.

We have already observed that linear top-k queries may fail to discover tuples that are well *balanced* across the various dimensions when the dataset contains tuples that are extremely good in one attribute but poor in the others. In order to give the user the opportunity to retrieve also these hard-to-find results, at the same time avoiding the above-discussed negative effects of non-linear scoring functions of high polynomial degree, we propose to enrich (linear) scoring functions with a term explicitly favoring balanced results. The key observation is that, when a user specifies the weights, they already include an indication of the relative importance of the attributes, and, thus, what parts of the data space may contain the tuples corresponding to the best compromise of the different attributes. Thus, with no need of introducing new parameters, the weight vector itself may be used to determine a direction, that we call *preference line*, in the data space: the closer to the preference line, the better a tuple matches the original user's intentions.

Example 4.2. Going back to the NBA dataset, shown in Figure 4a, a user indicating equal importance for the two attributes (i.e., a weight vector $w = \langle 0.5, 0.5 \rangle$) would obtain, with a linear top-k query, very unbalanced results first. In particular, S. Curry (best *3PM* but almost worst *OREB*) ranks first, and M. Malone (opposite situation) ranks third; although overall excellent, these players may not express the most balanced combination of *3PM* and *OREB*, as suggested by w. Possibly better compromises would be expressed, e.g., by the more balanced (and much closer to the preference line) K. Love and A. Jamison, who however are ranked 13^{th} and, resp., 23^{rd} by the same linear top-k query.

When all weights are equal, it is natural to assume that the preference line coincides with the main diagonal of the data space, i.e., the locus of points where all attribute values are equal. A consistent generalization to the case of unequal weights is to consider the requirement of *weighted balance*, which can be expressed by stating that *weighted* attribute values should now be equal, i.e., for any point t on the preference line, we should have $w_i t[i] = w_j t[j]$ for all i, j. This immediately leads to defining the *preference line* $\text{PL}(w)$ corresponding to the weight vector $w = \langle w_1, \ldots, w_d \rangle$ as the set:

$$\text{PL}(w) = \{\langle \bar{w}_1 x, \ldots, \bar{w}_d x \rangle \mid x \geq 0\},$$

where $\bar{w}_i = 1/w_i$ for $1 \leq i \leq d$.

In order to determine how much a tuple t is balanced, we compute its (Euclidean) distance, $\textsc{Dist}(t, \text{PL}(w))$, from $\text{PL}(w)$, which requires standard geometric techniques. First observe that the value x_0 of parameter x such that the distance of t from $\langle \bar{w}_1 x, \ldots, \bar{w}_d x \rangle$ is minimum is $x_0 = \frac{\sum_{j=1}^{d} \bar{w}_j t[j]}{\sum_{j=1}^{d} \bar{w}_j^2}$, as can be obtained by deriving with respect to x the distance between t and $\langle \bar{w}_1 x, \ldots, \bar{w}_d x \rangle$ and setting it to zero. Then we have: $\textsc{Dist}(t, \text{PL}(w)) = \sqrt{\sum_{i=1}^{d} \left(t[i] - \bar{w}_i \frac{\sum_{j=1}^{d} \bar{w}_j t[j]}{\sum_{j=1}^{d} \bar{w}_j^2} \right)^2}$.

Clearly, being close to the preference line is not sufficient to characterize a good tuple. Indeed, a tuple such as $\langle 1, 1 \rangle$ lies on the preference line $\text{PL}(\langle 0.5, 0.5 \rangle)$, but can hardly be preferred to any other tuple at all! Being as close as possible to the origin (the target point) continues to matter.

With all this in mind, we are now ready to define a new kind of query that overcomes all the aforementioned limitations. In particular, *i)* we do not overburden the users with new specification requirements, *ii)* we allow them to obtain any skyline tuple (not just the convex ones) as a top-1 result, and *iii)* we preferentially select those tuples that are close to the origin and whose balance across different attributes best corresponds to the preference line determined by the weight vector.

Our proposal, called *directional query*, is a top-k query whose scoring function combines two components: a weighted mean and a distance. The mean component is a weighted sum of the attribute values through a weight vector w, i.e., a function in $\mathcal{L}_1$ using w. The distance component is the distance from the preference line determined by w. For both, then, we just need exactly the same specification as a linear top-k query: a weight vector w. The family $\textsc{Dir}$ of scoring functions of directional queries is then defined as:

$$\textsc{Dir} = \left\{ f \mid f(t) = \beta \sum_{i=1}^{d} w_i t[i] + (1 - \beta)\, \textsc{Dist}(t, \text{PL}(w)) \right\}, \quad (8)$$

where $\beta \in [0, 1]$ is a parameter that expresses how to combine the two components. While the weight vector determines the preference line, changing β causes the query to range between the two extreme cases of a purely linear and a purely distance-based top-k query.[6] In more geometric terms, directional queries have query fronts shaped like an "arrowhead", whose width is determined by β (the larger the value of β, the wider the shape). Intuitively, lowering β would increase the chance of tuples close to the preference line, thus with more balanced attribute values, to be ranked better.

Figure 5 shows the iso-score curves for several combinations of β and w, where darker colors indicate better (lower) scores. When $\beta = 0$ only the distance from the preference line matters (Figure 5a), whereas only the (weighted) mean matters when $\beta = 1$ (Figure 5e). The weight vector determines the slope of the preference line, as

[6] Thus, directional queries are a generalization of standard linear top-k queries, since, when $\beta = 1$, we have a function in $\mathcal{L}_1$.

can be seen in Figure 5f for $w_1 = 0.7$. When $d \geq 3$ the shape of the query front of the directional query is that of a cone (see Figures 5g and 5h for examples in 3D with just one score value).

The β parameter is not intended to be user-specified; in Section 5.1 we will analyze its effect on the queries and how to conveniently set it.

Example 4.3. In the dataset shown in Figure 2, if the user chooses a scoring function $f \in$ Dir with a balanced weight vector such as $w = \langle 0.5, 0.5 \rangle$, a more balanced tuple like t_2 may be preferred to t_1 or t_3 (which are farther from PL(w)). This happens, e.g., with $\beta = 0.5$, since $f(t_2) = 0.27 < f(t_1) = f(t_3) = 0.38$.

The following result states that for any skyline tuple there is a directional query that ranks it as top-1 with no tie.

Theorem 4.4. *Let* $t \in$ Sky(r). *Then* $\exists f \in$ Dir *such that* $\mathsf{rank}(t, r; f) = 1$ *and,* $\forall s \in r, s \neq t \rightarrow \mathsf{rank}(s, r; f) > 1$.

For readabilty, we shall henceforth use Dir to refer to directional queries and Lin for linear queries.

4.1 Algorithms

The sequential evaluation of a Dir query adopts a standard, heap-based, implementation, which guarantees a worst-case complexity of $O(N \log k)$ for a dataset with N tuples.

