Reconciling Skyline and Ranking Queries

Paolo Ciaccia
Dipartimento di Informatica - Scienza e Ingegneria (DISI)
Università di Bologna, Italy
paolo.ciaccia@unibo.it

Davide Martinenghi
Dipartimento di Elettronica, Informazione e Bioingegneria (DEIB)
Politecnico di Milano, Italy
davide.martinenghi@polimi.it

ABSTRACT

Traditionally, skyline and ranking queries have been treated separately as alternative ways of discovering interesting data in potentially large datasets. While ranking queries adopt a specific scoring function to rank tuples, skyline queries return the set of non-dominated tuples and are independent of attribute scales and scoring functions. Ranking queries are thus less general, but usually cheaper to compute and widely used in data management systems.

We propose a framework to seamlessly integrate these two approaches by introducing the notion of restricted skyline queries (R-skylines). We propose R-skyline operators that generalize both skyline and ranking queries by applying the notion of dominance to a set of scoring functions of interest. Such sets can be characterized, e.g., by imposing constraints on the function's parameters, such as the weights in a linear scoring function. We discuss the formal properties of these new operators, show how to implement them efficiently, and evaluate them on both synthetic and real datasets.

1. INTRODUCTION

When dealing with the problem of simultaneously optimizing different criteria (such as those represented by the different attributes of the objects in a dataset), a problem known as *multi-objective optimization*, three main approaches are commonly adopted [7] in all data-intensive contexts, including data mining and database systems. *(1)* The *ranking queries* (or top-k) approach: the original multi-objective problem is reduced to a single-objective problem by using a so-called scoring function, in which parameters such as weights are used to express the relative importance of the different attributes and to adjust scales. *(2)* The *lexicographical* approach: a strict priority among the attributes is established. *(3)* The *skyline* approach: all the non-dominated objects are returned to the user (object t dominates object s iff t is no worse than s on all the attributes, and strictly better on at least one). As argued in [7] and virtually in all papers focusing on a specific approach, each of these methods has pros and cons (also refer to Table 1). Ranking queries heavily depend on the particular choice of weights in the scoring function, and thus fail to offer an overall view of the dataset. This may require to repeat the query several times with different choices of weights, or to include in the scoring function other notions such as the diversity of the result set, which have problems of their own. The point of view of lexicographic queries is too narrow, in that they enforce a linear priority between attributes, and even the smallest difference in the most important attribute can never be compensated by the other attributes. Skyline queries provide a good overview of potentially interesting tuples, but may contain too many objects.

Table 1: Pros and cons of multi-objective optimization approaches.

Evaluation criteria ↓ Queries →	Ranking	Lexicographic	Skyline
Simplicity of formulation	No	Yes	Yes
Overall view of interesting results	No	No	Yes
Control of result cardinality	Yes	Yes	No
Trade-off among attributes	Yes	No	No
Relative importance of attributes	Yes	Yes	No

With the aim of reducing the size of the result of a skyline query, [15] has introduced *prioritized skyline* (p-skyline) queries, in which modalities 2 and 3 are combined together. However this "mixed" approach inherits the basic problem of lexicographic queries, in that it allows no trade-off between attributes. Furthermore, still no explicit control on the result cardinality is possible.

In this paper we introduce an original framework that combines the approaches 1 and 3 described above. In particular, based on the concepts of scoring functions and skyline, we introduce the notion of *restricted skyline* (R-skyline) queries. Similarly to p-skylines, R-skylines can take into account the different importance that different attributes might have. However, unlike p-skylines, in which a strict priority between attributes is assumed, R-skylines may be used to model priority by means of arbitrary constraints on the space of the weights, thus allowing for more flexibility. Furthermore, R-skylines may consider arbitrary families of scoring functions. Overall, this leads to the novel concept of $\mathcal{F}$-*dominance*, in which a tuple t $\mathcal{F}$-dominates tuple s when t is always better than or equal to s according to *all* the scoring functions in $\mathcal{F}$ (and strictly better for at least one scoring function in $\mathcal{F}$).

We present two R-skyline operators: ND, characterizing the set of non-$\mathcal{F}$-dominated tuples; PO, referring to the tuples that are potentially optimal, i.e., best according to some function in $\mathcal{F}$. While ND and PO coincide and capture the

Proceedings of the VLDB Endowment, Vol. 10, No. 11

Table 2: The UsedCars relation (shown column-wise).

CarID	C1	C2	C3	C4	C5	C6	C7
Price ($\times 10^3$)	10	18	20	20	25	35	40
Mileage ($\times 10^3$)	35	25	30	15	20	10	5

traditional skyline when $\mathcal{F}$ is the family of all monotone scoring functions, their behaviors differ when subsets thereof are considered. R-skylines indeed capture in a single framework all the practically relevant approaches to multi-objective optimization, traditionally dealt with separately, and enable the study of other scenarios of practical interest. For example, in multicriteria analysis, decision makers may encounter objectives in which the model parameters lack completeness or confidence, and are characterized by complex preferences between, e.g., attribute weights, such as "attribute C is more important than attribute A, but no more than twice as important" [18]. Other complex constraints characterizing the objective might come from preference elicitation from a crowd (see, e.g., [5] and references therein for strategies for collecting preferences between tuples).

EXAMPLE 1. *For the relation* UsedCars(ID,Price,Mileage) *in Table 2, a skyline query over the attributes Price and Mileage (both to be minimized) will return cars C1, C2, C4, C6, and C7. Now assume that your preferences consider Price more important than Mileage (in which case a p-skyline query would just return car C1, since it has the minimum price). By considering the family of scoring functions* $\mathcal{F} = \{w_P\texttt{Price} + w_M\texttt{Mileage} \mid w_P \geq w_M\}$, ND, *i.e., the set of non-*$\mathcal{F}$*-dominated cars, includes C1, C2, and C4, with only C1 and C4 being also part of* PO. *Although in the skyline, both C6 and C7 are* $\mathcal{F}$*-dominated by C4, which is reasonable since they both have a relatively high price. However, car C2 is non-*$\mathcal{F}$*-dominated, yet there is no combination of weights values making it a top-1 result.*

The main contributions of this paper are as follows.

1. We introduce two operators, called R-skylines, generalizing both skyline and ranking queries.
2. We study the properties of R-skylines and in particular their relationship with skyline and top-1 queries, as well as their behavior as the set $\mathcal{F}$ of scoring functions under consideration varies.
3. We study the application of R-skylines when the scoring functions in $\mathcal{F}$ are L_p norms or, generally, functions that are linear in the weights (or monotonic transforms thereof).
4. We discuss two alternative approaches to computing R-skylines based on Linear Programming, one addressing a direct $\mathcal{F}$-dominance test between tuples, the other characterizing the "dominance region" wrt. $\mathcal{F}$ of a tuple.
5. We evaluate the effectiveness of R-skylines (i.e., their ability to restrict the set of tuples of interest) in a number of different experimental settings including synthetic as well as real datasets; we also discuss different implementations of the operators and test their efficiency.

Related work. Due to the limits that each of the basic methods for multi-objective optimization exhibits, several approaches have been attempted to help in more easily finding interesting results in large datasets.

Several techniques have been proposed for reducing the skyline size, a recent survey of which can be found in [12]. Among them, distance-based representative skylines [21] aim to determine the k tuples in the skyline for which the maximum distance to the excluded skyline points is minimized. Since this problem is NP-hard, only approximate solutions can be provided. Furthermore, the method is also sensitive to the specific metric used to measure distance between tuples. Another approach to select a limited subset of skyline tuples is to assign to each of them a measure of interestingness based on some specific properties. Top-k Representative Skyline Points (RSP) [11] are the k skyline points that together dominate the maximum number of (non-skyline) points. Computing top-k RSP is NP-hard for three or more dimensions, thus approximate solutions are adopted in practice. Top-k dominating queries [22] return the k tuples that dominate the highest number of tuples in the dataset, i.e., they rank tuples according to the number of other tuples they dominate. Besides the high computational cost incurred by this approach if the input dataset is not indexed, a major drawback is that the score of a tuple depends on how worse tuples are distributed, a problem that this method shares with top-k RSP.

Among the methods that only rely on the order properties of skylines, i.e., without any reference to the actual underlying attribute domains (which can consequently also be categorical), we mention p-skylines and trade-off skylines. P-skyline (or Prioritized skyline) queries [15] are a generalization of skyline queries in which the user can specify that some attributes are more important than others, by respecting the syntax of so-called *p-expressions*. In practice, a p-expression over d attributes will have fewer than d "most important" attributes. Since these ultimately determine the size of the result, p-skylines usually contain many fewer tuples than skylines. P-skylines can be efficiently computed by taking advantage of the reduced cardinality of the result, i.e., with an output-sensitive algorithm [14]. The idea of trade-off skylines [13] is similar to the one we adopt in this paper. However, while we consider numerical domains and consequently *numerical* trade-offs, [13] adopts the view of *qualitative* trade-offs. Although the latter has the advantage of being also applicable to categorical attributes, the price to be paid is increased computational complexity.

Somehow related to what we study in this paper are those works on top-k queries in which the scoring function is not univocally defined, e.g., [24, 16]. Along these lines, [20] studies representative orderings (such as the most probable ordering) and their stability wrt. a change of parameter values, by assuming that the set of parameters (weights) is a random variable with a uniform distribution.

2. PRELIMINARIES

Consider a relational schema $R(A_1, \ldots, A_d)$, with $d \geq 1$. Without loss of generality, we assume that the domain of each attribute A_i is $[0, 1]$, since each numeric domain could be normalized in this interval. In this paper, we consider lower values to be better than higher ones, but the opposite convention would of course also be possible. A *tuple* t over R is a function that associates a value v_i in $[0, 1]$ with each attribute A_i; t is also written as $\langle v_1, \ldots, v_d \rangle$, and each v_i may be denoted by $t[A_i]$. Given the geometric interpretation of a tuple in this context, in the following we sometimes also call it a *point*. An *instance* over R is a set of tuples over R. In the following, we refer to an instance r over R.

DEFINITION 1 (DOMINANCE AND SKYLINE). *Let* s, t *be tuples over* R. *Then,* t dominates s, *written* $t \prec s$, *if*

(i) $\forall i.\ 1 \leq i \leq d \rightarrow t[A_i] \leq s[A_i]$*, and (ii)* $\exists j.\ 1 \leq j \leq d \wedge t[A_j] < s[A_j]$*. The* skyline *of* r *(*$\textsc{Sky}(r)$*) is defined as:*

$$\textsc{Sky}(r) = \{t \in r \mid \nexists s \in r.\ s \prec t\}. \quad (1)$$

Equivalent definitions of skyline may be derived by resorting to the notion of monotone scoring functions, i.e., those monotone functions that can be applied to tuples over R to obtain a non-negative value representing a score.

