Flexible Skylines: Dominance for Arbitrary Sets of Monotone Functions

Flexible Skylines: Dominance for Arbitrary Sets of Monotone Functions

PAOLO CIACCIA, Università di Bologna, Italy
DAVIDE MARTINENGHI, Politecnico di Milano, Italy

Skyline and ranking queries are two popular, alternative ways of discovering interesting data in large datasets. Skyline queries are simple to specify, as they just return the set of all non-dominated tuples, thereby providing an overall view of potentially interesting results. However, they are not equipped with any means to accommodate user preferences or to control the cardinality of the result set. Ranking queries adopt, instead, a specific scoring function to rank tuples, and can easily control the output size. While specifying a scoring function allows one to give different importance to different attributes by means of, e.g., weight parameters, choosing the "right" weights to use is known to be a hard problem.

In this paper we embrace the skyline approach by introducing an original framework able to capture user preferences by means of constraints on the weights used in a scoring function, which is typically much easier than specifying precise weight values. To this end, we introduce the novel concept of $\mathcal{F}$*-dominance*, i.e., dominance with respect to a *family* of scoring functions $\mathcal{F}$: a tuple t is said to $\mathcal{F}$-dominate tuple s when t is always better than or equal to s according to *all* the functions in $\mathcal{F}$.

Based on $\mathcal{F}$-dominance, we present two *flexible skyline* (F-skyline) operators, both returning a subset of the skyline: ND, characterizing the set of non-$\mathcal{F}$-dominated tuples; PO, referring to the tuples that are also potentially optimal, i.e., best according to some function in $\mathcal{F}$. While ND and PO coincide and reduce to the traditional skyline when $\mathcal{F}$ is the family of all monotone scoring functions, their behaviors differ when subsets thereof are considered. We discuss the formal properties of these new operators, show how to implement them efficiently, and evaluate them on both synthetic and real datasets.

CCS Concepts: • **Information systems → Database query processing**.

Additional Key Words and Phrases: Skyline queries, Ranking queries, Monotone functions

ACM Reference Format:
Paolo Ciaccia and Davide Martinenghi. 2020. Flexible Skylines: Dominance for Arbitrary Sets of Monotone Functions. *ACM Trans. Datab. Syst.* 1, 1 (June 2020), 61 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Determining the most interesting objects in a dataset is a fundamental task in many modern data-intensive scenarios, including data mining and database systems. When many, possibly conflicting, criteria (such as those represented by the different attributes of the tuples in a dataset) have to be simultaneously met in the best possible way according to the user's preferences, a problem known as *multi-objective optimization*, three main approaches are usually considered [20]:

Authors' addresses: Paolo Ciaccia, Università di Bologna, Dipartimento di Informatica - Scienza e Ingegneria (DISI), Viale Risorgimento, 2, Bologna, 40136, Italy, paolo.ciaccia@unibo.it; Davide Martinenghi, Politecnico di Milano, Dipartimento di Elettronica, Informazione e Bioingegneria (DEIB), Via Ponzio, 34/5, Milan, 20122, Italy, davide.martinenghi@polimi.it.

https://doi.org/10.1145/nnnnnnn.nnnnnnn

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

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

- The *lexicographic* approach [27], in which a strict priority among the attributes is established. Thus, if a tuple is better than another on the most important attribute, all other attributes play no role in determining the relative order of the two tuples.
- The *ranking queries* (or top-k) approach [25], where 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 both to adjust scales and to accommodate the importance that the user gives to the different attributes.
- The *skyline* approach [10], which returns all the non-dominated tuples (tuple t dominates tuple s iff t is no worse than s on all the attributes, and strictly better on at least one).

As argued in [20] and virtually in all papers focusing on a specific approach, each of these methods has pros and cons (also refer to Table 1). For instance, consider a user who wants to buy a used car. The relevant attributes for an informed decision are, say, price and mileage (as in Figure 1), and the user cares more about the former than the latter. 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. In our case, cars would be ordered by price, with mileage only used to break ties. The result of a ranking query heavily depends on how user preferences are translated into a particular choice of weights in the scoring function. This is a difficult task, and it is usually hard to predict the effects on ranking of changing one or more parameters. For instance, the best car according to the scoring function $0.7 \cdot \texttt{Price} + 0.3 \cdot \texttt{Mileage}$ is C1, whereas by slightly changing the weights, e.g., by using the scoring function $0.6 \cdot \texttt{Price} + 0.4 \cdot \texttt{Mileage}$, the best car would be C4. Finally, skyline queries provide a good overview of potentially interesting results, but may contain too many tuples to choose from and offer no way to express user preferences (since all attributes have the same importance). In our example, the skyline consists of five cars: C1, C2, C4, C6, and C7.

In this paper we embrace the skyline approach and aim to introduce a novel, flexible framework able to capture user preferences by means of constraints. As a beneficial side effect, this also entails a reduction of the result size with respect to the standard skyline. Our approach, which can somehow be regarded as the incorporation of ranking principles into skyline queries, is quite different from that of *prioritized skyline* (p-skyline) queries [32], in which user preferences are integrated by introducing a partial order of priorities between attributes. Indeed, p-skylines, which essentially combine lexicographic and skyline queries, inherit a problem common to both, in that they allow no trade-off between attributes (see Table 1). For instance, with reference to Figure 1, assuming again a preference for low prices, a p-skyline query would return only car C1, regardless of its high mileage (note that C1 would still be the only p-skyline result if its mileage were, say, 10 times larger and its price only slightly cheaper than other cars).