Besides sequential processing, we also consider the case in which the dataset is indexed through an R-tree [18]. We adopt the branch-and-bound algorithmic pattern described in [2] and [32], which guarantees that top-k queries can be processed so as to minimize the number of visited nodes (*I/O optimality*), provided that for each node of the tree one is able to determine a tight bound on the best (i.e., lowest, in our scenario) score obtainable from a tuple reachable from that node. For a Lin query with weight vector w and a node with opposite vertices $\langle l_1, \ldots, l_d \rangle$ and $\langle h_1, \ldots, h_d \rangle$, with $\forall i.\, l_i < h_i$, this lower bound clearly equals $\sum_{i=1}^{d} w_i l_i$. For a Dir query, this bound is only loose, since it does not consider the component of the distance from the preference line. Nonetheless, a tight lower bound can be determined by minimizing (8), which we do through a non-linear programming solver [21]. In Section 5.2 we will contrast the performance of Dir queries when the search adopts either a loose or a tight bounding strategy, with the former cheaper to compute but expected to require more node accesses.

Algorithm 1 provides a simplified description of the branch-and-bound logic, which is the same for Lin and Dir queries. Nodes of the tree are visited in increasing order of the best possible score, *best*, that can be obtained from a tuple reachable from the node, as computed by the Bound function (line (9)). Note that this and the computation of a tuple score $f(t)$ (line (7)) are the only parts in which Lin and Dir queries differ. Entries corresponding to nodes to be explored are maintained in a priority queue *PQ*, ordered by increasing *best* values. If the first entry of *PQ* is a leaf node, the scores of its tuples are computed and the result set *RES* (organized as a max-heap) possibly updated. For an intermediate node, the entry of a child node is added to the queue only if it can improve the result. The algorithm stops when the best entry popped from *PQ* is no better than the k-th (i.e., worst) score value in *RES*. For simplicity of pseudocode, we assume that $RES[k].score = +\infty$ if *RES* has less than k elements.

Algorithm 1: R-tree processing of linear and directional top-k queries.

```
Input:  scoring function f, root node R of R-tree.
Output: heap RES with the top-k tuples according to f.
(1)  PQ := ∅ // queue storing nodes and their best scores
(2)  RES := ∅ // empty heap of max size k
(3)  node := R; best := 0
(4)  while RES[k].score > best do
(5)    if isLeaf(node) then
(6)      for each t in node
(7)        if f(t) < RES[k].score then RES.Insert(⟨t, f(t)⟩)
(8)    else for each child node C of node
(9)        best := Bound(C; f)
(10)       if best < RES[k].score then PQ.Push(⟨C, best⟩)
(11)   ⟨node, best⟩ := PQ.Pop // retrieves node and loads it
(12) return RES
```

5 EXPERIMENTS

In this section we aim to assess the effectiveness and efficiency of Dir queries and compare them to Lin as well as other kinds of top-k queries. We consider a number of different scenarios, and study the effect of *i)* data distribution, *ii)* dataset size, *iii)* number of dimensions, and *iv)* output size. For our study on effectiveness, we also consider different values for β in directional queries. The relevant parameters are shown in Table 3, with defaults in bold.

Table 3: Operating parameters (defaults in bold).

Full name	Tested value
Distribution	synthetic: ANT; real: NBA, HOU, EMP, RES, SEN
Synthetic dataset size (N)	5K, 10K, 50K, 100K, 500K, **1M**, 5M, 10M
# of dimensions (d)	2, **3**, 4, 5, 6
Output size (k)	1, 5, **10**, 50, 100
Mean-distance trade-off (β)	1/3, 1/2, **2/3**, 1

Datasets. We use both real and synthetic datasets. The real datasets we use, often adopted in the context of skylines, include after cleaning, normalization and attribute selection: NBA – all-time stats for 4832 NBA players from nba.com as of Oct. 2023; HOU – 127,931 6D tuples regarding household data scraped from www.ipums.org; EMP – 291,825 6D tuples about City employees in San Francisco [30]; RES – real estate data from zillow.com, with 3,569,678 6D tuples; SEN – sensor data with 7 numeric attributes and 2,049,280 tuples [19].

For synthetic datasets, we refer to the generator in [3] to produce, for any value of d and N mentioned in Table 3, a d-dimensional dataset (ANT) of size N with values in the $[0, 1]$ interval, anti-correlated across different dimensions. In our experiments, we have considered several other synthetic distributions (uniform, correlated, multi-variate, exponential, normal) as well as real datasets. However, we only focus on the main findings and refrain from showing the entirety of our results due to space constraints.

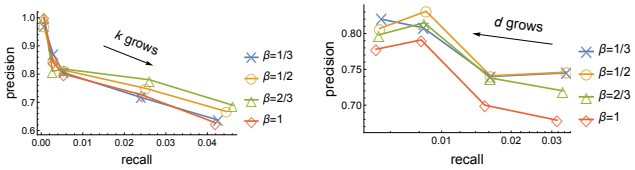
When ambiguity may arise, we indicate the number of dimensions and size as subscripts (e.g., $\mathsf{ANT}_{2;5K}$).

5.1 Experiments on effectiveness

Here, we aim to assess the effectiveness of Dir queries in retrieving relevant results. To this end, due to space constraints, we focus on

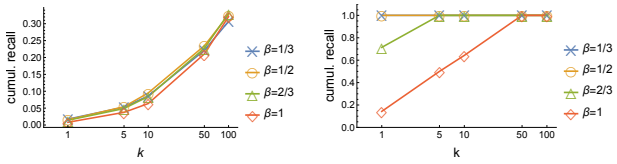
	linear ($\beta = 1$)				directional ($\beta = 2/3$)				directional ($\beta = 1/2$)				directional ($\beta = 1/3$)			
#	Name	*3PM*	*OREB*	DIST	Name	*3PM*	*OREB*	DIST	Name	*3PM*	*OREB*	DIST	Name	*3PM*	*OREB*	DIST
1	*S. Curry	0.00	0.91	0.64	*A. Jamison	0.66	0.62	0.03	*A. Jamison	0.66	0.62	0.03	*A. Jamison	0.66	0.62	0.03
2	*R. Allen	0.12	0.82	0.49	*K. Love	0.54	0.68	0.10	*K. Love	0.54	0.68	0.10	S. Pippen	0.71	0.69	0.01
3	*M. Malone	1.00	0.00	0.71	*C. Anthony	0.49	0.71	0.15	S. Pippen	0.71	0.69	0.01	D. Marshall	0.73	0.72	0.01
4	J. Harden	0.19	0.88	0.49	*J. Kidd	0.41	0.74	0.23	D. Marshall	0.73	0.72	0.01	R. Horry	0.77	0.76	0.01
5	*V. Carter	0.33	0.75	0.30	*L. James	0.33	0.75	0.30	C. Robinson	0.63	0.72	0.07	K. Towns	0.75	0.78	0.02
6	*L. James	0.33	0.75	0.30	*V. Carter	0.33	0.75	0.30	A. Walker	0.59	0.72	0.09	C. Robinson	0.63	0.72	0.07
7	R. Miller	0.25	0.87	0.44	A. Walker	0.59	0.72	0.09	*C. Anthony	0.49	0.71	0.15	*K. Love	0.54	0.68	0.10
8	*J. Kidd	0.41	0.74	0.23	S. Pippen	0.71	0.69	0.01	R. Westbrook	0.64	0.74	0.07	B. Lopez	0.77	0.71	0.04
9	*C. Anthony	0.49	0.71	0.15	C. Robinson	0.63	0.72	0.07	R. Horry	0.77	0.76	0.01	R. Rogers	0.80	0.81	0.01
10	D. Nowitzki	0.42	0.78	0.26	*R. Allen	0.12	0.82	0.49	S. Perkins	0.75	0.65	0.07	R. Westbrook	0.64	0.74	0.07

Table 4: Results of top-10 queries on NBA with $w = \langle 0.5, 0.5\rangle$ and distance DIST from PL(w). Starred players are in SKY(NBA).