Definition 2 (Monotone scoring function). *A* scoring function f *is a function* $f : [0,1]^d \rightarrow \mathbb{R}^+$*. 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* $f(t)$*. Function* f *is* monotone *if, for any tuples* t, s *over* R*, the following holds:*

$$(\forall i \in \{1, \ldots, d\}.\ t[A_i] \leq s[A_i]) \rightarrow f(t) \leq f(s). \quad (2)$$

The (infinite) set of all monotone scoring functions is denoted by $\mathcal{M}$*.*

Note that, as a consequence of our preference for lower attribute values, lower score values are also preferred over higher ones. Intuitively, scoring functions could be thought of as measuring a sort of distance from the "origin" tuple $\langle 0, \ldots, 0 \rangle$, and we prefer tuples closer to the origin.

It is well known [3] that, for every tuple t in the skyline, there exists a monotone scoring function such that t minimizes that scoring function. Therefore, the skyline of r can be equivalently specified as:

$$\textsc{Sky}(r) = \{t \in r \mid \exists f \in \mathcal{M}.\ \forall s \in r.\ s \neq t \rightarrow f(t) < f(s)\}. \quad (3)$$

The previous expressions emphasize two possible ways to regard a skyline: *(i)* as the set of all *non-dominated* tuples (Equation (1)), or *(ii)* as the set of *potentially optimal* tuples, i.e., those that are better than all the others according to at least one monotone scoring function (Equation (3)). While the former view is typically adopted for skylines, the latter is commonly applied to "top-k" queries (here with $k = 1$), i.e., those queries whose goal is to return the k best tuples according to a given scoring function. As we shall see in Section 3, although these two views coincide here, their underlying concepts are different.

3. RESTRICTED SKYLINES

We now adopt the two different views of skylines to introduce two corresponding operators, called *restricted skyline* operators (R-skylines), whose behavior is the same as Sky, but applied to a limited set of monotone scoring functions $\mathcal{F} \subseteq \mathcal{M}$. In the following, we always assume $\mathcal{F}$ to be non-empty. In order to precisely characterize the notions to be presented in this paper, we introduce the following property regarding sets of scoring functions.

Definition 3. *A set* $\mathcal{F}$ *of scoring functions is said to be* tuple-distinguishing *if the following holds:*

$$\forall t, s \in [0,1]^d.\ t \neq s \rightarrow (\exists f \in \mathcal{F}.\ f(t) \neq f(s)). \quad (4)$$

Intuitively, $\mathcal{F}$ satisfies Equation (4) if $\mathcal{F}$ is "rich enough" to distinguish between any two different tuples, i.e., if there is at least a function in $\mathcal{F}$ associating two different scores to two different tuples. Most interesting cases of sets of monotone scoring functions are tuple-distinguishing. However, there are a few notable exceptions, among which the case of sets of one single function, or the case of sets of functions independent of an attribute. All of these cases are also captured by our framework. However, in order to simplify the presentation, we shall henceforth only consider tuple-distinguishing sets of functions, and implicitly assume this property in the rest of the paper.

We now extend the notion of dominance introduced in Definition 1 so as to take into account the set of scoring functions under consideration.

Definition 4 ($\mathcal{F}$-Dominance). *Let* $\mathcal{F}$ *be a set of monotone scoring functions. A tuple* t $\mathcal{F}$-dominates *another tuple* $s \neq t$*, denoted by* $t \prec_{\mathcal{F}} s$*, iff* $\forall f \in \mathcal{F}.\ f(t) \leq f(s)$*.*

Example 2. *Consider the tuples* $t = \langle 0.5, 0.5 \rangle$*,* $s = \langle 0, 1 \rangle$*, the monotone scoring functions* $f_1(x, y) = x + y$ *and* $f_2(x, y) = x + 2y$*, and the set* $\mathcal{F} = \{f_1, f_2\}$*. We have* $t \prec_{\mathcal{F}} s$*, since* $f_1(t) = f_1(s) = 1$ *and* $f_2(t) = 1.5 < f_2(s) = 2$*, and therefore the condition of Definition 4 holds.*

However, $t \not\prec_{\mathcal{M}} s$*, since* $\mathcal{M}$ *includes, among others,* $f_3(x, y) = 2x + y$*, for which* $f_3(t) = 1.5 > f_3(s) = 1$*, thereby violating the condition of Definition 4.*

With Definition 4 at hand, we can now introduce the first R-skyline operator, called *non-dominated restricted skyline*, which consists of the set of non-$\mathcal{F}$-dominated tuples in r.

Definition 5 (nd). *Let* $\mathcal{F} \subseteq \mathcal{M}$ *be a set of monotone scoring functions. The* non-dominated restricted skyline *of* r *with respect to* $\mathcal{F}$*, denoted by* $\textsc{nd}(r; \mathcal{F})$*, is defined as the following set of tuples:*

$$\textsc{nd}(r; \mathcal{F}) = \{t \in r \mid \nexists s \in r.\ s \prec_{\mathcal{F}} t\}. \quad (5)$$

Note that the right-hand side of Equation (5) is similar to that of Equation (1), where $\prec$ has been replaced by $\prec_{\mathcal{F}}$. Observe that, clearly, $\prec_{\mathcal{M}}$ coincides with $\prec$.

The second R-skyline operator, called *potentially optimal restricted skyline*, returns the tuples that are best (i.e., top 1) according to some scoring function in $\mathcal{F}$.

Definition 6 (po). *Let* $\mathcal{F} \subseteq \mathcal{M}$ *be a set of monotone scoring functions. The* potentially optimal restricted skyline *of* r *with respect to* $\mathcal{F}$*, denoted by* $\textsc{po}(r; \mathcal{F})$*, is defined as:*

$$\textsc{po}(r; \mathcal{F}) = \{t \in r \mid \exists f \in \mathcal{F}.\ \forall s \in r.\ s \neq t \rightarrow f(t) < f(s)\}. \quad (6)$$

Note that the right-hand side of Equation (6) is similar to that of Equation (3), where $\mathcal{M}$ has been replaced by $\mathcal{F}$.

In the remainder of the paper we discuss the main properties of these operators and study how to compute them efficiently, thus addressing Problem 1 below.

Problem 1. *To efficiently compute* $\textsc{nd}(r; \mathcal{F})$ *and* $\textsc{po}(r; \mathcal{F})$ *for any given instance* r *and set of monotone scoring functions* $\mathcal{F}$*.*

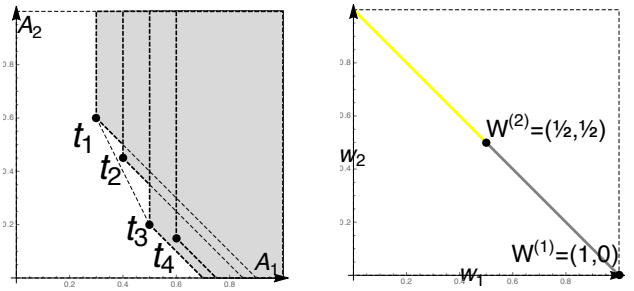
3.1 Basic Properties

In the following we present basic facts about nd and po, and further investigate their relationship with Sky.

As a direct consequence of the definitions, we observe that, when the set $\mathcal{F}$ of scoring functions under consideration coincides with $\mathcal{M}$, it is $\textsc{po}(r; \mathcal{M}) = \textsc{nd}(r; \mathcal{M}) = \textsc{Sky}(r)$. In general, though, there is a containment relationship, as indicated in Proposition 1 below.[1]

Proposition 1. *For any set* $\mathcal{F}$ *of monotone scoring functions, it is* $\textsc{po}(r; \mathcal{F}) \subseteq \textsc{nd}(r; \mathcal{F}) \subseteq \textsc{Sky}(r)$*.*

[1]All proofs are omitted in the interest of space.



(a) Tuples from Example 3 in $[0,1]^d$, $d=2$. $\mathcal{F}$-dominance regions in gray.

(b) Region of normalized weights such that $w_1 \geq w_2$ (in gray).

Figure 1: Example 3 – tuples and weights in $[0,1]^d$, $d=2$, $\mathcal{C}=\{w_1 \geq w_2\}$, $\mathcal{F}=\mathcal{L}_1^{\mathcal{C}}$, where $\mathcal{L}_1$ is the set of monotone scoring functions that are weighted sums of attribute values.

We also observe that ND and PO are monotone operators with respect to the set of scoring functions, as specified in Proposition 2 below.

PROPOSITION 2. *For any two sets $\mathcal{F}_1$ and $\mathcal{F}_2$ of monotone scoring functions such that $\mathcal{F}_1 \subseteq \mathcal{F}_2$, it is:*

$$\text{ND}(r;\mathcal{F}_1) \subseteq \text{ND}(r;\mathcal{F}_2), \qquad \text{PO}(r;\mathcal{F}_1) \subseteq \text{PO}(r;\mathcal{F}_2). \quad (7)$$

A case of practical relevance is when one starts with a set $\mathcal{F}$ of scoring functions and adds some constraints on the way they are defined. A notable example is that of functions characterized by parameters such as weights. To this end, let $\mathcal{W}$ be the set of all normalized weight vectors, i.e., $\mathcal{W} \subseteq [0,1]^d$ and, for each $W=(w_1,\ldots,w_d) \in \mathcal{W}$, we have $\sum_{i=1}^d w_i = 1$. Let $\mathcal{C}$ be a, possibly empty, set of (linear) constraints on weights, and denote with $\mathcal{W}(\mathcal{C})$ the subset of $\mathcal{W}$ that satisfies $\mathcal{C}$, i.e.,: $\mathcal{W}(\mathcal{C})=\{W \in \mathcal{W} \mid \mathcal{C}(W)=\texttt{true}\}$. If $\mathcal{F}$ is a set of functions with parameters $w_1,\ldots,w_d$, we denote by $\mathcal{F}^{\mathcal{C}}$ the set of functions obtained from set $\mathcal{F}$ by imposing the set of constraints $\mathcal{C}$. Henceforth, we always assume that $\mathcal{C}$ is not contradictory, i.e., $\mathcal{W}(\mathcal{C}) \neq \emptyset$, and that the application of $\mathcal{C}$ leads to a non-empty set $\mathcal{F}^{\mathcal{C}} \neq \emptyset$.

COROLLARY 1. *For any set $\mathcal{F}$ of monotone functions and sets of constraints $\mathcal{C}_1$ and $\mathcal{C}_2$ such that $\mathcal{W}(\mathcal{C}_1) \subseteq \mathcal{W}(\mathcal{C}_2)$, it is:*

$$\text{ND}(r;\mathcal{F}^{\mathcal{C}_1}) \subseteq \text{ND}(r;\mathcal{F}^{\mathcal{C}_2}), \quad \text{PO}(r;\mathcal{F}^{\mathcal{C}_1}) \subseteq \text{PO}(r;\mathcal{F}^{\mathcal{C}_2}). \quad (8)$$

We now define the $\mathcal{F}$-dominance region of a tuple t.