Based on the concepts of skyline and scoring functions, in this paper we introduce the notion of *flexible skyline* (F-skyline) queries. Similarly to p-skylines, F-skylines can take into account the different importance of different attributes. However, unlike p-skylines, in which a strict priority between attributes is assumed, F-skylines can model user preferences by means of constraints on

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

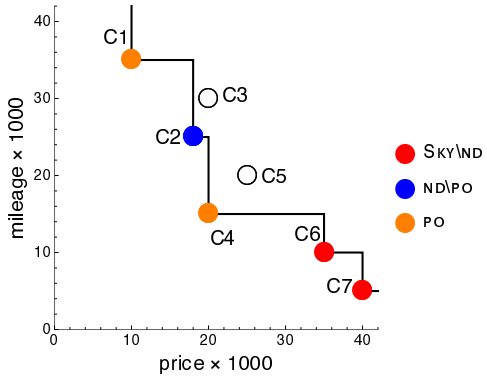


Fig. 1. The UsedCars relation from Example 1.1 and a graphical representation of its tuples. With scoring functions of form $\mathcal{F} = \{w_P\texttt{Price} + w_M\texttt{Mileage} \mid w_P \geq w_M\}$, C1, C2, and C4 are non-$\mathcal{F}$-dominated and C1 and C4 are also potentially optimal. The thick contour separates the cars in the skyline (represented as filled circles) from the other cars (empty circles).

the weights used in a scoring function, thus allowing for a much greater flexibility. Furthermore, F-skylines may consider arbitrary (typically infinite) families of monotone scoring functions. Overall, this leads to the novel concept of $\mathcal{F}$*-dominance*: 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 function in $\mathcal{F}$).

We present two F-skyline operators, both returning a subset of the skyline: ND, characterizing the set of *non-$\mathcal{F}$-dominated* tuples; PO, referring to the tuples that are *potentially optimal*, i.e., the only best option according to some function in $\mathcal{F}$. While ND and PO coincide and reduce to the traditional skyline when $\mathcal{F}$ is the family of all monotone scoring functions (in which case $\mathcal{F}$-dominance reduces to standard dominance), their behaviors differ when subsets thereof are considered, with PO always being a subset of ND.

Example 1.1. Consider the relation UsedCars(CarID,Price,Mileage) shown in Figure 1, for which the skyline consists of cars C1, C2, C4, C6, and C7. With the F-skyline framework, we can translate the preference for cheaper cars into a *constraint* on weights of a scoring function, e.g., by focusing on the family $\mathcal{F} = \{w_P\texttt{Price} + w_M\texttt{Mileage} \mid w_P \geq w_M\}$, in which price weighs more than mileage, since $w_P \geq w_M$. In such a case, the set of non-$\mathcal{F}$-dominated cars, i.e., ND, includes C1, C2, and C4, whereas C6 and C7 are both $\mathcal{F}$-dominated by C4, which is reasonable since they both have a relatively high price. Out of the tuples in ND, only C1 and C4 are also part of PO; indeed C1 is the best option when, say, $w_P = 0.9$ and $w_M = 0.1$, while C4 wins when weights are more comparable, e.g., when $w_P = 0.6$ and $w_M = 0.4$. Also note that car C2 is non-$\mathcal{F}$-dominated, but not potentially optimal, since there is no combination of weight values satisfying the constraint $w_P \geq w_M$ making it a top-1 result.

We advocate the use of constraints on weights as a practical way to accommodate user preferences into the skyline framework, thus somehow preserving the simplicity of formulation typical of skyline queries. Interestingly, this policy is adopted in the field of Multi-Attribute Decision Theory, where it commonly goes under the banner of "preference programming" (see, e.g., [39]), and several techniques have been developed to assist decision makers in eliciting their preferences. Besides constraints such as the one used in Example 1.1 (also called a *weak ranking* [17]), user's feedback on pairs of alternatives can be fruitfully exploited to issue suitable constraints [38]. For

instance, a preference for C4 over C6 induces the constraint $w_P \cdot 20 + w_M \cdot 15 \leq w_P \cdot 35 + w_M \cdot 10$, i.e., $w_P \geq w_M/3$; if such a constraint held instead of the one used in Example 1.1, then C7 would belong to both ND and PO (while C6 would still belong to neither). Another relevant scenario for F-skyline applicability is when weights of a scoring function are learned via crowdsourcing tasks (see, e.g., [14] and references therein for strategies for collecting preferences between tuples), in which case the remaining uncertainty can still be modeled as constraints (e.g., *interval constraints* around an average value, also known as *fixed bounds* [17]). We observe that our framework can generally deal with arbitrary (linear) constraints on weights, thus being able to express complex preferences such as "attribute C is more important than attribute A, but no more than twice as important", which might be encountered by decision makers in multicriteria analysis [39].

With respect to the other criteria in Table 1, F-skylines clearly allow one to establish a trade-off among different attributes. They inherit from skyline queries the capability of providing an overall view of interesting results, but can now focus on specific parts of the skyline, depending on the user preferences. For instance, the constraint $w_P \geq w_M$ in Example 1.1 allows one to focus on the upper-left part of the skyline, whereas $w_M \geq w_P$ would concentrate on the lower-right part. Constraints are also effective in controlling the cardinality of the result, albeit implicitly, in that the tighter the constraints, the more restricted the result set.