(a) PRE/REC on SEN as k varies **(b) PRE/REC on SEN as d varies**

Figure 6: Precision/recall curves on SEN.



(a) $\widehat{\text{REC}}$ on SEN as k varies **(b) $\widehat{\text{REC}}$ on NBA as k varies**

Figure 7: Cumulative recall $\widehat{\text{REC}}$ on SEN and NBA.

the real SEN and NBA, and synthetic ANT datasets, but the results we show extend to the other datasets introduced above.

In order to provide an intuition on how LIN and DIR queries differ in ranking tuples, we consider again the NBA dataset shown in Figure 4b and run on it both a LIN and three DIR top-10 queries with $w = \langle 0.5, 0.5\rangle$ and different values for β. The results of these queries are shown in Table 4 along with their distance from PL(w). The obtained tuples are much more balanced in the case of DIR queries, and even more so for lower values of β. Indeed, unlike the LIN query, no DIR query includes in its top-10 results either S. Curry or M. Malone, whose scores are extremely unbalanced with respect to PL(w), whereas the LIN query omits the very balanced skyline tuples A. Jamison and K. Love, which are in the top-10 of the DIR queries. We complement this qualitative analysis on the NBA dataset by reporting the results of a simple user study we conducted. We asked graduate students to compare the top-5 results of LIN and DIR queries (graphically shown in Figure 16a) and to determine which result set was better when the task was to retrieve the best players if we give the same importance to offensive rebounds (*OREB*) and 3-point field goals made (*3PM*). Out of around 44 answers, 29 preferred the results delivered by the DIR query and only 15 those of the LIN query.

Precision and recall. We now try to assess the ability of LIN and DIR queries to retrieve relevant (skyline) tuples. To this end, we start with classical precision and recall measures. Formally, let $\text{ST}_k(r; f) = \text{SKY}(r) \cap \text{TOP}_k(r; f)$. The *precision* $\text{PRE}(k, f)$ and *recall* $\text{REC}(k, f)$, for natural k and scoring function f, measure how many top-k tuples (found by f in r) are also in $\text{SKY}(r)$ and, respectively, how many tuples in $\text{SKY}(r)$ are also top-k tuples:

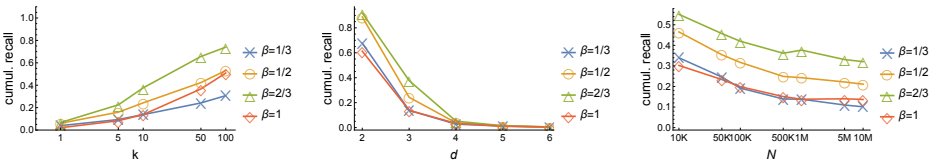
$$\text{PRE}(k, f) = \frac{|\text{ST}_k(r; f)|}{k}; \quad \text{REC}(k, f) = \frac{|\text{ST}_k(r; f)|}{|\text{SKY}(r)|}$$

In Figure 6, we show average precision and recall values as measured over a set of $q = 100$ queries with random weight vectors on the SEN dataset. Generally, as a consequence of their definitions, precision decreases as k grows, while recall increases (Figure 6a). Precision generally improves for larger values of d, since it is easier for a tuple to be in the skyline in higher dimensions, but the increased cardinality of the skyline also determines a heavy decrease in recall. Figure 6b shows this for $d = 4..7$ (we omit $d = 2, 3$ as the skyline of SEN consists of less than 5 tuples). In both figures, LIN queries ($\beta = 1$) have the worst precision for almost any recall level.

A major motivation underlying the introduction of directional queries is to allow more relevant tuples to be retrievable by top-k queries, thus increasing the effectiveness of preferences in obtaining different results. To this end, we introduce a measure of *cumulative recall*, $\widehat{\text{REC}}(k, \mathcal{F})$, where $\mathcal{F}$ is a set of scoring functions, defined as the fraction of skyline tuples *collectively* retrieved by the functions in $\mathcal{F}$. This gives us an indication of how many tuples in the skyline have a chance to appear in the result of at least one top-k query. Let us denote with $\text{ST}_k(r; \mathcal{F})$ the set $\cup_{f \in \mathcal{F}} \text{ST}_k(r; f)$. We define:

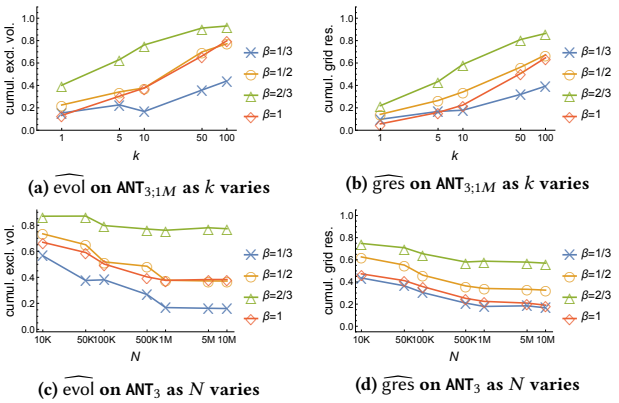
$$\widehat{\text{REC}}(k, \mathcal{F}) = \frac{|\text{ST}_k(r; \mathcal{F})|}{|\text{SKY}(r)|}$$

Figure 7a shows the cumulative recall $\widehat{\text{REC}}$ for $q = 100$ random top-k queries on the SEN dataset, which appears to be slightly worse in the case of LIN queries. Instead, in the small NBA dataset (Figure 7b), 100 top-k queries with $k = 5$ are sufficient to retrieve the full skyline when $\beta < 1$, whereas LIN queries require $k = 50$ to do so, even though $|\text{SKY}(\text{NBA})|$ is just 14.



(a) $\widehat{\text{REC}}$ on $\text{ANT}_{3;1M}$ as k varies (b) $\widehat{\text{REC}}$ on ANT_{1M} as d varies (c) $\widehat{\text{REC}}$ on ANT_3 as N varies

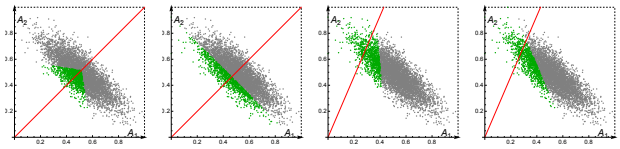
Figure 8: Cumulative recall $\widehat{\text{REC}}$ on ANT.