DEFINITION 7. *The $\mathcal{F}$-*dominance region *$DR(t;\mathcal{F})$ of a tuple t under a set $\mathcal{F}$ of monotone scoring functions is the set of all points in $[0,1]^d$ that are $\mathcal{F}$-dominated by t:*

$$DR(t;\mathcal{F}) = \{s \in [0,1]^d \mid t \prec_{\mathcal{F}} s\}. \quad (9)$$

A consequence of Definition 7 is that the $\mathcal{F}$-dominance region grows larger for smaller sets of functions, as specified in Corollary 2 below.

COROLLARY 2. *For any tuple t over R and any two sets $\mathcal{F}_1$ and $\mathcal{F}_2$ of monotone scoring functions such that $\mathcal{F}_1 \subseteq \mathcal{F}_2$, it is $DR(t;\mathcal{F}_1) \supseteq DR(t;\mathcal{F}_2)$.*

We now illustrate Proposition 1 and Definition 7 with the following Example.

EXAMPLE 3. *Let $\mathcal{L}_1$ be the set of all the linear scoring functions of the form $f(x,y)=w_1x+w_2y$ and let $\mathcal{F}=\mathcal{L}_1^{\mathcal{C}}$, where $\mathcal{C}=\{w_1 \geq w_2\}$. Consider tuples $t_1=\langle 0.3,0.6\rangle$, $t_2=\langle 0.4,0.45\rangle$, $t_3=\langle 0.5,0.2\rangle$, $t_4=\langle 0.6,0.15\rangle$, and instance $r=\{t_1,t_2,t_3,t_4\}$, shown in Figure 1a. We have $\text{PO}(r;\mathcal{F})=\{t_1,t_3\} \subseteq \text{ND}(r;\mathcal{F})=\{t_1,t_2,t_3\} \subseteq \text{SKY}(r)=r$.*

To see this, first observe that no tuple in r dominates any other tuple in r, and therefore $\text{SKY}(r)=r$. However, we note that $t_3 \prec_{\mathcal{F}} t_4$: indeed, checking whether $f(t_3) \leq f(t_4)$ amounts to checking whether $w_1(0.5-0.6) \leq w_2(0.15-0.2)$, which is always true in $\mathcal{F}$, since $w_1 \geq w_2$. Therefore $t_4 \notin \text{ND}(r;\mathcal{F})$. To further emphasize this, Figure 1a shows in gray the region of $[0,1]^d$ whose points (including tuple t_4) are $\mathcal{F}$-dominated by some tuple in r, i.e., $\cup_{t\in r} DR(t;\mathcal{F})$, whereas Figure 1b shows in gray the region of normalized weights such that $w_1 \geq w_2$. The computation of such regions will be studied in depth in Section 4.1.

Finally, with linear scoring functions, as is well known [20], top-1 tuples can only lie in the boundary of the convex hull of the $\mathcal{F}$-dominated region, thus $t_2 \notin \text{PO}(r;\mathcal{F})$. Indeed, there is no function $f \in \mathcal{F}$ for which both $f(t_2) < f(t_1)$ and $f(t_2) < f(t_3)$, as there are no w_1,w_2 such that $w_1(0.4-0.3) < w_2(0.6-0.45)$, $w_1(0.4-0.5) < w_2(0.2-0.45)$, and $w_1 \geq w_2$ all hold.

4. R-SKYLINES AND L_P NORMS

A practically relevant case to consider is that of the *weighted* L_p norms, defined as follows, where $W=(w_1,\ldots,w_d) \in \mathcal{W}$ is a normalized weight vector:

$$L_p^W(t) = \left(\sum_{i=1}^d w_i t[A_i]^p\right)^{1/p}, \quad p \in \mathbb{N}. \quad (10)$$

We therefore turn our attention to the case in which the set of monotone scoring functions coincides with the family $\mathcal{L}_p$ of weighted L_p norms:

$$\mathcal{L}_p = \{L_p^W \mid W \in \mathcal{W}\}, \quad p \in \mathbb{N}. \quad (11)$$

The behaviors of ND and PO are very different under $\mathcal{L}_p$.

THEOREM 4. *For every value of p and every r, $\text{ND}(r;\mathcal{L}_p)=\text{SKY}(r)$.*

Thus, any $\mathcal{L}_p$ family is "powerful enough" to reveal all skyline points with ND. However, this does not hold for PO, as indicated in the following theorems.

THEOREM 5. *Let $p<p'$, with $p,p' \in \mathbb{N}$. Then, for every r, $\text{PO}(r;\mathcal{L}_p) \subseteq \text{PO}(r;\mathcal{L}_{p'})$.*

THEOREM 6. *For each $p \in \mathbb{N}$, there exists a relation r such that $\text{PO}(r;\mathcal{L}_p) \subset \text{SKY}(r)$.*

The results of Theorems 4, 5 and 6, together with the observation of Corollary 2, suggest that, by imposing some constraints $\mathcal{C}$ on the weights, one can use any $\mathcal{L}_p$ family to smoothly move from the full skyline (when $\mathcal{C}=\emptyset$) to top-1 queries (when $\mathcal{W}(\mathcal{C})$ amounts to a single weight vector).

4.1 Checking $\mathcal{F}$-Dominance for L_p norms

In order to better understand the notion of $\mathcal{F}$-dominance, we focus on the case $\mathcal{F}=\mathcal{L}_p^{\mathcal{C}}$, where $\mathcal{C}$ are linear constraints on the weights, and show that the problem is in PTIME and that its time complexity is independent of p.

THEOREM 7 ($\mathcal{F}$-DOMINANCE TEST). *Let p be a finite positive integer and $\mathcal{C} = \{C_1, \ldots, C_c\}$ a set of linear constraints on weights, where $C_j = \sum_{i=1}^{d} a_{ji} w_i \leq k_j$ (for $j \in \{1, \ldots, c\}$). Then, $t \prec_{\mathcal{L}_p^{\mathcal{C}}} s$ iff the following linear programming problem (LP) in the unknowns $W = (w_1, \ldots, w_d)$ has a non-negative solution:*

$$\begin{array}{lll} \text{minimize} & \sum_{i=1}^{d} w_i (s[A_i]^p - t[A_i]^p) & (12) \\ \text{subject to} & w_i \in [0,1] & i \in \{1, \ldots, d\} \\ & \sum_{i=1}^{d} w_i = 1 & \\ & \sum_{i=1}^{d} a_{ji} w_i \leq k_j & j \in \{1, \ldots, c\}. \end{array}$$

Note that Theorem 7 has a validity that goes far beyond $\mathcal{L}_p^{\mathcal{C}}$ families, since it applies to *any* set $\mathcal{F}$ whose functions are weighted sums of monotone functions of single attributes, e.g., $\sum_i w_i \log(1 + t[A_i])$, and even monotonic transforms of those, e.g., $\exp(\sum_i w_i t[A_i])$.

Computing $\text{ND}(r; \mathcal{F})$ using Theorem 7 is likely to be time-consuming, since a different LP problem needs to be solved for each $\mathcal{F}$-dominance test. An alternative approach is to explicitly compute the $\mathcal{F}$-dominance regions of tuples, and then discard those tuples that belong to at least one of such regions. The advantage of this approach is that the computation of the $\mathcal{F}$-dominance region of a tuple t can be performed just once, thus independently of how many $\mathcal{F}$-dominance tests involve t. Furthermore, as shown below, the cost of the most expensive component (i.e., vertex enumeration of a polytope) of the calculation of $\mathcal{F}$-dominance regions has to be paid just once for *all* tuples.

In order to compute $DR(t; \mathcal{L}_p^{\mathcal{C}})$, a fundamental observation is that, for any set $\mathcal{C}$ of linear constraints on weights, $\mathcal{W}(\mathcal{C})$ is a convex polytope contained in the standard (or unit) $(d-1)$-simplex.[2] We have the following major result.

THEOREM 8 ($\mathcal{F}$-DOMINANCE REGION). *Let $p \in \mathbb{N}$ and $\mathcal{C} = \{C_1, \ldots, C_c\}$ be a set of linear constraints on weights, where $C_j = \sum_{i=1}^{d} a_{ji} w_i \leq k_j$ ($j \in \{1, \ldots, c\}$). Let $W^{(1)}, \ldots, W^{(q)}$ be the vertices of $\mathcal{W}(\mathcal{C})$. The dominance region $DR(t; \mathcal{L}_p^{\mathcal{C}})$ of a tuple t under $\mathcal{L}_p^{\mathcal{C}}$ is the locus of points s defined by the q inequalities:*

$$\sum_{i=1}^{d} w_i^{(\ell)} s[A_i]^p \geq \sum_{i=1}^{d} w_i^{(\ell)} t[A_i]^p, \quad \ell \in \{1, \ldots, q\}. \qquad (13)$$

As a consequence of Definition 7, $DR(t; \mathcal{F})$ is a closed region.

EXAMPLE 9. *Let $d = 2$, $p = 1$, and consider tuples $t_1 = \langle 0.3, 0.6 \rangle$, $t_3 = \langle 0.5, 0.2 \rangle$, $t_4 = \langle 0.6, 0.15 \rangle$ from Example 3. For $\mathcal{C} = \{w_1 \geq w_2\}$ and considering that $w_1 + w_2 = 1$ and $0 \leq w_1, w_2 \leq 1$, the vertices of $\mathcal{W}(\mathcal{C})$ are $W^{(1)} = (1, 0)$ and $W^{(2)} = (0.5, 0.5)$. Figure 1a shows the tuples along with their $\mathcal{L}_1^{\mathcal{C}}$-dominance regions, while Figure 1b shows $\mathcal{W}(\mathcal{C})$. By Theorem 8, $DR(t_3; \mathcal{L}_1^{\mathcal{C}})$ is characterized by the system of inequalities:*

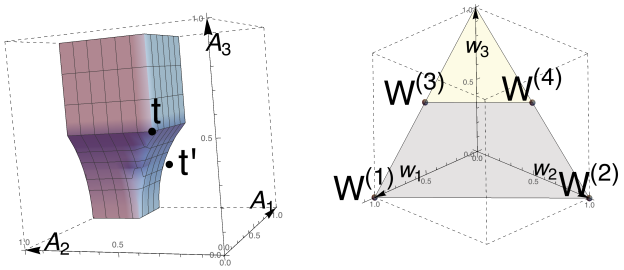
$$\{s[A_1] \geq 0.5, \quad s[A_1] + s[A_2] \geq 0.7\}. \qquad (14)$$

Tuple t_4 satisfies (14) and thus $t_3 \prec_{\mathcal{L}_1^{\mathcal{C}}} t_4$. For tuple t_1, the system becomes:

$$\{s[A_1] \geq 0.3, \quad s[A_1] + s[A_2] \geq 0.9\}. \qquad (15)$$

Here, t_4 does not satisfy (15) and therefore $t_1 \not\prec_{\mathcal{L}_1^{\mathcal{C}}} t_4$.