After precisely characterizing the core notions used for defining the ND and PO operators (namely, $\mathcal{F}$-dominance and $\mathcal{F}$-dominance region), in Section 3 we study the basic properties of F-skylines and their relationship with traditional skylines, as well as their behavior as $\mathcal{F}$ and/or the input dataset change. In Section 4 we address the problem of how to efficiently compute ND and PO results for a large class of scoring functions, which includes as particular cases all the *weighted power means*. For computing ND, we provide two alternative approaches, one in which $\mathcal{F}$-dominance is tested via Linear Programming, the other based on the $\mathcal{F}$-dominance region of a tuple. The evaluation of PO can also be carried through with two alternative methods, both based on Linear Programming. In Section 5, we introduce several orthogonal opportunities for efficiently implementing algorithms computing ND and PO, giving rise to more than 10 algorithmic alternatives. In Section 6, we assess all algorithmic variants on a number of synthetic and real datasets, and discuss how efficiency varies with respect to several parameters, including the size and dimensionality of the datasets, the number of constraints, and the family of scoring functions in use. Additionally, we evaluate the effectiveness of F-skylines in reducing the cardinality of the result with respect to classical skylines, and compare the obtained results with those of both skylines and ranking queries. Our experimental findings show that F-skylines admit efficient implementations in most practical scenarios and provide an effective way to reduce the cardinality of the result set. Section 7 discusses related work and Section 8 concludes. The four appendices report, respectively, a detailed analysis of the behavior of weighted power means, an extension of our results to cover all cases in which the set of scoring functions fails to distinguish all possible tuples, additional experiments, and all the proofs of the claims included in the main body of the paper.

Summarizing, the main contributions of this paper are as follows.

(1) We introduce two operators, called F-skylines, incorporating user preferences into the skyline framework.

(2) We detail the basic formal properties of F-skylines for arbitrary families of monotone scoring functions.

(3) We study the application of F-skylines when the scoring functions in $\mathcal{F}$ are linear in the weights (and arbitrary monotone transforms are applied to attribute values).

(4) We discuss two alternative approaches to checking $\mathcal{F}$-dominance (thus to computing ND).

(5) For computing potentially optimal tuples (PO), we propose two methods, both based on Linear Programming.

(6) We introduce more than 10 algorithmic alternatives for efficiently computing ND and PO, all of which are experimentally analyzed in a number of different experimental settings including synthetic as well as real datasets.

(7) We evaluate the effectiveness of F-skylines (i.e., their ability to restrict the set of tuples of interest) by considering their relationship with both skylines and ranking queries.

2 PRELIMINARIES

Consider a relational schema $R^+(A_1, \ldots, A_d, B_1, \ldots, B_z)$, with $d \geq 1$ and $z \geq 0$, where the first d attributes are numeric attributes relevant for the analysis to follow, while the remaining ones are supplemental attributes that will be henceforth disregarded. We denote the projection of R^+ on $A_1, \ldots, A_d$ as R. 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.[1] In the following, we refer to an instance r over R.

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

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

An equivalent definition 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 2.2 (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:

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

The (infinite) set of all monotone scoring functions is denoted by $\mathtt{MF}$.

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 [10] that, for every tuple t in the skyline, there exists a monotone scoring function such that t minimizes that scoring function. Therefore, as formally shown in [10], the skyline of r can be equivalently specified as:

$$\textsc{Sky}(r) = \{t \in r \mid \exists f \in \mathtt{MF}.\ \forall s \in r.\ s \neq t \rightarrow f(t) < f(s)\}. \tag{3}$$

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 strictly better than all the others according to at least one monotone scoring function (Equation (3)). As we shall see in Section 3, although these two views coincide here, their underlying concepts are different.

[1] In order to simplify the presentation, we assume that the projection of R^+ over $A_1, \ldots, A_d$ does not generate duplicates. The extension to the general case of our results is straightforward.

Example 2.3. Let $r = \{t_1, t_2, t_3\}$, where $t_1 = \langle 1, 2\rangle$, $t_2 = \langle 1, 3\rangle$, $t_3 = \langle 4, 1\rangle$. According to Definition 2.1, the skyline of r is $\textsc{Sky}(r) = \{t_1, t_3\}$, since t_2 is dominated by t_1, while neither t_1 nor t_3 are dominated. As for the equivalent specification given in (3), t_1 is in the skyline because there exists a monotone function $f_1(x, y) = x + y$ such that $f_1(t_1) < f_1(t_2)$ and $f_1(t_1) < f_1(t_3)$. Similarly, t_3 is the optimal point in r according to function $f_2(x, y) = y$, and thus t_3 is in the skyline. The reason why in (3) a strict inequality is required (i.e., $f(t) < f(s)$) is evident if one looks at tuple t_2, for which there is no monotone function that makes it the *only* optimal point in r. Indeed, even though, for $f_3(x, y) = x$, t_2 reaches the best value obtainable for tuples in r, $f_3(t_2)$ is *not* strictly smaller than $f_3(t_1)$, and thus t_2 is not a skyline point.

3 MAIN DEFINITIONS AND PROBLEM STATEMENT

We now adopt the two different views of skylines to introduce two corresponding operators, called *flexible skyline* operators, whose behavior is the same as Sky, but applied to a limited set of monotone scoring functions $\mathcal{F} \subseteq \mathtt{MF}$. In the following, we always assume $\mathcal{F}$ to be non-empty, and, for simplicity, although the main focus of our results and algorithms to follow is on the infinite case, we will sometimes offer examples in which $\mathcal{F}$ is a finite set. In order to simplify the presentation, we initially consider that the set of monotone scoring functions $\mathcal{F}$ satisfies the natural property of being *tuple-distinguishing*, as defined below (the case of sets of functions that are not tuple-distinguishing is analyzed in Section 4.3 and in Appendix B).