(a) $\widehat{\text{evol}}$ on $\text{ANT}_{3;1M}$ as k varies (b) $\widehat{\text{gres}}$ on $\text{ANT}_{3;1M}$ as k varies

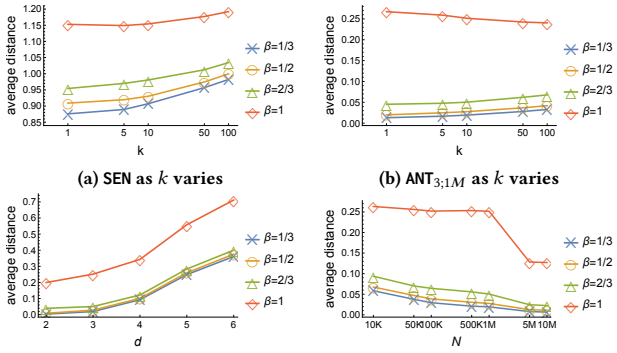
(c) $\widehat{\text{evol}}$ on ANT_3 as N varies (d) $\widehat{\text{gres}}$ on ANT_3 as N varies

Figure 9: Exclusive volume $\widehat{\text{evol}}$ and grid resistance $\widehat{\text{gres}}$.



(a) $w = \langle 0.5, 0.5\rangle$, $\beta = \frac{2}{3}$ (b) $w = \langle 0.5, 0.5\rangle$, $\beta = 1$ (c) $w = \langle 0.7, 0.3\rangle$, $\beta = \frac{2}{3}$ (d) $w = \langle 0.7, 0.3\rangle$, $\beta = 1$

Figure 10: Tuples retrieved by directional ($\beta = 2/3$) and linear ($\beta = 1$) top-k queries on $\text{ANT}_{2;5K}$ with $k = 1K$.



(a) SEN as k varies (b) $\text{ANT}_{3;1M}$ as k varies

(c) ANT_{1M} as d varies (d) ANT_3 as N varies

Figure 11: Average distance from the preference line.

For synthetic datasets, Figure 8a shows that, for $\text{ANT}_{3;1M}$, the value $\beta = 2/3$ entails a $\widehat{\text{REC}}$ consistently larger than the one obtained with other values, reaching almost 80% of the skyline (1022 tuples). The relative performance of the different values of β is confirmed also on ANT_{1M} as d varies (Figure 8b) and on ANT_3 as N varies (Figure 8c), with $k = 10$. The steep growth in the number of skyline tuples as d grows determines a heavy decrease in REC, which also causes $\widehat{\text{REC}}$ to decrease and the impact of β to be less visible. An increased size also causes a larger skyline, which, in turn, determines a decrease in REC and $\widehat{\text{REC}}$. LIN queries are almost always the worst choice.

Robustness indicators. In a similar fashion, we consider the cumulative versions $\widehat{\text{evol}}$ and $\widehat{\text{gres}}$ of our indicators of robustness, so as to characterize the portion of the overall exclusive volume that is covered by the extracted tuples (and similarly for grid resistance):

$$\widehat{\text{evol}}(k, \mathcal{F}) = \frac{\sum_{t \in \text{ST}_k(r,\mathcal{F})} \text{evol}(t; r)}{\sum_{t \in \text{SKY}(r)} \text{evol}(t; r)}$$

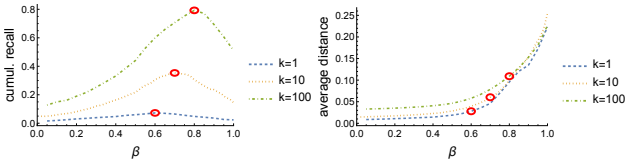
$$\widehat{\text{gres}}(k, \mathcal{F}) = \frac{\sum_{t \in \text{ST}_k(r,\mathcal{F})} \text{gres}(t; r)}{\sum_{t \in \text{SKY}(r)} \text{gres}(t; r)}$$

Very similar trends are observed when studying how the cumulative exclusive volume $\widehat{\text{evol}}$ (first column of Figure 9) and grid resistance $\widehat{\text{gres}}$ (second column) vary, for which $\widehat{\text{REC}}$ is a good proxy.

Balance. The distance $\text{DIST}(t, \text{PL}(w))$ of a tuple t from the preference line gives an indication of how balanced t is with respect to the user requirements. To better appreciate how results are spatially distributed, Figure 10 shows the top-1000 tuples (in green) obtained with various weight vectors on the $\text{ANT}_{2;5K}$ dataset with a DIR query ($\beta = 2/3$) and LIN query ($\beta = 1$), along with the preference line (in red). It is apparent that the results are closer to the preference line with DIR queries, and more dispersed with LIN queries, especially for balanced weight vectors.

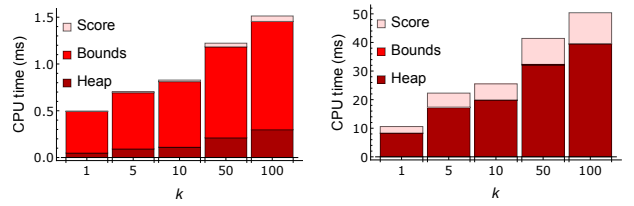
Figure 11 confirms this behavior both on the real SEN and the synthetic ANT datasets. As a general trend, the average distance grows as k grows (later tuples in the ranking are likely farther from $\text{PL}(w)$, Figures 11a and 11b) or as d grows (tuples are more sparse, Figure 11c), while it slightly decreases as N grows (tuples are more dense, Figure 11d). Clearly, the lower the value of β, the lower the average distance, but, as shown in the figure, all considered values of $\beta < 1$ yield comparable performance, whereas LIN queries show a much poorer behavior.

Another indication of the effectiveness of a query with weight vector w in retrieving relevant results while abiding by the preferences specified by w can be obtained by considering how the skyline tuple closest to $\text{PL}(w)$ is ranked by the query. Here, the difference between LIN and DIR queries is very substantial: for example, with $q = 100$ queries on $\text{ANT}_{3;1M}$, we found that the median



(a) $\widehat{\text{REC}}$ as β varies, $k \in \{1, 10, 100\}$ (b) Avg DIST as β varies, $k \in \{1, 10, 100\}$

Figure 12: Effect of β on $\text{ANT}_{3;1M}$; β^* values circled.



(a) CPU time: TIGHT strategy (b) CPU time: LOOSE strategy

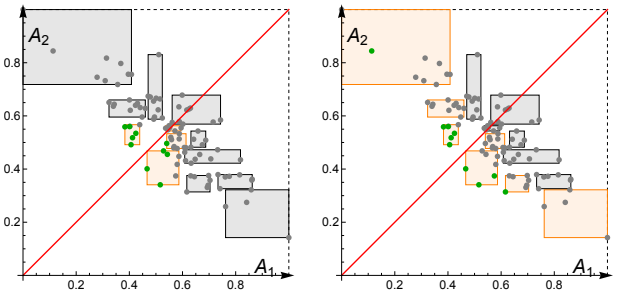
Figure 13: R-trees: CPU times for directional queries on $\text{ANT}_{3;1M}$ as k varies using two different bound strategies.

rank of the skyline tuple closest to $\text{PL}(w)$ was 1 for $\beta \in \{1/3, 1/2\}$, 2 for $\beta = 2/3$, and 4724 for $\beta = 1$; even the average rank of the closest skyline tuple was a shockingly high 8330 with $\beta = 1$, but only 67 for $\beta = 2/3$.