[2]Note that the standard $(d-1)$-simplex is a $(d-1)$-dimensional region in $\mathbb{R}^d$.



(a) $\mathcal{F}$-dominance region $DR(t, \mathcal{F})$.

(b) $\mathcal{W}(\mathcal{C})$ (in gray) on the 2-simplex.

Figure 2: Example 10 – tuples and weights in $[0,1]^d$, $d = 3$, $\mathcal{C} = \{w_1 + w_2 \geq w_3\}$, $\mathcal{F} = \mathcal{L}_2^{\mathcal{C}}$.

As Example 9, Figure 1a and Inequalities (13) suggest, the "shape" of $DR(t; \mathcal{L}_p^{\mathcal{C}})$ (modulo cropping in the $[0,1]^d$ hypercube) is independent of t, since the left-hand sides are the same and the right-hand sides are, for any given t, a constant.

EXAMPLE 10. *For a non-linear example, let $d = 3$, $p = 2$, and $\mathcal{C} = \{w_1 + w_2 \geq w_3\}$. The vertices of $\mathcal{W}(\mathcal{C})$ are: $W^{(1)} = \langle 1, 0, 0 \rangle$, $W^{(2)} = \langle 0, 1, 0 \rangle$, $W^{(3)} = \langle 0.5, 0, 0.5 \rangle$, and $W^{(4)} = \langle 0, 0.5, 0.5 \rangle$. For $t = \langle 0.5, 0.5, 0.5 \rangle$, $DR(t; \mathcal{L}_2^{\mathcal{C}})$ is characterized by:*

$$\{ \quad s[A_1]^2 \geq 0.25, \qquad s[A_2]^2 \geq 0.25, \\ s[A_1]^2 + s[A_3]^2 \geq 0.5, \quad s[A_2]^2 + s[A_3]^2 \geq 0.5 \quad \}. \qquad (16)$$

Therefore, tuple $t' = \langle 0.7, 0.5, 0.3 \rangle$ is not $\mathcal{L}_2^{\mathcal{C}}$-dominated by t, as the last inequality in (16) is not satisfied. See Figure 2 for a graphical representation.

The only significant overhead introduced by this approach is the enumeration of the vertices of $\mathcal{W}(\mathcal{C})$. However, due to the above observation, this has to be done just once.

As with Theorem 7, even Theorem 8 also holds for the family of linear functions wrt. the weights.

4.2 Computing potentially optimal tuples

We observe that, for any set $\mathcal{F}$, $\text{PO}(r; \mathcal{F})$ can be computed starting from $\text{ND}(r; \mathcal{F})$ by retaining only the tuples that are not $\mathcal{F}$-dominated by any "virtual" tuple obtained by combining other tuples in $\text{ND}(r; \mathcal{F})$. When $\mathcal{F}$ belongs to the $\mathcal{L}_p^{\mathcal{C}}$ family with linear constraints $\mathcal{C}$, this can be done again efficiently by solving an LP problem.

THEOREM 11 (POTENTIAL OPTIMALITY TEST). *Let p be a finite positive integer, $\mathcal{C}$ a set of linear constraints on weights. Let $W^{(1)}, \ldots, W^{(q)}$ be the vertices of $\mathcal{W}(\mathcal{C})$ and let $\text{ND}(r; \mathcal{L}_p^{\mathcal{C}}) = \{t_1, t_2, \ldots, t_\sigma, t\}$. Then, $t \in \text{PO}(r; \mathcal{L}_p^{\mathcal{C}})$ iff there is no convex combination s of $t_1, \ldots, t_\sigma$ such that $s \prec_{\mathcal{L}_p^{\mathcal{C}}} t$, i.e., iff the following linear system in the unknowns $\alpha = (\alpha_1, \ldots, \alpha_\sigma)$ is unsatisfiable:*

$$\begin{array}{rl} \sum_{i=1}^{d} w_i^{(\ell)} (\sum_{j=1}^{\sigma} \alpha_j t_j[A_i]^p) \leq \sum_{i=1}^{d} w_i^{(\ell)} t[A_i]^p & \\ \ell \in \{1, \ldots, q\} & (17) \\ \alpha_j \in [0,1] \quad j \in \{1, \ldots, \sigma\} & \\ \sum_{j=1}^{\sigma} \alpha_j = 1. & \end{array}$$

As with Theorems 7 and 8, even Theorem 11 also holds for *any* set $\mathcal{F}$ whose functions are weighted sums of monotone functions of single attributes and monotonic transforms thereof.

Algorithm 1: SLP2 and SVE2 for ND.

Input: *relation* r, *constraints* $\mathcal{C}$, *family* $\mathcal{F} = \mathcal{L}_p^{\mathcal{C}}$. Output: $\text{ND}(r; \mathcal{F})$.

1. Prepare
2. **let** $\mathcal{S} := \text{SKY}(r)$ *// phase one*
3. **for each** s **in** $\mathcal{S}$ *// candidate $\mathcal{F}$-dominated tuple*
4. **if** SVE2 **then** compute left-hand sides of Inequalities (13)
5. **for each** t **in** ND *// candidate $\mathcal{F}$-dominant tuple*
6. **if** $t \prec_{\mathcal{F}} s$ **then continue** to line 3
7. **let** ND := ND $\cup \{s\}$
8. **return** ND

Subprocedure: Prepare

9. **let** $W^{(1)}, \ldots, W^{(q)}$ be the vertices of $\mathcal{W}(\mathcal{C})$
10. sort r using the coordinates of the centroid of $\mathcal{W}(\mathcal{C})$ as weights
11. **let** ND := $\emptyset$

Algorithm 2: SVE1 for ND.

Input: *relation* r, *constraints* $\mathcal{C}$, *family* $\mathcal{F} = \mathcal{L}_p^{\mathcal{C}}$. Output: $\text{ND}(r; \mathcal{F})$.

1. Prepare *// same as in Algorithm 1*
2. **for each** s **in** r *// candidate $\mathcal{F}$-dominated tuple*
3. **for each** t **in** ND *// candidate $\mathcal{F}$-dominant tuple*
4. **if** $t \prec s$ **then continue** to line 2
5. compute left-hand sides of Inequalities (13)
6. **for each** t **in** ND *// candidate $\mathcal{F}$-dominant tuple*
7. **if** $t \prec_{\mathcal{F}} s$ **then continue** to line 2
8. **let** ND := ND $\cup \{s\}$
9. **return** ND

Algorithm 3: SVE1F for ND.

Input: *relation* r, *constraints* $\mathcal{C}$, *family* $\mathcal{F} = \mathcal{L}_p^{\mathcal{C}}$. Output: $\text{ND}(r; \mathcal{F})$.

1. Prepare *// same as in Algorithm 1*
2. **for each** s **in** r *// candidate $\mathcal{F}$-dominated tuple*
3. compute left-hand sides of Inequalities (13)
4. **for each** t **in** ND *// candidate $\mathcal{F}$-dominant tuple*
5. **if** $t \prec s \vee t \prec_{\mathcal{F}} s$ **then continue** to line 2
6. **let** ND := ND $\cup \{s\}$
7. **return** ND

5. ALGORITHMS

Based on different options for computing ND and PO, we consider several algorithmic alternatives.

5.1 Computing ND

Sorting. The first option regards whether to sort the dataset beforehand to produce a topological sort with respect to the $\mathcal{F}$-dominance relation, similarly to what the SFS algorithm [4] does to compute SKY. The sorting function we adopt is a weighted sum using as weights the coordinates of the centroid of the polytope $\mathcal{W}(\mathcal{C})$ determined by the constraints in the space of weights. This choice guarantees that if tuple t precedes tuples s in the sorted relation, then $s \not\prec_{\mathcal{F}} t$. In the following, the letter "S" in the algorithm's name will indicate that sorting is used, "U" that the dataset is unsorted.

$\mathcal{F}$-dominance. Another option regards the $\mathcal{F}$-dominance test. The alternatives for testing whether $s \prec_{\mathcal{F}} t$ are: *i)* solving an LP problem as in Theorem 7 (indicated by "LP" in the algorithm's name); *ii)* checking whether $t \in DR(s; \mathcal{F})$ (i.e., the $\mathcal{F}$-dominance region of s) as in Theorem 8 through vertex enumeration of the polytope ("VE" in the name).

Phases. We can also choose whether the computation of ND should be applied after computing SKY (i.e., in two phases, indicated by "2" in the algorithm's name) or whether dominance and $\mathcal{F}$-dominance tests should be integrated (i.e., in one phase, "1" in the algorithm's name).

So far, there are 8 alternatives. We start by describing the 4 2-phase alternatives (ULP2, UVE2, SLP2, SVE2).

The pseudocode for the sorted variants SLP2 and SVE2 is shown in Algorithm 1: the main idea is to scan the tuples sortedly and to populate a current window ND of non-dominated tuples among those that are in $\text{SKY}(r)$; $\text{SKY}(r)$ is computed via the SFS algorithm, since the dataset is sorted. Thanks to sorting, no tuple will ever be removed from ND (no tuple can be $\mathcal{F}$-dominated by a tuple found later in the sorted relation). The vertices of the polytope $\mathcal{W}(\mathcal{C})$ are computed just once (line 9). We enumerate sortedly every candidate $\mathcal{F}$-dominated tuple s (line 3) and compare it against every candidate $\mathcal{F}$-dominant tuple t (line 6) to decide whether s should be added to ND. The $\mathcal{F}$-dominance test of line 6 is done via Theorem 7 for SLP2 and via Theorem 8 for SVE2. In the latter case, it is useful to precompute the left-hand sides of Inequality (13) already at line 4.

The unsorted counterparts ULP2 and UVE2 are similar, but, without sorting, *i)* we cannot compute $\text{SKY}(r)$ via SFS, and thus use the classical BNL algorithm [1], and *ii)* when a tuple s is added to ND, other tuples in ND may be $\mathcal{F}$-dominated by s, and thus need to be removed (also the second phase of ULP2 behaves essentially as BNL, but with $\mathcal{F}$-dominance instead of dominance tests).

Since the above described algorithms adopt in the first phase either SFS or BNL for computing $\text{SKY}(r)$, it follows that: *i)* for the unsorted variants ULP2 and UVE2, multiple passes over the datasets may be required depending on the available memory space, as in BNL; *ii)* for the sorted variants SLP2 and SVE2, multiple passes are needed only if the size of SKY exceeds the memory space, as in SFS.