Definition 3.1 (Tuple-distinguishing set). A set $\mathcal{F}$ of scoring functions is said to be *tuple-distinguishing* if the following holds:

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

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 one function in $\mathcal{F}$ associating two different scores with any two distinct tuples. Note that the assumption of tuple-distinguishability entails that, for each attribute A_i, there exists at least one function in $\mathcal{F}$ that depends on A_i.[2]

Example 3.2. Assume $d = 2$ and consider the monotone scoring function $f_1(x, y) = x + y$ and the set $\mathcal{F}_1 = \{f_1\}$. Set $\mathcal{F}_1$ is not tuple-distinguishing. Indeed, $f_1(0, 1) = f_1(1, 0) = 1$, and therefore tuples $\langle 0, 1\rangle$ and $\langle 1, 0\rangle$ cannot be distinguished by any function in $\mathcal{F}_1$.

Consider now $d = 3$ and the set of functions $\mathcal{F}_2 = \{f_2, f_3\}$, with $f_2(x, y, z) = x + 2y$ and $f_3(x, y, z) = x + 3y$. Note that none of the functions in $\mathcal{F}_2$ depends on the third attribute, and, indeed, set $\mathcal{F}_2$ is not tuple-distinguishing, as it fails to distinguish between $\langle 0, 0, 0\rangle$ and $\langle 0, 0, 1\rangle$, since $f_2(0, 0, 0) = f_2(0, 0, 1) = f_3(0, 0, 0) = f_3(0, 0, 1) = 0$.

We now extend the notion of dominance introduced in Definition 2.1 so as to take into account the set of scoring functions under consideration.

Definition 3.3 ($\mathcal{F}$-Dominance). Let $\mathcal{F}$ be a set of monotone scoring functions. A tuple t *$\mathcal{F}$-dominates* another tuple $s \neq t$, denoted by $t \prec_{\mathcal{F}} s$, if $\forall f \in \mathcal{F}.\ f(t) \leq f(s)$.

An immediate consequence of Definition 3.3, stated in the following corollary, is that $\mathcal{F}$-dominance between tuples cannot be lost by reducing the set of scoring functions.

Corollary 3.4. *For any sets $\mathcal{F}$ and $\mathcal{F}'$ of monotone scoring functions, the following holds:*

$$\text{if } \mathcal{F} \subset \mathcal{F}' \text{ then } \prec_{\mathcal{F}'} \subseteq \prec_{\mathcal{F}}\ . \tag{5}$$

In particular, $\prec \subseteq \prec_{\mathcal{F}}$, i.e., dominance entails $\mathcal{F}$-dominance.

[2] When some attributes are omitted in all the functions in $\mathcal{F}$, we fall into the same case considered for *subspace skylines* [37], which we discuss in Section 7.

However, when considering a larger set of functions, $\mathcal{F}$-dominance between tuples may not be preserved, as shown in Example 3.5 below.

Example 3.5. Assume $d = 2$ and 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 in Definition 3.3 holds. However, $t \not\prec_{\mathtt{MF}} s$, since MF includes, among others, $f_3(x, y) = 2x + y$, for which $f_3(t) = 1.5 > f_3(s) = 1$, thereby violating the condition in Definition 3.3.

Another consequence of Definition 3.3 is the that $\mathcal{F}$-dominance is transitive.

Corollary 3.6. *For any set $\mathcal{F}$ of monotone scoring functions and any tuples t, s, u, the following holds:*

$$\text{if } t \prec_{\mathcal{F}} s \text{ and } s \prec_{\mathcal{F}} u \text{ then } t \prec_{\mathcal{F}} u. \tag{6}$$

With Definition 3.3 at hand, we are now ready to introduce the first flexible skyline operator, called *non-dominated flexible skyline*, which consists of the set of non-$\mathcal{F}$-dominated tuples in r, as specified in Definition 3.7 below.

Definition 3.7 (ND). Let $\mathcal{F} \subseteq \mathtt{MF}$ be a set of monotone scoring functions. The *non-dominated flexible skyline* of r with respect to $\mathcal{F}$, denoted by $\text{ND}(r; \mathcal{F})$, is defined as the following set of tuples:

$$\text{ND}(r; \mathcal{F}) = \{t \in r \mid \nexists s \in r.\ s \prec_{\mathcal{F}} t\}. \tag{7}$$

Note that the right-hand side of Equation (7) is similar to that of Equation (1), where $\prec$ has been replaced by $\prec_{\mathcal{F}}$. Observe that, clearly, $\prec_{\mathtt{MF}}$ coincides with $\prec$.

The second flexible skyline operator, called *potentially optimal flexible skyline*, returns the tuples that are best (i.e., top-1) according to some scoring function in $\mathcal{F}$, as specified in Definition 3.8 below.

Definition 3.8 (PO). Let $\mathcal{F} \subseteq \mathtt{MF}$ be a set of monotone scoring functions. The *potentially optimal flexible skyline* of r with respect to $\mathcal{F}$, denoted by $\text{PO}(r; \mathcal{F})$, is defined as:

$$\text{PO}(r; \mathcal{F}) = \{t \in r \mid \exists f \in \mathcal{F}.\ \forall s \in r.\ s \neq t \rightarrow f(t) < f(s)\}. \tag{8}$$

Again, similarly to ND, the only difference between the right-hand side of Equation (8) and that of Equation (3) is that MF has been replaced by $\mathcal{F}$.