Choice of β. Before moving to our experiments on efficiency (Section 5.2), we now discuss how to choose an appropriate value for β. To this end, we probe the dataset at hand with various values of β, checking how they affect $\widehat{\text{REC}}$ and average DIST; the former accounts for the ability of directional queries to discover relevant results and is also a good proxy for the robustness indicators, while the latter measures the balance of results with respect to user preferences. Figure 12 reports, for $k \in \{1, 10, 100\}$, the corresponding plots considering $q = 100$ weight vectors on the $\text{ANT}_{3;1M}$ dataset. We observe that $\widehat{\text{REC}}$ is maximized for a value $\beta = \beta^*$ in $[0.6, 0.8]$ (circled in red) and that, for the same β^*, the average distance from the preference line starts increasing more steeply. This suggests that a value close to β^* might be an adequate choice for β. This analysis can be easily performed, statically, for any dataset. We did this on all scenarios described in Table 3 and found that β^* always lies in $[0.6, 0.8]$. In particular, choosing $\beta = 0.7$ consistently determines at least a 64% improvement in terms of $\widehat{\text{REC}}$ with respect to the case $\beta = 1$, while being at most 10% off the maximum $\widehat{\text{REC}}$ value obtained with *any* β. Additionally, the choice $\beta = 0.7$ grants an improvement of at least 79% in terms of average DIST with respect to $\beta = 1$ in all tested synthetic scenarios. For these reasons, we shall henceforth set $\beta = 0.7$ in our experiments on efficiency.

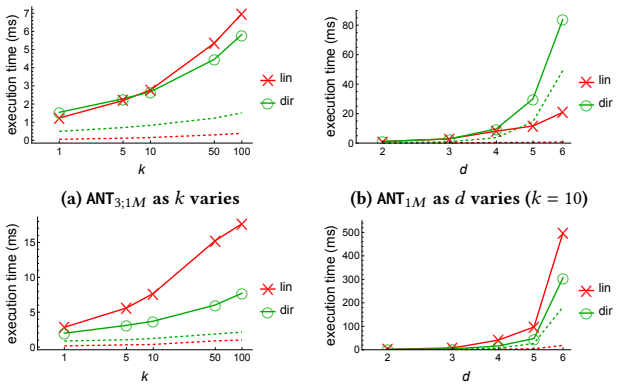
5.2 Experiments on efficiency

In this section we evaluate the performance of DIR queries from the point of view of efficiency, by contrasting it to classical LIN top-k queries. All reported times are averaged out over 10 executions (and, in the case of synthetic datasets, on different instances of the same dataset type) using a 1.80GHz 4-core Intel processor with 16 GB of RAM. We report execution (CPU) times only, since all our datasets fit into main memory. For the experiments with R-trees, we adopted the implementation in [1] and extended it to support top-k queries.



(a) Directional query (b) Linear query

Figure 14: R-trees: accessed leaves (in orange) for directional and linear top-k queries with $w = \langle 0.5, 0.5 \rangle$, with $\text{PL}(w)$ in red. The green dots are the results of the query.



(a) $\text{ANT}_{3;1M}$ as k varies (b) ANT_{1M} as d varies ($k = 10$)

(c) $\text{ANT}_{3;1M}$ as k varies (d) ANT_{1M} as d varies ($k = 10$)

Figure 15: R-trees: total execution time on ANT – all weights (first row) vs balanced weights (second row).

Results on a sequential evaluation of top-k queries (not shown here due to lack of space) confirm the expected $O(N \log k)$ complexity of such an implementation, with the effect of k hardly noticeable in larger datasets such as SEN and $\text{ANT}_{3;1M}$. In all tested scenarios, DIR queries incur a slight computational overhead (about 15% more CPU time than LIN queries), due to the increased cost required for computing the score through non-linear function (8). Yet, CPU times are largely sub-second in all configurations up to 1M tuples.

Similar observations hold with NBA, HOU, EMP, and RES. With default parameter values and varying k, on EMP we obtained execution times (in ms) in the $[89, 131]$ range with DIR queries and in $[78, 115]$ with LIN queries; on HOU, the ranges were $[39, 75]$ and $[33, 64]$; on RES $[1158, 1262]$ and $[895, 968]$; on NBA, all times were comparable and always < 10ms.

Figure 13 reports the CPU time spent on the $\text{ANT}_{3;1M}$ dataset as k varies, for the two bounding strategies described in Section 4.1 that DIR queries can exploit when using R-trees. Results strongly suggest that it always pays off to compute a tight bound. Indeed,

the total CPU time with the LOOSE strategy is 21 to 35 times higher than with TIGHT (although largely sub-second in all cases). The figure also shows the composition of these times and highlights that, while, with TIGHT, the largest part is the bound computation ("Bounds", Figure 13a), the LOOSE strategy requires virtually no time for bound computation but needs to deal with many more tuples (needlessly accessed due to non-I/O-optimality), and spends most of the time for maintaining its heap (and other internal structures) and computing scores ("Heap" and "Score", Figure 13b). For instance, when $k = 10$, TIGHT accesses, on average, 19 nodes and 409 tuples, whereas LOOSE accesses 2925 nodes and 137,035 tuples. We shall therefore only focus on the TIGHT strategy for DIR queries and compare their execution times with that of LIN queries. We observe that, with R-trees, efficiency improves on average by more than two orders of magnitude with respect to the sequential implementation (for instance, less than 1ms vs more than 300ms on $\mathsf{ANT}_{3;1M}$ and $k = 10$).

For an intuition of how R-trees work with the two query types, Figure 14 offers a visualization of the leaves of the R-tree for an ANT_2 dataset with just $N = 100$ tuples. When the weights are perfectly balanced, in order to find the top 10 tuples, the DIR query (Figure 14a) only accesses the four leaves (shown in orange) closest to the preference line and to the origin, and in total 28 tuples. The LIN query (Figure 14b) accesses, instead, 7 leaves and 52 tuples.

Figure 15 reports the overall execution times spent by top-k queries on the ANT dataset. Besides CPU times (shown as dotted lines), the figure also accounts for the time spent for accessing the R-tree nodes, where we assume that each node access requires approximately 0.1ms. This has the purpose of highlighting the differences between the two query types in terms of node accesses, as also exemplified in Figure 14. In lower dimensionalities, DIR queries keep paying a small overhead in terms of increased CPU time, due to the heavier calculation of the lower bound, but benefit from a reduced number of accesses to nodes. Such benefits become more visible as k increases (Figure 15a); for instance, when $k = 100$ in $\mathsf{ANT}_{3;1M}$, the DIR query accesses 44 nodes on average, while the LIN query requires 68 nodes, which more than compensates the difference in CPU time. This phenomenon is generally also present in the real datasets. We also observe (Figure 15b) that d negatively affects the relative performance of DIR queries, in that the non-linear problem becomes more challenging (indeed, the DIR query spends 47ms computing the bound out of a total of 49ms CPU time, when $d = 6$ and $k = 10$ on ANT_{1M}). Although the overall times are always largely sub-second in all tested scenarios with R-trees, we also observe that the focus of DIR queries on the preference line more clearly distinguishes it from the LIN query when the weight vector is balanced. Figures 15c and 15d show the same experiments as Figures 15a and 15b, respectively, but with weights at most 20% off the (perfectly balanced) value $1/d$. With these weights, although the overall times increase since more tuples are accessed by both queries, the advantage of DIR queries becomes more evident, and even in the most adverse scenarios ($d = 6$, $N = 1M$, $k = 10$) the higher CPU times are compensated by the difference in accessed tuples and nodes (15,799 out of 37,840 nodes accessed by the LIN query vs only 4845 by the DIR query).