As will be shown in Section 6.2, S strategies are faster than U strategies (except perhaps for small datasets), and VE is orders of magnitude faster than LP. Therefore, we shall consider 1-phase counterparts for SVE2 only. The pseudocode of SVE1 is shown in Algorithm 2. Instead of first computing SKY and then carving ND out of it, SVE1 first tries to discard the candidate $\mathcal{F}$-dominated tuple s by using only the easier dominance tests (lines 3–4) against the non-$\mathcal{F}$-dominated tuples in ND; only if all such tests fail, the harder $\mathcal{F}$-dominance (lines 6–7) tests are executed.

The last 1-phase alternative we consider (Algorithm 3) interleaves dominance and $\mathcal{F}$-dominance tests, thus performing, for each new tuple s, a single pass over ND (and is thus denoted SVE1F since $\mathcal{F}$-dominance is checked first, before moving to the next tuple in ND). The rationale behind SVE1F is that, for those cases in which $\mathcal{F}$-dominance is much more effective in pruning tuples than simple dominance, this approach can lead to saving many dominance tests wrt. SVE1, although perhaps attempting more $\mathcal{F}$-dominance tests.

Note that both SVE1 and SVE1F require multiple passes over the dataset only if ND does not fit in main memory.

5.2 Computing PO

For the computation of $\text{PO}(r; \mathcal{F})$, we start from the tuples in $\text{ND}(r; \mathcal{F})$ and, by Theorem 11, we discard any tuple t that is $\mathcal{F}$-dominated by a convex combination of tuples in $\text{ND}(r; \mathcal{F}) \setminus \{t\}$. However, directly checking $\mathcal{F}$-dominance via (17) may be prohibitively time consuming when $\sigma = |\text{ND}(r; \mathcal{F})| - 1$ is large. We therefore try, in Algo-

Algorithm 4: POND for PO.

Input: *relation* r, *constraints* $\mathcal{C}$, *family* $\mathcal{F} = \mathcal{L}_p^{\mathcal{C}}$. Output: $\text{PO}(r;\mathcal{F})$.
1. **let** PO := $\text{ND}(r;\mathcal{F})$ // *including* Prepare *as in Algorithm 1*
2. **let** $\tilde{\sigma} := 2$; **let** *lastRound* := **false**
3. **while**($\neg$*lastRound*)
4. **if** $\tilde{\sigma} \geq |\text{PO}| - 1$ **then** *lastRound* := **true**
5. **for each** t **in** PO in reverse order // *candidate $\mathcal{F}$-dominated tuple*
6. **if** $\exists s.\ s \prec_{\mathcal{F}} t, s$ is convex comb. of the first $\min(\tilde{\sigma}, |\text{PO}| - 1)$ tuples in $\text{PO} \setminus \{t\}$ **then let** $\text{PO} := \text{PO} \setminus \{t\}$
7. **let** $\tilde{\sigma} := \tilde{\sigma} \cdot 2$
8. **return** PO

Table 3: Time complexity of algorithms for computing ND and PO.

algorithm	first phase	second phase
ULP2	$\mathcal{O}(N^2)$	$\mathcal{O}(\lvert\text{SKY}\rvert^2 \cdot \mathsf{lp}(c,d))$
UVE2	$\mathcal{O}(N^2)$	$\mathcal{O}(\mathsf{ve}(c) + \lvert\text{SKY}\rvert^2 \cdot q)$
SLP2	$\mathcal{O}(N \cdot (\log N + \lvert\text{SKY}\rvert))$	$\mathcal{O}(\lvert\text{SKY}\rvert \cdot \lvert\text{ND}\rvert \cdot \mathsf{lp}(c,d)))$
SVE2	$\mathcal{O}(N \cdot (\log N + \lvert\text{SKY}\rvert))$	$\mathcal{O}(\mathsf{ve}(c) + \lvert\text{SKY}\rvert \cdot \lvert\text{ND}\rvert \cdot q)$
SVE1, SVE1F	$\mathcal{O}(\mathsf{ve}(c) + N \cdot (\log N + \lvert\text{ND}\rvert \cdot q))$	
POND	$\mathcal{O}(\lvert\text{ND}\rvert \cdot \log \lvert\text{ND}\rvert \cdot \mathsf{lp}(q, \lvert\text{ND}\rvert))$	

rithm 4 (POND, i.e. PO via ND), to reduce as early as possible the set of candidate potentially optimal tuples (PO) by adopting the following heuristics: *i)* we start with a convex combination of only $\tilde{\sigma} = 2$ tuples (line 2), which will give rise to smaller, faster-to-solve systems (17) for testing $\mathcal{F}$-dominance; as long as $\tilde{\sigma} < |\text{PO}| - 1$, this condition is only sufficient for pruning, but not necessary; after each round, we double $\tilde{\sigma}$ (line 7); *ii)* we sortedly enumerate candidate $\mathcal{F}$-dominated tuples from PO in reverse order (line 5), as the worst tuples wrt. the ordering are the most likely to be $\mathcal{F}$-dominated; *iii)* using linear system (17), we check the existence of a convex combination of the *first* $\tilde{\sigma}$ tuples in PO (line 6), as they are the best wrt. the ordering and thus more likely to $\mathcal{F}$-dominate other tuples. After this early pruning, in the last round (enabled by line 4) all the remaining tuples are checked against a convex combination of all the other tuples still in PO, which is now a necessary and sufficient condition for pruning, as in Theorem 11.

5.3 Considerations about complexity

We provide details about the input-output worst-case complexity of our algorithms when both the number of tuples N and the number of constraints c vary. In order to remain parametric wrt. auxiliary problems we have to solve, namely vertex enumeration and $\mathcal{F}$-dominance via linear programming, we consider that they will be solved by algorithms whose worst-case complexity is in $\mathcal{O}(\mathsf{ve}(c))$ and $\mathcal{O}(\mathsf{lp}(x,y))$, respectively, where x is the number of LP inequalities and y is the number of variables in the LP problem. The vertex enumeration problem is NP-hard in general and it is not known whether for the special case of bounded polytopes (like $\mathcal{W}(\mathcal{C})$) an algorithm exists with PTIME input-output complexity. We also observe that, for any fixed value of d, the number of vertices q is at most $\mathcal{O}(c^{\lfloor d/2 \rfloor})$ (see [10] and references therein).

Table 3 summarizes our results. For brevity, we only discuss ULP2, SVE2, SVE1, SVE1F, and POND. For the 2-phase algorithms, the complexity of the first phase is that of the corresponding skyline algorithm [9]. In the second phase, ULP2 performs at most $\mathcal{O}(|\text{SKY}|^2)$ $\mathcal{F}$-dominance tests, each of which costs $\mathcal{O}(\mathsf{lp}(c,d))$. On the other hand, SVE2 first enumerates the vertices of $\mathcal{W}(\mathcal{C})$, which costs $\mathcal{O}(\mathsf{ve}(c))$, and then will perform at most $|\text{SKY}| \cdot |\text{ND}|$ $\mathcal{F}$-dominance tests using Theorem 8, each of which costs $\mathcal{O}(q)$, where q is the number of vertices of $\mathcal{W}(\mathcal{C})$. The worst-case complexity of SVE1 and SVE1F is the same, since, besides enumerating vertices of $\mathcal{W}(\mathcal{C})$ and sorting the dataset, both execute $\mathcal{O}(N \cdot |\text{ND}|)$ $\mathcal{F}$-dominance tests. From the comparison between SVE2 and SVE1 (and SVE1F), we argue that the larger the skyline, the more SVE2 will be penalized.

The POND algorithm will execute the loop at most $\lceil \log_2 |\text{ND}| \rceil$ times, which happens in the very unlikely case in which most of the tuples in $\text{ND} \setminus \text{PO}$ appear before those in PO in the ordering; at each iteration, POND will execute at most $|\text{ND}|$ $\mathcal{F}$-dominance tests, the cost of which is bounded by $\mathcal{O}(\mathsf{lp}(q, |\text{ND}|))$, from which the result follows.

In terms of space, besides that needed by the 2-phase approaches to store the intermediate result SKY, and that required by the ND window, the only additional overhead is introduced by the specific procedures used to enumerate vertices and test $\mathcal{F}$-dominance by LP. Notice that the input to vertex enumeration is a set of c constraints, whereas the output is a set of q vertices. On the other hand, the largest LP problem (corresponding to the last round of the POND algorithm) will be a matrix of size at most $q \times (|\text{ND}| - 1)$.

6. EXPERIMENTS

In this section we aim to assess the efficiency of the various algorithmic alternatives for computing R-skylines, and to understand how R-skylines compare to both skylines and ranking queries. For a comprehensive analysis, we measure efficiency and effectiveness in a number of different scenarios, and study in particular how they are affected by *i)* data distribution, *ii)* dataset size, *iii)* number of dimensions, and *iv)* number of constraints. The relevant parameters are shown in Table 4, with defaults in bold.

Table 4: Operating parameters for performance evaluation (defaults, when available, are in bold).

Full name	Tested value
Distribution	synthetic: ANT, UNI; real: NBA, HOU
Synthetic dataset size (N)	10K, 50K, **100K**, 500K, 1M
# of dimensions (d)	2, 4, **6**, 8, 10
# of constraints (c)	1, 2, 3, 4, 5 (default: **d/2**)
Parameter of L_p norm (p)	**1**, 2, 3, 4, 5

6.1 Datasets and constraints

We use two families of datasets: synthetic datasets and real datasets. Synthetic datasets are generated by the standard data generation tool used in [1]. For any value of d and N mentioned in Table 4, we produced two d-dimensional datasets of size N with values in the $[0,1]$ interval; one of these datasets (UNI) has values distributed uniformly in $[0,1]$, while the other (ANT) has values anti-correlated across different dimensions – informally, points that are good in one dimension are bad in one or all of the other dimensions. In the interest of space, we do not include correlated synthetic datasets in our study, as they are the least challenging when it comes to computing skyline points.

The real datasets analyzed here are two well-known datasets used in the context of skylines.

The first one (NBA), reports statistics for each player in each game regarding NBA seasons from 2008 to 2015. [3] We selected 10 attributes with a clear numeric semantics for our experiments, including offensive rating, defensive rating and other measures of players' performance. Although irrelevant to the experiments, for coherence with the conventions used in this paper, all these values have been normalized in the $[0, 1]$ interval, 0 being the best value. After cleaning entries with null values, the dataset consisted of 190862 points. Figure 3 offers a representation of the notions of R-skylines on the NBA dataset when offensive and defensive ratings are the only dimensions and there is a constraint stating that the former weighs more than the latter: the skyline consists of ten tuples (C. Anthony occurs for two different games), three of which are potentially optimal (shown as green diamonds, including NBA superstar LeBron James), and four of which are non-$\mathcal{F}$-dominated but not potentially optimal (orange triangles); the $\mathcal{F}$-dominance regions are shown in gray and the slope of the diagonal lines on their border is $-45°$.