Example 3.9. Let $r = \{t_1, t_2, t_3\}$, where $t_1 = \langle 1, 2\rangle$, $t_2 = \langle 1, 3\rangle$, $t_3 = \langle 4, 1\rangle$, as in Example 2.3, and consider the set $\mathcal{F} = \{f_1, f_2\}$, where $f_1(x, y) = x$ and $f_2(x, y) = y$. Here, $\text{ND}(r; \mathcal{F}) = \{t_1, t_3\}$, since t_2 is $\mathcal{F}$-dominated by t_1. Tuple t_3 is also potentially optimal with respect to $\mathcal{F}$, since $f_2(t_3) < f_2(t_1)$ and $f_2(t_3) < f_2(t_2)$. Tuples t_1 and t_2, instead, are never the single best alternative for any of the functions in $\mathcal{F}$, since $f_1(t_1) = f_1(t_2)$, and thus $\text{PO}(r; \mathcal{F}) = \{t_3\}$.

Although the above example might suggest that F-skylines are somehow similar to dynamic skylines [35], this only holds when we consider *finite* sets of functions.[3] The next example illustrates F-skylines with an infinite set of scoring functions.

Example 3.10. Consider the tuples of Example 1.1, shown in Figure 1, and the set $\mathcal{F} = \{w_P\texttt{Price} + w_M\texttt{Mileage} \mid w_P \geq w_M\}$. Car C6 is $\mathcal{F}$-dominated by car C4, since, for every function f in $\mathcal{F}$ we have $f(C4) < f(C6)$. Indeed, $f(C4) = w_P \cdot 20 + w_M \cdot 15 < w_P \cdot 35 + w_M \cdot 10 = f(C6)$ reduces to $3w_P > w_M$, which is clearly entailed by $w_P \geq w_M$. With similar arguments, we conclude that C1$\prec_{\mathcal{F}}$C3, C2$\prec_{\mathcal{F}}$C5, and C4$\prec_{\mathcal{F}}$C7, and, thus, that $\text{ND}(\texttt{UsedCars}; \mathcal{F})$ = {C1,C2,C4}. Clearly,

[3] In Section 7, we discuss dynamic skylines more in detail.

since C6, C3, C5, and C7 are never better than their $\mathcal{F}$-dominating cars, such cars are not potentially optimal with respect to $\mathcal{F}$. Car C1 is the best when, say, $w_P = 1$ and $w_M = 0$, whereas C4 is the best when, say, $w_P = w_M$, and thus both C1 and C4 are potentially optimal. Car C2, although not $\mathcal{F}$-dominated by any other car in UsedCars, is never better than both C1 and C4. Indeed, $f(\text{C2}) = w_P \cdot 18 + w_M \cdot 25 < f(\text{C1}) = w_P \cdot 10 + w_M \cdot 35$ reduces to $4w_P < 5w_M$, while $f(\text{C2}) < f(\text{C4}) = w_P \cdot 20 + w_M \cdot 15$ reduces to $w_P > 5w_M$. Clearly, $4w_P < 5w_M$ and $w_P > 5w_M$ cannot be both satisfied at the same time. Therefore, $\text{PO}(\texttt{UsedCars}; \mathcal{F}) = \{\text{C1,C4}\}$.

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* $\text{ND}(r; \mathcal{F})$ *and* $\text{PO}(r; \mathcal{F})$ *for any given instance* r *and set of monotone scoring functions* $\mathcal{F}$.

Table 2. Table of notation.

Notation	Meaning
$t[A_i]$	Value of tuple t on attribute A_i
MF	Set of all monotone scoring functions
r	A generic instance of cardinality N over a schema $A_1, \ldots, A_d$
$\text{Sky}(r)$	Skyline of r
$\mathcal{F}$	A generic set of monotone scoring functions
$t \prec_{\mathcal{F}} s$	Tuple t $\mathcal{F}$-dominates tuple s
$\text{ND}(r; \mathcal{F})$	Non-dominated flexible skyline of r with respect to $\mathcal{F}$
$\text{PO}(r; \mathcal{F})$	Potentially optimal flexible skyline of r with respect to $\mathcal{F}$
$\mathcal{W}$	Set of all normalized weight vectors
C	A set of c (linear) constraints on weights
$\mathcal{W}(C)$	Set of normalized weight vectors satisfying the set of constraints C
$DR(t; \mathcal{F})$	$\mathcal{F}$-dominance region of a tuple t under a set of 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. For convenience, a summary of the main notation adopted in this paper is offered in Table 2.

We start by observing that, as a direct consequence of the definitions, in the limit case in which the set $\mathcal{F}$ of scoring functions coincides with MF, both PO and ND coincide with the skyline:

$$\text{PO}(r; \texttt{MF}) = \text{ND}(r; \texttt{MF}) = \text{Sky}(r). \tag{9}$$

In the more general case, which is our main focus, it can be proved that ND is a subset of the skyline and PO is a subset of ND, as the following Proposition 3.11 states.