Finally, we also experimented with the threshold algorithm (TA) [13], for which it is assumed that all attributes of interest can be independently accessed in an ordered way (i.e., by having d ordered lists or d single-attribute indices). Scanning the lists can be stopped when one has retrieved k tuples whose score is better than the *threshold* value T, which is a bound on the best (i.e., lowest) score of the yet-to-be-seen tuples. Let l_i be the last value seen on the i-th list. Then, $T = \sum_{i=1}^{d} w_i l_i$ for LIN queries, as well as for the LOOSE bounding strategy of DIR queries, whereas for a TIGHT bounding strategy one can use the same optimization approach adopted for R-trees.[7] Experiments on $\mathsf{ANT}_{3;1M}$ confirm the relative behavior observed with R-trees, in that DIR queries pay a moderate (around 10%) overhead due to the computation of the distance-based component. However, both types of queries require scanning about 10% of the lists due to the anticorrelated nature of the dataset (which negatively affects TA's performance). In particular, DIR queries with a TIGHT bounding strategy require between 200ms and 262ms when k varies between 1 and 100.

5.3 Other competitors

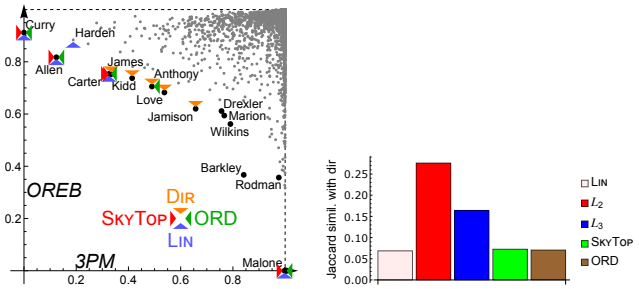
Here we compare DIR queries with other methods besides LIN queries.

Figure 16a shows how the top-5 tuples from Table 4 obtained from the LIN and the DIR query with $\beta = 2/3$ are spatially distributed. Also shown are the top-5 players according to two other methods, namely (i) SKYTOP, that first computes the skyline and then selects from it the top-5 tuples according to a linear function, and (ii) ORD [25]. ORD starts with a *seed* weight vector and then considers the set V of all weight vectors v whose distance from w is at most ρ. The value of ρ is implicitly determined by the number of results, m, one wants to obtain. Then, a tuple s is *ρ-dominated* if there exists a tuple t whose score is better than or equal to that of s when considering all weight vectors in V. ORD also requires the specification of another parameter, k, that guarantees that all the m results are ρ-dominated by less than k tuples. In Figure 16a we set $m = 5$ and $k = 1$, thus ORD returns 5 ρ-undominated tuples (i.e., a subset of the skyline). Besides confirming that DIR queries provide more balanced results, even with respect to SKYTOP and ORD, it is apparent that such results are quite dissimilar from those of all the other methods. In particular, 3 out of 5 players obtained with $\beta = 2/3$ are not present in any of the other top-5 result sets.

For a more complete analysis, we measured the Jaccard similarity of top-10 results over a sample of 100 random queries on $\mathsf{ANT}_{3;1M}$. Figure 16b confirms the peculiarity of DIR queries, with all competing methods sharing no more than 27% of the results with DIR queries. Note that the highest similarities are obtained when using non-linear functions in $\mathcal{L}_2$ and $\mathcal{L}_3$, whereas the Jaccard similarity wrt to all the methods shown in Figure 16a is only at most 7%.

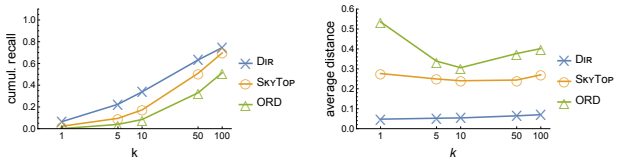
Figure 17 shows that, even with respect to ORD and SKYTOP, DIR queries are consistently better both in terms of cumulative recall and average distance from the preference line. The same holds for non-linear queries using $\mathcal{L}_p$ functions (see Figure 18), for which increasing the non-linearity (i.e., the value of p) leads to worsening both $\widehat{\text{REC}}$ and average DIST. This confirms what was observed at

[7] This follows from the observation that yet-to-be-seen tuples lie in a (hyper-)rectangular region with opposite vertices $\langle l_1, \ldots, l_d \rangle$ and $\langle 1, \ldots, 1 \rangle$, thus arguments used for R-trees still apply here.



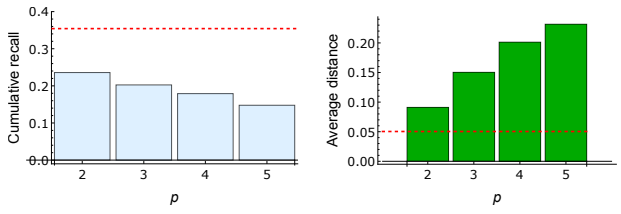
(a) Top-5 queries on NBA with $w = \langle 0.5, 0.5 \rangle$ **(b) Jaccard similarity wrt DIR queries on $\mathtt{ANT}_{3;1M}$**

Figure 16: Comparing results of different kinds of queries.



(a) Cumulative recall **(b) Average distance**

Figure 17: Cumulative recall and average distance from PL(w) for SKYTOP and ORD on $\mathtt{ANT}_{3;1M}$.



(a) Cumulative recall **(b) Average distance**

Figure 18: Cumulative recall and average distance from PL(w) for non-linear $\mathcal{L}_p$ functions on $\mathtt{ANT}_{3;1M}$. The red dotted line refers to directional queries.

the beginning of Section 4, in that the higher p is, the less changing weights can alter the result of a query.