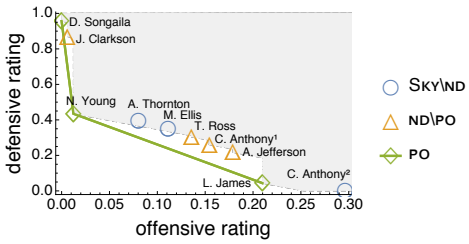


Figure 3: NBA dataset, $d = 2$, $w_1 \geq w_2$: SKY, ND and PO.

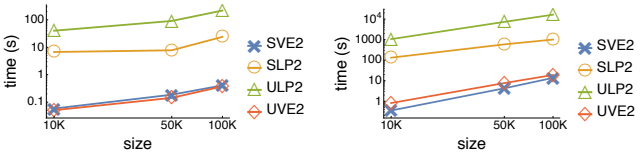
The second dataset (HOU) consists of 127931 6-dimensional points regarding household data scraped from www.ipums.org. The HOU dataset shows a higher correlation and has a limited number of dimensions; being less challenging than the other datasets, we shall only report results about HOU in the text, but not in the figures.

R-skylines are based on the notion of $\mathcal{F}$-dominance, which requires the definition of a set of monotone scoring functions $\mathcal{F}$. In the previous sections, we discussed how to characterize $\mathcal{F}$ by means of constraints on the space of weights applied to the family $\mathcal{L}_p$ of weighted L_p norms. For our experiments, we consider one of the most common types of constraints on weights: *weak rankings* (see, e.g., [6] for an overview of useful constraints). In particular, for any number c of constraints mentioned in Table 4 (with $c < d$), we considered the following set: $\{w_i \geq w_{i+1} | i \in \{1, \ldots, c\}\}$. We omit the results concerning other types of common constraints, as they show trends similar to weak rankings. Moreover, we study the effect of varying p on weighted L_p norms.

6.2 Results on efficiency

We assess efficiency of the different algorithms for computing ND by measuring, in a number of different scenarios, *i)* execution time (as measured on a machine sporting a 2.2 GHz Intel Core i7 with 16 GB of RAM), *ii)* number of dominance tests, *iii)* number of $\mathcal{F}$-dominance tests. For computing PO, we only report the execution time. The "windows" storing partial results were implemented as plain arrays, with no particular indexing structure built on the fly.

[3] Available at http://www.quantifan.com.



(a) Uniform distribution. (b) Anti-correlated distribution.

Figure 4: Performance of 2-phase algorithms for ND on smaller-scale datasets as the size N varies.

All LP problems were solved by integrating our system with the linear programming tool lp_solve.[4] In the worst-case scenario ($d = 10$, $c = 5$), solving the LP problems in Theorem 7 required $0.24ms$ on average. In the same scenario, it took $3.7ms$ on average to check $\mathcal{F}$-dominance with respect to a convex combination of tuples (as in line 6 of Algorithm 4); in the worst case, the convex combination included 2198 variables (i.e., tuples), requiring $39ms$. By monitoring actual occupation, we observed that lp_solve never required more than 32 MB of memory.

Vertex enumeration, relevant for Theorem 8, was performed with the lrs tool,[5] which never required more than $18ms$ to complete; in all the polytopes analyzed in our experiments, there were never more than 10 vertices. Observe that vertex enumeration needs to be done only once during the entire computation of ND. The memory footprint of lrs never exceeded 44 MB.

As a common trend to all the analyzed variants, execution times increase as N and/or d grow. Also, times decrease as c grows, because more constraints make it more likely to discard a tuple quickly and thus to reduce the problem size.

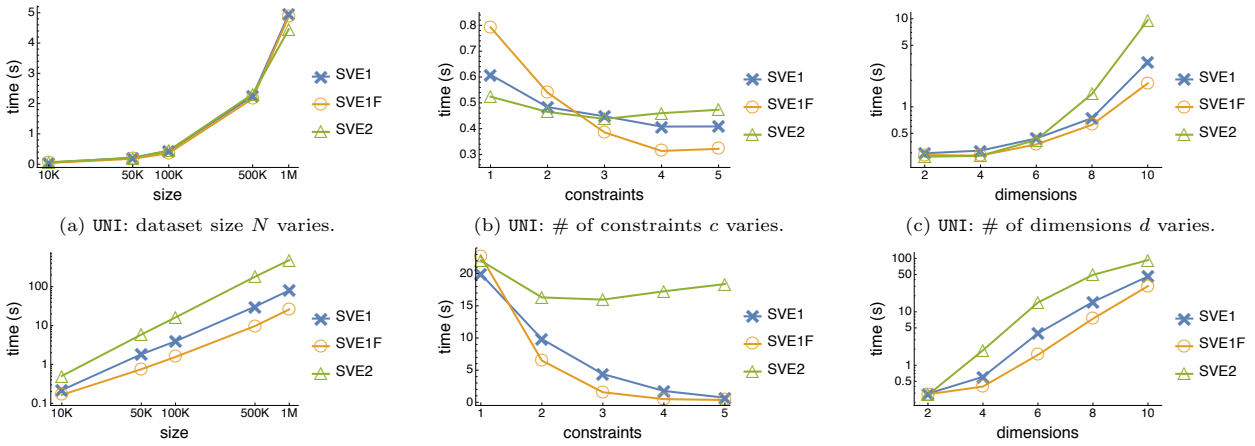
Computing ND. We first assess all 2-phase variants (ULP2, UVE2, SLP2, SVE2) for smaller-scale synthetic datasets with varying size up to 100K tuples. Figure 4 shows that ULP2 and SLP2 are clearly outperformed by UVE2 and SVE2 by at least two orders of magnitude. This supports the intuition that led us to Theorem 8, since the high number of $\mathcal{F}$-dominance tests highly penalizes both ULP2 and SLP2; for instance, when $N = 10^5$, SLP2 performs up to 5.8 million tests on ANT, each requiring to solve a different LP problem.

Although comparable for less challenging datasets (UNI, Figure 4a), SVE2 prevails over UVE2 in the more difficult cases (ANT, Figure 4b). This is in line with what other observed when comparing sorted (i.e., SFS) vs. unsorted (i.e., BNL) skyline algorithms, which are used in the first phase of SVE2 and UVE2, respectively. As to the second phase, we observe that UVE2 requires many more $\mathcal{F}$-dominance tests than SVE2 (up to 6 times more tests on ANT). Furthermore, since sorting enables the application of the heuristics used for computing PO (Algorithm 4), in the following we therefore only consider SVE2 and its 1-phase counterparts SVE1 and SVE1F.

In the next set of experiments, we varied the dataset size N up to 1M tuples (see Figures 5a and 5d). Execution times of SVE2, SVE1 and SVE1F are almost the same on the simpler UNI datasets, whereas, when data are anti-correlated, SVE1F performs much better than SVE1 (second-best) and SVE2 (last). In order to understand this phenomenon, we analyze the number of dominance and $\mathcal{F}$-dominance tests executed by the algorithms, shown in Figures 6a and 6b for the ANT dataset. We observe that SVE1F performs at least

[4] http://lpsolve.sourceforge.net.

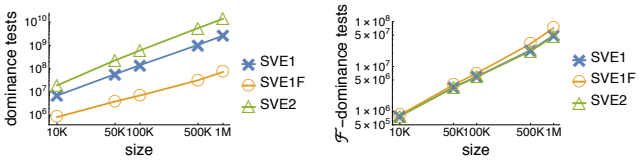
[5] http://cgm.cs.mcgill.ca/~avis/C/lrs.html.



(a) UNI: dataset size N varies. (b) UNI: # of constraints c varies. (c) UNI: # of dimensions d varies.

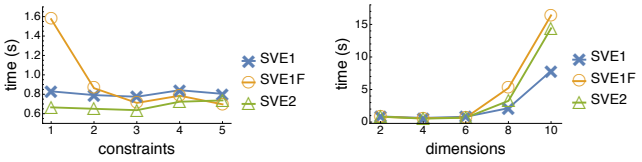
(d) ANT: dataset size N varies. (e) ANT: # of constraints c varies. (f) ANT: # of dimensions d varies.

Figure 5: Performance for computing ND. Distribution: UNI in (a)–(c); ANT in (d)–(f).



(a) ANT: dataset size N varies. (b) ANT: dataset size N varies.

Figure 6: Dominance (a) and $\mathcal{F}$-dominance (b) tests for computing ND on ANT.



(a) NBA: # of constraints c varies. (b) NBA: # of dimensions d varies.

Figure 7: Performance for computing ND on the real dataset NBA.

ten times less dominance tests than SVE1 and SVE2, while doing only a little more $\mathcal{F}$-dominance tests. The additional pruning capability of $\mathcal{F}$-dominance is tightly correlated to the constraints. To this end, for the default values $N = 10^5$ and $d = 6$, in Figures 5b and 5e we vary the number of constraints from $c = 1$ to $c = 5$. As the figures show, the relative performance of SVE1F monotonically improves as c grows, whereas with fewer constraints, i.e., smaller pruning, the number of additional $\mathcal{F}$-dominance tests is so high that the relatively small overhead incurred by SVE1F in anticipating (unsuccessful) $\mathcal{F}$-dominance tests becomes the heaviest factor in overall performance, and hence SVE1F is not the best choice for ANT when $c = 1$ and for UNI when $c \in \{1, 2\}$.

The behavior of the algorithms as the number of dimensions d grows is shown in Figures 5c and 5f. The relative performance of the algorithms remains unchanged, with SVE1F being, again, the most efficient regardless of d. Both 1-phase variants prevail over SVE2, since they do not have the burden of computing SKY as an intermediate step, which heavily increases its size as d grows (as an example, 75% of the tuples in ANT are in the skyline when $d = 10$, $N = 10^5$, and around 25% in UNI).

The experiments on the real NBA dataset confirm the previous observations. In particular, Figure 7a shows that, on NBA, the algorithms' trend is similar to that on UNI, i.e., SVE1F tends to improve its relative performance as the number of constraints grows. It is confirmed that SVE1F still incurs fewer dominance tests than the other alternatives; however the difference is smaller than in synthetic datasets, so SVE1F only prevails when $c = 5$. Since the relative size of ND(NBA; $\mathcal{F}$) wrt. SKY(NBA) is larger than for the synthetic datasets (see Table 5), the effectiveness of $\mathcal{F}$-dominance tests reduces (and consequently the relative effectiveness of SVE1F), which also favors the 2-phase SVE2 approach. This is made evident in Figure 7b, where performance of all approaches is similar for low dimensionalities ($d \leq 6$), and the 1-phase approach SVE1 pays off only for the more challenging cases with $d \in \{8, 10\}$.