Proposition 3.11. *For any set* $\mathcal{F}$ *of monotone scoring functions, the following relationships hold:*

$$\text{PO}(r; \mathcal{F}) \subseteq \text{ND}(r; \mathcal{F}) \subseteq \text{Sky}(r). \tag{10}$$

A major concept that we exploit in order to check $\mathcal{F}$-dominance is that of the *$\mathcal{F}$-dominance region* of a tuple t.

Definition 3.12. 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:

$$DR(t;\mathcal{F}) = \{s \in [0,1]^d \mid t \prec_{\mathcal{F}} s\}. \tag{11}$$

The following Example provides some intuition on Proposition 3.11 and Definition 3.12.

Example 3.13. Let $\mathcal{F}$ be the set of all the linear scoring functions of the form $f(x,y) = w_1x + w_2y$ such that $w_1 \geq w_2$. 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 2a. We have

$$\textsc{po}(r;\mathcal{F}) = \{t_1, t_3\} \subset \textsc{nd}(r;\mathcal{F}) = \{t_1, t_2, t_3\} \subset \textsc{Sky}(r) = r.$$

To see this, first observe that no tuple in r dominates any other tuple in r, and therefore $\textsc{Sky}(r) = r$. 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 $w_1 0.5 + w_2 0.2 \leq w_1 0.6 + w_2 0.15$, that is, $w_1(0.5 - 0.6) \leq w_2(0.15 - 0.2)$, which is always true in $\mathcal{F}$, since $w_1 \geq w_2$. Therefore $t_4 \notin \textsc{nd}(r;\mathcal{F})$. To further emphasize this, Figure 2a 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., $\cup_{t\in r} DR(t;\mathcal{F})$.

To see that $t_2 \notin \textsc{po}(r;\mathcal{F})$, note that there is no function $f \in \mathcal{F}$ for which both $f(t_2) < f(t_1)$ and $f(t_2) < f(t_3)$ hold, as there are no non-negative w_1, w_2 satisfying the system of inequalities

$$\{ \quad w_1(0.4 - 0.3) < w_2(0.6 - 0.45), \quad w_1(0.4 - 0.5) < w_2(0.2 - 0.45) \quad \}.$$

The case on which we focus is when the set $\mathcal{F}$ is defined by starting with an infinite family of scoring functions and then some constraints on the weight parameters of such functions are added. 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 C be a set of (linear) constraints on weights, and denote with $\mathcal{W}(C)$ the subset of $\mathcal{W}$ that satisfies C, i.e., $\mathcal{W}(C) = \{W \in \mathcal{W} \mid C(W) = \texttt{true}\}$.[4] Given two sets of constraints C_1 and C_2 applied to the same family $\mathcal{F}$, and leading to the sets $\mathcal{F}_1$ and $\mathcal{F}_2$, respectively, we say that C_1 is *more restrictive* than C_2 if $\mathcal{W}(C_1) \subset \mathcal{W}(C_2)$, thus $\mathcal{F}_1 \subseteq \mathcal{F}_2$.

The following Proposition 3.14, which has general validity, shows that, the more the constraints are restrictive, the more the results of both ND and PO reduce.

Proposition 3.14. ND *and* PO *are monotone operators with respect to the set of scoring functions, i.e., for any two sets* $\mathcal{F}_1$ *and* $\mathcal{F}_2$ *of monotone scoring functions such that* $\mathcal{F}_1 \subseteq \mathcal{F}_2$*, the following relationships hold:*

$$\textsc{po}(r;\mathcal{F}_1) \subseteq \textsc{po}(r;\mathcal{F}_2), \tag{12}$$

$$\textsc{nd}(r;\mathcal{F}_1) \subseteq \textsc{nd}(r;\mathcal{F}_2). \tag{13}$$

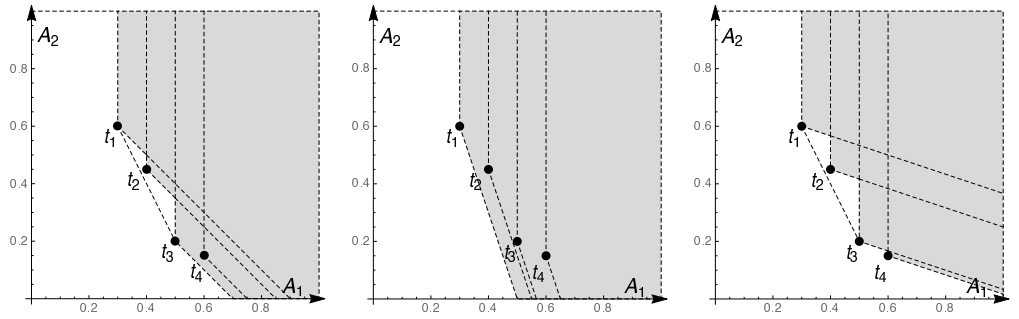
An immediate consequence of Definition 3.12 and Corollary 3.4 is that the $\mathcal{F}$-dominance region grows larger for smaller sets of functions.

Corollary 3.15. *For any tuple* t *over* R *and any two sets* $\mathcal{F}_1$ *and* $\mathcal{F}_2$ *of monotone scoring functions such that* $\mathcal{F}_1 \subseteq \mathcal{F}_2$*, the following holds:*

$$DR(t;\mathcal{F}_1) \supseteq DR(t;\mathcal{F}_2). \tag{14}$$

Example 3.13 (cont.). Let $\mathcal{F}$ be the set of all the linear scoring functions of the form $f(x,y) = w_1x + w_2y$ such that $C = \{w_1 \geq 3w_2\}$ holds. The $\mathcal{F}$-dominance regions of t_1, t_2, t_3, t_4 are shown in

[4] Henceforth, we always assume that C is not contradictory, i.e., $\mathcal{W}(C) \neq \emptyset$ and, consequently, the so-derived $\mathcal{F}$ is non-empty.