6 RELATED WORK

There are several lines of research that are somehow related to our work. Some of these consider the notions of skyline queries and top-k queries together, trying to get the best of both worlds. In particular, given a value for k and a family of scoring functions $\mathcal{F}$, k-regret queries [8, 28] aim to return a set S of k tuples such that, for each function $f \in \mathcal{F}$, the ratio between the best score of a tuple in S and the best score in the dataset is maximized. When $\mathcal{F} = \mathcal{L}_1$, as is usually the case, the optimal S is a subset of the convex tuples in the skyline, thus not including any concave skyline tuple, which is a major objective of our approach. Non-linear k-regret queries are considered in [14], in particular extending results of [28] to classes of functions that include $\mathcal{L}_p$ norms. We observe, however, that, as clearly argued in [25], approaches based on regret are unable to accommodate user preferences, which are one of the main ingredients of directional queries. Notice that adapting a query to user preferences is a requirement that remains valid even in the latest approaches to the problem of combining top-k and skyline queries. With the common idea of extending the notion of dominance, both the works in [9–11, 26] and the one in [25] consider regions in the space of weights (namely, a convex polytope or a hyper-sphere) instead of a single weight vector. The former group of papers deals with arbitrary families of $\mathcal{L}_p$ functions; however, they are unable to explicitly control the cardinality of the result. On the contrary, [25] offers cardinality control (like top-k queries do), yet it is limited to linear scoring functions, thereby incurring all the limitations described in this paper.

We have already mentioned in Section 3 some of the works that try to overcome the lack of cardinality control of skylines (a prominent study on the cardinality of skylines is available in [16]). This gave rise to many skyline variants, based on point-wise ranking [29, 36, 37], subspace reference [5, 38], set-wide properties [22, 31], and more [24, 34, 35]; some of these variants are discussed in a survey [23]. Part of these ideas inspired our work on the indicators for robustness (evol and gres). We also mentioned the maximum rank [27], which we extended into our brank indicator. Our cdeg indicator builds on the analysis of the $\mathcal{L}_p$ families of scoring functions in [11]. The notions of stability and scale invariance were introduced in [28] as a requirement for skyline-based operators. We observe that all our indicators are scale-invariant, and all but brank are stable.

Preliminary observations on precision and recall of top-k queries with respect to the skyline are contained in [9, 11]. However, no work that we are aware of has ever tried to assess the general ability of (linear or non-linear) top-k queries to retrieve skyline tuples. As discussed in the introduction, all the mentioned approaches have limitations in retrieving the tuples of interest.

7 CONCLUSIONS

Based on the observation that linear top-k queries can fail to obtain several relevant results, regardless of user preferences, and that they also have difficulties in returning balanced results respecting such preferences, in this paper we have first quantified this problem through the introduction of four novel indicators. These indicators apply to skyline tuples and measure both their robustness and the difficulty in retrieving them. The presence of skyline tuples that are hard to retrieve but robust is observed to occur very frequently in all the datasets we considered.

Then, to obviate the above-mentioned limitations, we have introduced a novel type of scoring functions, leading to what we call directional queries. They complement the mean-based component typical of linear queries with a term that accounts for the balance of a tuple with respect to a so-called preference line, thus adhering more faithfully to the user preferences.

Experimental results obtained on both real and synthetic datasets indeed show the effectiveness of directional queries, with nearly no computational overhead with respect to the classical linear queries.

As future work, we observe that balance is only seemingly opposed to the requirement of diversity of the tuples in the result set [12, 15]. Indeed, the very notion of diversity is independent of

the family of scoring functions in use, and could thus be combined with directional queries into an integrated framework.

REFERENCES

[1] Y. Barkan. 2009. Rtree. https://github.com/nushoin/RTree.

[2] Stefan Berchtold, Christian Böhm, Daniel A. Keim, and Hans-Peter Kriegel. 1997. A Cost Model For Nearest Neighbor Search in High-Dimensional Data Space. In *Proceedings of the Sixteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, May 12-14, 1997, Tucson, Arizona, USA*, Alberto O. Mendelzon and Z. Meral Özsoyoglu (Eds.). ACM Press, 78–86. https://doi.org/10.1145/263661.263671

[3] Stephan Börzsönyi, Donald Kossmann, and Konrad Stocker. 2001. The Skyline Operator. In *Proceedings of the 17th International Conference on Data Engineering, April 2-6, 2001, Heidelberg, Germany*. 421–430. https://doi.org/10.1109/ICDE.2001.914855

[4] Karl Bringmann and Tobias Friedrich. 2012. Approximating the least hypervolume contributor: NP-hard in general, but fast in practice. *Theor. Comput. Sci.* 425 (2012), 104–116. https://doi.org/10.1016/J.TCS.2010.09.026

[5] Chee Yong Chan, H. V. Jagadish, Kian-Lee Tan, Anthony K. H. Tung, and Zhenjie Zhang. 2006. On High Dimensional Skylines. In *Advances in Database Technology - EDBT 2006, 10th International Conference on Extending Database Technology, Munich, Germany, March 26-31, 2006, Proceedings*. 478–495. https://doi.org/10.1007/11687238_30

[6] Yuan-Chi Chang, Lawrence D. Bergman, Vittorio Castelli, Chung-Sheng Li, Ming-Ling Lo, and John R. Smith. 2000. The Onion Technique: Indexing for Linear Optimization Queries. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, May 16-18, 2000, Dallas, Texas, USA*. 391–402. https://doi.org/10.1145/342009.335433

[7] Bernard Chazelle. 1991. An Optimal Convex Hull Algorithm and New Results on Cuttings (Extended Abstract). In *32nd Annual Symposium on Foundations of Computer Science, San Juan, Puerto Rico, 1-4 October 1991*. IEEE Computer Society, 29–38. https://doi.org/10.1109/SFCS.1991.185345

[8] Sean Chester, Alex Thomo, S. Venkatesh, and Sue Whitesides. 2014. Computing k-Regret Minimizing Sets. *PVLDB* 7, 5 (2014), 389–400. https://doi.org/10.14778/2732269.2732275

[9] Paolo Ciaccia and Davide Martinenghi. 2017. Reconciling Skyline and Ranking Queries. *PVLDB* 10, 11 (2017), 1454–1465. https://doi.org/10.14778/3137628.3137653

[10] Paolo Ciaccia and Davide Martinenghi. 2018. FA + TA <FSA: Flexible Score Aggregation. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management, CIKM 2018, Torino, Italy, October 22-26, 2018*. 57–66. https://doi.org/10.1145/3269206.3271753

[11] Paolo Ciaccia and Davide Martinenghi. 2020. Flexible Skylines: Dominance for Arbitrary Sets of Monotone Functions. *ACM Trans. Database Syst.* 45, 4 (2020), 18:1–18:45. https://doi.org/10.1145/3406113

[12] Marina Drosou and Evaggelia Pitoura. 2010. Search result diversification. *SIGMOD Record* 39, 1 (2010), 41–47.