Table 5: Cardinalities of SKY, ND, PO with default values; in brackets, the ratio with the cardinality of SKY.

Dataset	SKY	ND	PO
ANT	26637	2616 (9.8%)	271 (1%)
UNI	2626	445 (16.9%)	122 (4.6%)
NBA	1137	264 (23.2%)	32 (2.8%)
HOU	49	26 (53%)	19 (41.3%)

Computing PO. We measure the overall execution time incurred by POND for the different datasets when ND is computed via SVE1F. Given the intrinsic higher complexity of computing PO wrt. ND, from Figure 8 it can be argued that times are always acceptable for the UNI and NBA datasets (less than 10s in all cases but $d \geq 8$); ANT is harder to deal with when the problem size gets larger, because the starting ND set contains more tuples; execution times remain around 10s or less with default parameter values or easier combinations, i.e., $N \leq 100K$, $c \geq 3$ and $d \leq 6$.

The sufficient condition used by POND for early pruning proves very effective. For instance, in the ANT dataset with default parameter values, ND consists of 2616 tuples, 271 of which are in PO; POND removes 96% of the non-potentially-optimal tuples before the last round, and 43% in the first five (very quick) rounds, where combinations of at most 32 tuples are considered; once it gets to the last of 9 rounds, there are only 367 tuples left (14% of |ND|). Note that a single $\mathcal{F}$-

dominance test with a combination of 2615 tuples takes on average approximately 26 times as much as a test with 366 tuples, therefore the gains are conspicuous. Overall, POND completes in 11s, whereas directly computing PO using LP problem (17) requires 45s. Similar effects are measurable with the UNI (resp., NBA) dataset: with default parameter values, 89% (resp., 98%) of the non-potentially-optimal tuples are removed before the last round.

6.3 Comparison with skyline queries

In order to better understand the behavior of ND and PO, we first characterize them in terms of user results by considering how much they are able to reduce the size of the result as compared to standard skyline queries. In particular, we compute the ratio of points retained by these operators among those in the skyline (see also Table 5).

Figure 9 shows this ratio in several scenarios for the ANT, UNI and NBA datasets. The results indicate that R-skylines are always much more effective than skylines, and PO is more effective than ND, as expected. Table 5 shows that, for default values of the parameters, the effectiveness of R-skylines is already remarkable, with the largest reductions in ANT (9.8% of the skyline points are in ND and only 1% in PO).

Figures 9a and 9d show that the reduction of points increases as the size N of the dataset grows, the highest pruning being obtained in the ANT dataset with $N = 1M$ points; in this setting, the skyline has 99181 points, 6407 of which (6.5%) are retained by ND, and only 462 (0.46%) by PO. Although we observed this reduction increase in all our experiments, an analytic explanation is still missing and we expect this to be difficult to provide, as already for standard skylines the size estimation problem is open [8].

Figures 9b, 9e, and 9h show that the effectiveness of R-skylines steadily improves as the number of constraints c increases. This is mainly due to the fact that each constraint reduces the space of weights, and thus the set $\mathcal{F}$ of functions to consider for $\mathcal{F}$-dominance; the number of retained tuples decreases as a consequence of Proposition 2. The experiments also show that PO proves very effective even with few constraints, with the most dramatic improvement wrt. ND in the NBA dataset with 1 constraint, where the ratio decreases from 88.6% for ND to 6.2% for PO. The figures suggest a possible correlation between the reduction of the space of weights caused by constraints and the reduction of tuples caused by R-skylines. Figure 10 confirms a clear correlation between the ratio of points in ND and the ratio of preserved weight volume, i.e., the ratio between the (hyper-)volume of the polytope $\mathcal{W}(\mathcal{C})$ determined by the $|\mathcal{C}| = c$ constraints in the space of weights and the volume of the $(d-1)$-simplex.

Significant reduction is also observed when varying the number of dimensions, as shown in Figures 9c, 9f, and 9i. In all the scenarios with $d > 2$, the ratio of skyline points in ND is always below 35%, and much less in many cases, while for PO the ratio never exceeds 10% in these cases, reaching an impressive 0.38% when $d = 10$ for the NBA dataset.

Finally, Figure 9g shows the effect of parameter p of L_p norms (10) on NBA (the trends for the other datasets are similar and thus not reported). As p grows, the ratio grows slightly but steadily, although the effectiveness remains significant (in the worst case, the ratio is 35.2% for ND and 13.3% for PO when $p = 5$). This follows by observing that, for any tuple t and constraints $\mathcal{C} \neq \emptyset$, the dominance region $DR(t; \mathcal{L}^{\mathcal{C}}_p)$ strictly includes $DR(t; \mathcal{L}^{\mathcal{C}}_{p'})$ for $p' > p$.

Dataset	SKY by SFS	ND by SVE1F	PO by POND
ANT	11.17 (98%)	1.54	12.69
UNI	0.34 (94%)	0.36	1.41
NBA	0.58 (98%)	0.65	1.74

Table 6: Execution times (in seconds) of SKY, ND, PO with default values; in brackets, the percentage wrt. the execution time of SVE2.

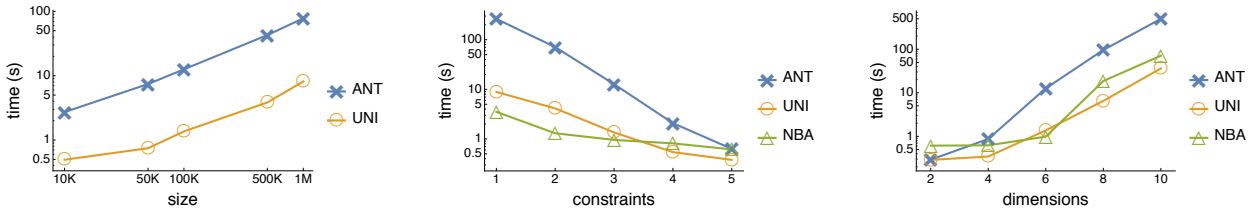
We now compare R-skylines and skylines in terms of execution times. Table 6 shows such times for default parameter values. It can be seen that computing SKY using SFS essentially requires the same time as computing ND with the 2-phase SVE2 approach (i.e., most of the time is spent in the first phase of the algorithm), a fact we observed in all the scenarios we tested. Therefore, computing SKY and ND via a 1-phase algorithm (like SVE1F) is in most cases in favor of the latter, possibly requiring even much less time, as can be seen in Table 6 for the ANT case.

Overall, we conclude that R-skylines are highly effective in reducing the size of the result without introducing any notable overhead wrt. the computation of standard skylines.

6.4 Comparison with ranking queries

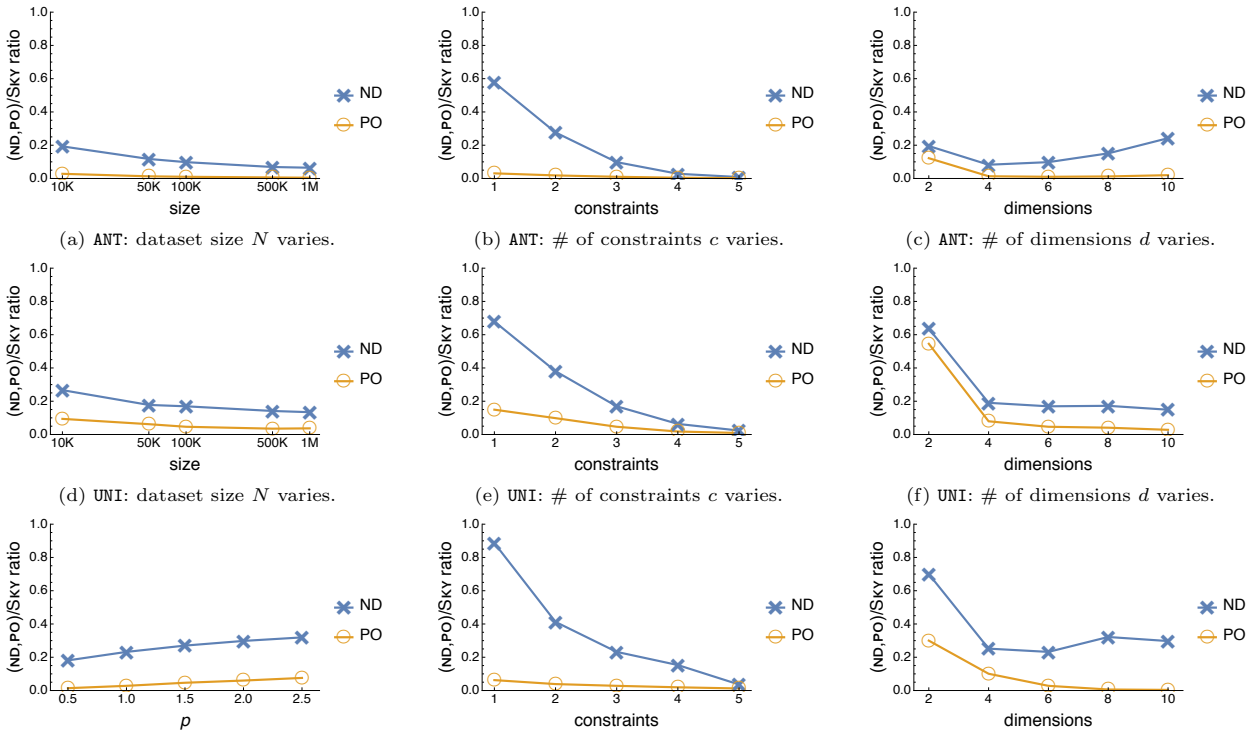
After having quantitatively analyzed the relationship between results of R-skylines and skylines, for which it is true that former ones are included in the latter, for completeness, we now turn to compare results yielded by our operators and those of ranking (top-k) queries. To this end, we consider the classical *precision* and *recall* measures. Let $\mathcal{T}_k(r; f)$ indicate the set of top-k tuples in relation r wrt. a scoring function f. The precision of $\mathcal{T}_k(r; f)$ wrt. a set $\mathcal{S}$ is defined as $\text{PRE}(\mathcal{S}) = |\mathcal{S} \cap \mathcal{T}_k(r; f)|/k$, whereas the recall is $\text{REC}(\mathcal{S}) = |\mathcal{S} \cap \mathcal{T}_k(r; f)|/|\mathcal{S}|$. Notice that, in our context, where $\mathcal{S} \in \{\text{SKY}, \text{ND}, \text{PO}\}$, high precision would indicate that most of the top-k tuples are also in $\mathcal{S}$, whereas a high recall would mean that most of the tuples in $\mathcal{S}$ are also top-k tuples. We first remark that, to the best of our knowledge, a detailed comparative analysis between the results of top-k and skyline queries has never been attempted before, and indeed it would be worth investigating to better understand how tuples of a dataset are distributed. Indeed, if there is a small overlap between top-k and skyline results, we expect a similar or smaller overlap with R-skylines. Moreover, the overlap will largely depend on the specific scoring function f, and, in the case of R-skylines, also on the family of scoring functions $\mathcal{F}$. Intuitively, the more f is "dissimilar" to $\mathcal{F}$, the more R-skyline and top-k query results will differ.