(a) $C = \{w_1 \geq w_2\}$. (b) $C = \{w_1 \geq 3w_2\}$. (c) $C = \{3w_1 \geq w_2\}$.

Fig. 2. Tuples from Example 3.13 in $[0,1]^2$. $\mathcal{F}$-dominance regions in gray, where $\mathcal{F}$ is the set of monotone scoring functions of the form $f(t) = w_1 t[A_1] + w_2 t[A_2]$ subject to a set of constraints C on the weights.

gray in Figure 2b. Note that this set of constraints is more restrictive than $w_1 \geq w_2$, and thus each $\mathcal{F}$-dominance region contains the corresponding region shown in Figure 2a. We have

$$\text{PO}(r;\mathcal{F}) = \{t_1\} = \text{ND}(r;\mathcal{F}).$$

Assume now that $\mathcal{F}$ is of the same form but is, instead, subject to $C = \{3w_1 \geq w_2\}$. This time, the constraints are less restrictive than $w_1 \geq w_2$, and thus each $\mathcal{F}$-dominance region (shown in Figure 2c) is contained in the corresponding region shown in Figure 2a. Here we have

$$\text{PO}(r;\mathcal{F}) = \{t_1, t_3, t_4\} \subset \{t_1, t_2, t_3, t_4\} = \text{ND}(r;\mathcal{F})$$

■

3.2 Further properties

We now present further properties of our F-skyline operators, which extend their applicability beyond the core methods and algorithms illustrated in the next sections.

One may wonder whether having ND (resp., PO) unaffected by changes in the set of scoring functions would also leave PO (resp., ND) unaltered. This is not the case, as the following example shows.

Example 3.16. Consider the instance $r = \{t_1, t_2, t_3\}$, with $t_1 = \langle 0.3, 0.7\rangle$, $t_2 = \langle 0.65, 0.3\rangle$, and $t_3 = \langle 0.4, 0.4\rangle$, and the monotone scoring functions $f_1(x,y) = x$, $f_2(x,y) = y$, and $f_3(x,y) = x + y$. Let $\mathcal{F}_1 = \{f_1, f_2\}$ and $\mathcal{F}_2 = \{f_1, f_2, f_3\}$. We have $\text{ND}(r;\mathcal{F}_1) = \text{ND}(r;\mathcal{F}_2) = r$, but $\text{PO}(r;\mathcal{F}_1) = \{t_1, t_2\} \subset \text{PO}(r;\mathcal{F}_2) = r$, i.e., PO changes even if ND remains unchanged.

Consider now also $t_4 = \langle 0.6, 0.6\rangle$, the instance $r' = \{t_1, t_2, t_4\}$, and let $\mathcal{F}_1{}' = \{f_1, f_3\}$ and $\mathcal{F}_2{}' = \{f_1, f_2, f_3\}$. We have $\text{PO}(r';\mathcal{F}_1{}') = \text{PO}(r';\mathcal{F}_2{}') = \{t_1, t_2\}$, but $\text{ND}(r';\mathcal{F}_1{}') = \{t_1, t_2\} \subset \text{ND}(r';\mathcal{F}_2{}') = r'$, i.e., ND changes even if PO remains unchanged.

The next proposition highlights the different behavior of ND and PO when unions of sets of scoring functions are considered.

PROPOSITION 3.17. *Let $\mathcal{F} = \mathcal{F}_1 \cup \ldots \cup \mathcal{F}_m$, where each $\mathcal{F}_i$, $1 \leq i \leq m$, is a set of monotone scoring functions. Then:*

$$\text{ND}(r;\mathcal{F}) \supseteq \text{ND}(r;\mathcal{F}_1) \cup \ldots \cup \text{ND}(r;\mathcal{F}_m) \tag{15}$$

$$\text{PO}(r;\mathcal{F}) = \text{PO}(r;\mathcal{F}_1) \cup \ldots \cup \text{PO}(r;\mathcal{F}_m) \tag{16}$$

Note that there are cases in which the inclusion in (15) holds strictly, as demonstrated in Example 3.19 below. However, no tuple in the left-hand side but not in the right-hand side of (15) can be potentially optimal, as stated in the following corollary, which is a direct consequence of Propositions 3.11 and 3.17.

Corollary 3.18. *Let $\mathcal{F} = \mathcal{F}_1 \cup \ldots \cup \mathcal{F}_m$, where each $\mathcal{F}_i$, $1 \leq i \leq m$, is a set of monotone scoring functions. If $t \in \text{ND}(r;\mathcal{F}) \setminus \cup_{i=1}^{m}\text{ND}(r;\mathcal{F}_i)$ then $t \notin \text{PO}(r;\mathcal{F})$.*