[13] Ronald Fagin, Amnon Lotem, and Moni Naor. 2001. Optimal Aggregation Algorithms for Middleware. In *PODS*. https://doi.org/10.1145/375551.375567

[14] Taylor Kessler Faulkner, Will Brackenbury, and Ashwin Lall. 2015. k-Regret Queries with Nonlinear Utilities. *Proc. VLDB Endow.* 8, 13 (2015), 2098–2109. https://doi.org/10.14778/2831360.2831364

[15] Piero Fraternali, Davide Martinenghi, and Marco Tagliasacchi. 2012. Top-k bounded diversification. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2012, Scottsdale, AZ, USA, May 20-24, 2012*, K. Selçuk Candan, Yi Chen, Richard T. Snodgrass, Luis Gravano, and Ariel Fuxman (Eds.). ACM, 421–432. https://doi.org/10.1145/2213836.2213884

[16] Parke Godfrey. 2004. Skyline Cardinality for Relational Processing. In *Foundations of Information and Knowledge Systems, Third International Symposium, FoIKS 2004, Wilhelminenberg Castle, Austria, February 17-20, 2004, Proceedings*. 78–97. https://doi.org/10.1007/978-3-540-24627-5_7

[17] Andreia P. Guerreiro, Carlos M. Fonseca, and Luís Paquete. 2022. The Hypervolume Indicator: Computational Problems and Algorithms. *ACM Comput. Surv.* 54, 6 (2022), 119:1–119:42. https://doi.org/10.1145/3453474

[18] Antonin Guttman. 1984. R-Trees: A Dynamic Index Structure for Spatial Searching. In *SIGMOD'84, Proceedings of Annual Meeting, Boston, Massachusetts, USA, June 18-21, 1984*, Beatrice Yormark (Ed.). ACM Press, 47–57. https://doi.org/10.1145/602259.602266

[19] Georges Hebrail and Alice Berard. 2012. Individual household electric power consumption. https://archive.ics.uci.edu/dataset/235/individual+household+electric+power+consumption. Last accessed March 4, 2024.

[20] Ihab F. Ilyas, George Beskales, and Mohamed A. Soliman. 2008. A survey of top-k query processing techniques in relational database systems. *ACM Comput. Surv.* 40, 4 (2008). https://doi.org/10.1145/1391729.1391730

[21] Steven G. Johnson. 2007. The NLopt nonlinear-optimization package. https://github.com/stevengj/nlopt.

[22] Xuemin Lin, Yidong Yuan, Qing Zhang, and Ying Zhang. 2007. Selecting Stars: The k Most Representative Skyline Operator. In *Proceedings of the 23rd International Conference on Data Engineering, ICDE 2007, The Marmara Hotel, Istanbul, Turkey, April 15-20, 2007*. 86–95. https://doi.org/10.1109/ICDE.2007.367854

[23] Christoph Lofi and Wolf-Tilo Balke. 2013. On Skyline Queries and How to Choose from Pareto Sets. In *Advanced Query Processing, Volume 1: Issues and Trends*. 15–36. https://doi.org/10.1007/978-3-642-28323-9_2

[24] Hua Lu, Christian S. Jensen, and Zhenjie Zhang. 2011. Flexible and Efficient Resolution of Skyline Query Size Constraints. *IEEE Trans. Knowl. Data Eng.* 23, 7 (2011), 991–1005. https://doi.org/10.1109/TKDE.2010.47

[25] Kyriakos Mouratidis, Keming Li, and Bo Tang. 2021. Marrying Top-k with Skyline Queries: Relaxing the Preference Input while Producing Output of Controllable Size. In *SIGMOD '21: International Conference on Management of Data, Virtual Event, China, June 20-25, 2021*, Guoliang Li, Zhanhuai Li, Stratos Idreos, and Divesh Srivastava (Eds.). ACM, 1317–1330. https://doi.org/10.1145/3448016.3457299

[26] Kyriakos Mouratidis and Bo Tang. 2018. Exact Processing of Uncertain Top-k Queries in Multi-criteria Settings. *PVLDB* 11, 8 (2018), 866–879. https://doi.org/10.14778/3204028.3204031

[27] Kyriakos Mouratidis, Jilian Zhang, and HweeHwa Pang. 2015. Maximum Rank Query. *PVLDB* 8, 12 (2015), 1554–1565. http://www.vldb.org/pvldb/vol8/p1554-Mouratidis.pdf

[28] Danupon Nanongkai, Atish Das Sarma, Ashwin Lall, Richard J. Lipton, and Jun (Jim) Xu. 2010. Regret-Minimizing Representative Databases. *PVLDB* 3, 1 (2010), 1114–1124. https://doi.org/10.14778/1920841.1920980

[29] Dimitris Papadias, Yufei Tao, Greg Fu, and Bernhard Seeger. 2003. An Optimal and Progressive Algorithm for Skyline Queries. In *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data, San Diego, California, USA, June 9-12, 2003*, Alon Y. Halevy, Zachary G. Ives, and AnHai Doan (Eds.). ACM, 467–478. https://doi.org/10.1145/872757.872814

[30] San Francisco Open Data. 2016. Employee Compensation in SF. https://data.world/data-society/employee-compensation-in-sf. Last accessed November 23, 2023.

[31] Yufei Tao, Ling Ding, Xuemin Lin, and Jian Pei. 2009. Distance-Based Representative Skyline. In *Proceedings of the 25th International Conference on Data Engineering, ICDE 2009, March 29 2009 - April 2 2009, Shanghai, China*. 892–903. https://doi.org/10.1109/ICDE.2009.84

[32] Yufei Tao, Vagelis Hristidis, Dimitris Papadias, and Yannis Papakonstantinou. 2007. Branch-and-bound processing of ranked queries. *Inf. Syst.* 32, 3 (2007), 424–445. https://doi.org/10.1016/J.IS.2005.12.001

[33] Panayiotis Tsaparas, Themistoklis Palpanas, Yannis Kotidis, Nick Koudas, and Divesh Srivastava. 2003. Ranked Join Indices. In *Proceedings of the 19th International Conference on Data Engineering, March 5-8, 2003, Bangalore, India*, Umeshwar Dayal, Krithi Ramamritham, and T. M. Vijayaraman (Eds.). IEEE Computer Society, 277–288. https://doi.org/10.1109/ICDE.2003.1260799

[34] Akrivi Vlachou, Christos Doulkeridis, Kjetil Nørvrag, and Michalis Vazirgiannis. 2008. Skyline-based Peer-to-Peer Top-k Query Processing. In *Proceedings of the 24th International Conference on Data Engineering, ICDE 2008, April 7-12, 2008, Cancún, Mexico*. 1421–1423. https://doi.org/10.1109/ICDE.2008.4497576

[35] Akrivi Vlachou and Michalis Vazirgiannis. 2010. Ranking the sky: Discovering the importance of skyline points through subspace dominance relationships. *Data Knowl. Eng.* 69, 9 (2010), 943–964. https://doi.org/10.1016/j.datak.2010.03.008

[36] Man Lung Yiu and Nikos Mamoulis. 2007. Efficient Processing of Top-k Dominating Queries on Multi-Dimensional Data. In *Proceedings of the 33rd International Conference on Very Large Data Bases, University of Vienna, Austria, September 23-27, 2007*. 483–494. http://www.vldb.org/conf/2007/papers/research/p483-yiu.pdf

[37] Man Lung Yiu and Nikos Mamoulis. 2009. Multi-dimensional top-k dominating queries. *VLDB J.* 18, 3 (2009), 695–718. https://doi.org/10.1007/s00778-008-0117-y

[38] Zhenjie Zhang, Xinyu Guo, Hua Lu, Anthony K. H. Tung, and Nan Wang. 2005. Discovering strong skyline points in high dimensional spaces. In *Proceedings of the 2005 ACM CIKM International Conference on Information and Knowledge Management, Bremen, Germany, October 31 - November 5, 2005*. 247–248. https://doi.org/10.1145/1099554.1099610