Figure 11 shows precision and recall when f is a weighted sum with the centroid of the polytope $\mathcal{W}(\mathcal{C})$ as weight vector. This choice guarantees that $f \in \mathcal{F}$ and ensures that f is, in some sense, the most representative function in $\mathcal{F}$. Clearly, recall values grow with k, while precision values decrease. For every value of k and all datasets, $\text{PRE}(\text{PO}) \leq \text{PRE}(\text{ND}) \leq \text{PRE}(\text{SKY})$, i.e., a top-$k$ query will more easily retrieve skyline tuples than non-$\mathcal{F}$-dominated and potentially optimal tuples. For instance, among the top-50 tuples in UNI, 47 are also in SKY, while 36 are in ND and only 13 in PO. As expected, PRE(ND) and PRE(PO) somehow depend on PRE(SKY) and thus on data distribution. As for recall, when k equals the number of tuples returned by an R-skyline operator (and thus precision and recall are equal), we have that the range of REC(ND) is between 38% (for UNI) and 50% (for ANT), while such values drop to 18%–38% for REC(PO). Note that, for retrieving all the 264 tuples



(a) dataset size N varies. (b) # of constraints c varies. (c) # of dimensions d varies.

Figure 8: Performance for computing PO with the POND algorithm.



(a) ANT: dataset size N varies. (b) ANT: # of constraints c varies. (c) ANT: # of dimensions d varies.

(d) UNI: dataset size N varies. (e) UNI: # of constraints c varies. (f) UNI: # of dimensions d varies.

(g) NBA: parameter p of L_p norm varies. (h) NBA: # of constraints c varies. (i) NBA: # of dimensions d varies.

Figure 9: Cardinality ratio between R-skylines (PO, ND) and SKY: UNI in (a)–(c); ANT in (d)–(f); NBA in (g)–(i).

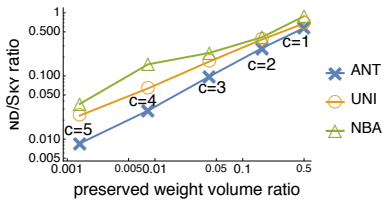
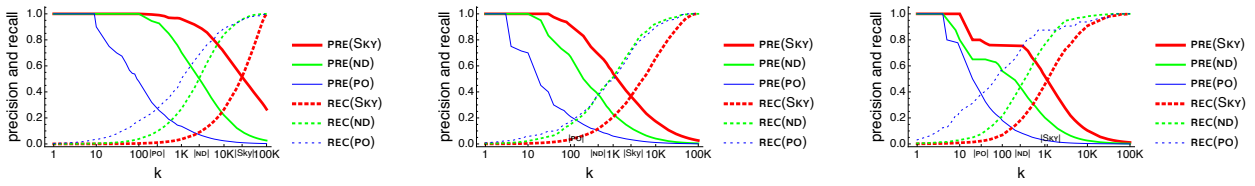


Figure 10: Correlation between ND/SKY cardinality ratio and percentage of preserved volume in the space of weights as the number of constraints c varies.



(a) ANT dataset. (b) UNI dataset. (c) NBA dataset.

Figure 11: Precision and recall of top-k queries wrt. SKY, ND, and PO. Centroid of the $\mathcal{W}(\mathcal{C})$ as weight vector; default parameter values.

in ND(NBA; $\mathcal{F}$) one should choose a value of $k > 30000$, and the same holds for the 32 tuples in PO(NBA; $\mathcal{F}$); even larger values of k are needed for ANT and UNI. As a further illustration of the difference of the results of R-skylines and top-k queries, we examined the ANT dataset when $d = 2$: here, a top-10 query retrieves only 4 out of the 8 tuples in ND, while all the other tuples in ND occur after position 1000. Note that the above results are the best possible for top-k queries, since using for f a weight vector other than the centroid of $\mathcal{W}(\mathcal{C})$ leads to lower precision as well as recall.

Similarly to skyline queries, R-skylines pay the increased capability of returning interesting results at the price of a higher computational overhead wrt. ranking queries. In all our experiments, in which the top-k algorithm was implemented using a max-heap along the lines described in [2], ranking queries required only a fraction of the execution time of SVE1F (between 0.1% and 7%), as expected.

7. DISCUSSION

In this paper, we have built a framework aiming to unify skyline and ranking queries. We have done so by introducing two R-skyline operators implementing the notions of non-dominated (ND) and potentially optimal (PO) tuples with respect to a set of scoring functions $\mathcal{F}$. The greater flexibility of these operators captures not only standard skyline and top-1 queries, but also constraints on the weights in the scoring functions, which are highly relevant in practice.

It is well known that choosing the "right" weights for a scoring function is a difficult task for users, since it is usually hard to predict the effects on ranking of changing one or more parameters. Replacing precise values with constraints on weights, as R-skylines do, is therefore a viable way to alleviate the problem. Besides methods already evaluated in the DB field (see, e.g., [23]), a large body of techniques have been investigated in Multi-Attribute Decision Theory (MADT), where the term "preference programming" has been coined to this purpose [18]. In general, preference programming methods deal with the problem of assisting a decision maker when her preferences are incomplete. R-skyline algorithms represent, as far as we know, the first successful attempt to apply a preference programming method on large datasets. It is remarkable that the only assumption used by our algorithms is that the (family of) scoring function(s) $\mathcal{F}$ is linear in the weight parameters. Besides L_p norms, one could therefore use any function of the form $f(t) = \tau(\sum_{i=1}^{d} w_i u_i(t[A_i]))$, where u_i is a monotone function of A_i values representing the "marginal utility" of attribute A_i and τ is a monotone transform.[6] Methods for choosing the "right" marginal utility functions abound in the MADT field (see, e.g., [19]), and have also been incorporated in some preference programming methods [18]. An interesting development of our work would be that of extending R-skyline algorithms to the more challenging scenario in which both weights and marginal utilities are partially specified.

We have shown that both ND and PO are very effective in focusing on tuples of interest, even in very large datasets. Overall, we have developed nine variants for efficiently computing ND and shown how they behave in a variety of scenarios, including both synthetic and real datasets. From a practical perspective, in most of these scenarios, computing ND requires no more than a few seconds. For the more complex problem of computing PO, we have developed heuristics that reduce the number of large LP problems to be solved.

Natural extensions of this framework include the notions of top-k query and k-skyband, with $k > 1$ [17].

Acknowledgments. D. Martinenghi acknowledges support from the H2020-EU.3.3.1 "ENCOMPASS" project (ID: 723059).

[6] With L_p norms, $u_i(t[A_i]) = t[A_i]^p$ and $\tau(x) = x^{1/p}$.

8. REFERENCES

[1] S. Börzsönyi, D. Kossmann, and K. Stocker. The skyline operator. In *ICDE*, pages 421–430, 2001.
[2] M. J. Carey and D. Kossmann. On saying "enough already!" in SQL. In *SIGMOD*, pages 219–230, 1997.
[3] J. Chomicki, P. Ciaccia, and N. Meneghetti. Skyline queries, front and back. *SIGMOD Record*, 42(3):6–18, 2013.
[4] J. Chomicki et al. Skyline with presorting. In *ICDE*, pages 717–719, 2003.
[5] E. Ciceri et al. Crowdsourcing for top-k query processing over uncertain data. *TKDE*, 28(1):41–53, 2016.
[6] Y. S. Eum, K. S. Park, and S. H. Kim. Establishing dominance and potential optimality in multi-criteria analysis with imprecise weight and value. *Computers & Operations Research*, 28:397–409, 2001.
[7] A. A. Freitas. A critical review of multi-objective optimization in data mining: a position paper. *SIGKDD Explorations*, 6(2):77–86, 2004.
[8] P. Godfrey. Skyline cardinality for relational processing. In *FoIKS*, pages 78–97, 2004.
[9] P. Godfrey et al. Algorithms and analyses for maximal vector computation. *VLDB J.*, 16(1):5–28, 2007.
[10] V. Kaibel and M. E. Pfetsch. Some algorithmic problems in polytope theory. In *Algebra, Geometry, and Software Systems*, pages 23–47, 2003.
[11] X. Lin et al. Selecting stars: The k most representative skyline operator. In *ICDE*, pages 86–95, 2007.
[12] C. Lofi and W. Balke. On skyline queries and how to choose from pareto sets. In *Advanced Query Processing, Volume 1: Issues and Trends*, pages 15–36. 2013.
[13] C. Lofi, U. Güntzer, and W. Balke. Efficient computation of trade-off skylines. In *EDBT*, pages 597–608, 2010.
[14] N. Meneghetti et al. Output-sensitive evaluation of prioritized skyline queries. In *SIGMOD*, pages 1955–1967, 2015.
[15] D. Mindolin and J. Chomicki. Preference elicitation in prioritized skyline queries. *VLDB J.*, 20(2):157–182, 2011.
[16] K. Mouratidis, J. Zhang, and H. Pang. Maximum rank query. *PVLDB*, 8(12):1554–1565, 2015.
[17] D. Papadias et al. Progressive skyline computation in database systems. *TODS*, 30(1):41–82, 2005.
[18] A. Salo and R. P. Hämäläinen. Preference programming multicriteria weighting models under incomplete information. In *Handbook of Multicriteria Analysis*, chapter 4, pages 167–187. Springer, 2010.
[19] Y. Siskos, E. Grigoroudis, and N. F. Matsatsinis. *UTA Methods*, pages 297–334. Springer New York, 2005.
[20] M. A. Soliman et al. Ranking with uncertain scoring functions: semantics and sensitivity measures. In *SIGMOD*, pages 805–816, 2011.
[21] Y. Tao et al. Distance-based representative skyline. In *ICDE*, pages 892–903, 2009.
[22] M. L. Yiu and N. Mamoulis. Efficient processing of top-k dominating queries on multi-dimensional data. In *VLDB*, pages 483–494, 2007.
[23] H. Yu et al. Enabling ad-hoc ranking for data retrieval. In *ICDE*, pages 514–515, 2005.
[24] J. Zhang, K. Mouratidis, and H. Pang. Global immutable region computation. In *SIGMOD*, pages 1151–1162, 2014.