Example 3.19. Consider the instance $r = \{t_1, t_2, t_3\}$, with $t_1 = \langle 0.3, 0.7\rangle$, $t_2 = \langle 0.7, 0.3\rangle$, and $t_3 = \langle 0.4, 0.4\rangle$. Let $\mathcal{F}_1 = \{f_{11}, f_{12}\}$, $\mathcal{F}_2 = \{f_{21}, f_{22}\}$, and $\mathcal{F} = \mathcal{F}_1 \cup \mathcal{F}_2$, with $f_{11}(x, y) = x$, $f_{12}(x, y) = x + 0.1y$ and $f_{21}(x, y) = y$, $f_{22}(x, y) = 0.1x + y$. We have $\text{ND}(r;\mathcal{F}_1) = \{t_1\}$, $\text{ND}(r;\mathcal{F}_2) = \{t_2\}$, yet $\text{ND}(r;\mathcal{F}) = \{t_1, t_2, t_3\} \supset \{t_1, t_2\} = \text{ND}(r;\mathcal{F}_1) \cup \text{ND}(r;\mathcal{F}_2)$. Note that $\text{PO}(r;\mathcal{F}_1) = \{t_1\}$, $\text{PO}(r;\mathcal{F}_2) = \{t_2\}$, and $\text{PO}(r;\mathcal{F}) = \{t_1, t_2\} \not\ni t_3$.

For the intersection of families of scoring functions ($\mathcal{F} = \mathcal{F}_1 \cap \ldots \cap \mathcal{F}_m$), the following relationships directly follow from Proposition 3.14:

$$\text{ND}(r;\mathcal{F}) \subseteq \text{ND}(r;\mathcal{F}_1) \cap \ldots \cap \text{ND}(r;\mathcal{F}_m) \tag{17}$$

$$\text{PO}(r;\mathcal{F}) \subseteq \text{PO}(r;\mathcal{F}_1) \cap \ldots \cap \text{PO}(r;\mathcal{F}_m) \tag{18}$$

As the following example shows, there are cases in which both inclusions are strict.

Example 3.20. Let $\mathcal{F}_1 = \{f_1, f_3, f_4\}$, $\mathcal{F}_2 = \{f_2, f_3, f_4\}$ and $\mathcal{F} = \mathcal{F}_1 \cap \mathcal{F}_2 = \{f_3, f_4\}$, with $f_1(x, y) = 0.1x+y$, $f_2(x, y) = 0.01x+y$, $f_3(x, y) = x+y$, $f_4(x, y) = x+0.1y$. Consider a relation $r = \{t_1, t_2\}$, with $t_1 = \langle 0.8, 0.1\rangle$ and $t_2 = \langle 0.1, 0.2\rangle$. We have $\text{ND}(r;\mathcal{F}_1) = \text{PO}(r;\mathcal{F}_1) = \text{ND}(r;\mathcal{F}_2) = \text{PO}(r;\mathcal{F}_2) = \{t_1, t_2\}$, but $\text{ND}(r;\mathcal{F}) = \text{PO}(r;\mathcal{F}) = \{t_2\}$.

We have so far considered how the behavior of our operators changes as the set of scoring functions changes. We now state an anti-monotonic property of ND and PO regarding their behavior with respect to changes in the input relation.

Proposition 3.21. *Let r and r' be two instances over R with $r \subset r'$. For any set of monotone scoring functions $\mathcal{F}$, the following holds:*

- $\text{ND}(r';\mathcal{F}) \cap r \subseteq \text{ND}(r;\mathcal{F})$,
- $\text{PO}(r';\mathcal{F}) \cap r \subseteq \text{PO}(r;\mathcal{F})$.

The above proposition guarantees that, if a tuple $t \in \text{ND}(r';\mathcal{F})$ (respectively, $t \in \text{PO}(r';\mathcal{F})$) is also a tuple in r, then t also belongs to $\text{ND}(r;\mathcal{F})$ (respectively, $\text{PO}(r;\mathcal{F})$). Thus, if one shrinks a dataset r' by retaining only a subset r of its tuples, then all tuples in ND (resp., PO) that are still present will continue to be in ND (resp., PO).

When the input data consists of several, possibly distributed relations (sharing the same schema), i.e., $r = r_1 \cup \ldots \cup r_m$, the computation of $\text{ND}(r;\mathcal{F})$ and $\text{PO}(r, \mathcal{F})$ can rely on the following result, which mimics strategies commonly adopted for skylines [10].

Proposition 3.22. *Let $r = r_1 \cup \ldots \cup r_m$, where r_i, $1 \leq i \leq m$, are relations over R, and $\mathcal{F}$ be a set of monotone scoring functions. Then:*

$$\text{ND}(r;\mathcal{F}) = \text{ND}(\text{ND}(r_1;\mathcal{F}) \cup \ldots \cup \text{ND}(r_m;\mathcal{F});\mathcal{F}), \tag{19}$$

$$\text{PO}(r;\mathcal{F}) = \text{PO}(\text{PO}(r_1;\mathcal{F}) \cup \ldots \cup \text{PO}(r_m;\mathcal{F});\mathcal{F}). \tag{20}$$

4 CHECKING $\mathcal{F}$-DOMINANCE

When $\mathcal{F}$ is a finite set, determining ND and PO can be easily done by case enumeration. However, most interesting cases occur when $\mathcal{F}$ is an infinite family of scoring functions, possibly restricted

by a set of constraints on weight parameters. Since devising a method for checking $\mathcal{F}$-dominance in the general case may be challenging as well as computationally prohibitive, in this section we discuss efficient strategies for computing both ND and PO when each function in $\mathcal{F}$ is what we call a *monotonically-transformed, linear-in-the-weights* (*MLW*) function, i.e., a function with the following form

$$f^W(t) = h\left(\sum_{i=1}^{d} w_i g_i(t[A_i])\right), \tag{21}$$

where $W = (w_1, \ldots, w_d) \in \mathcal{W}(C)$, C is a set of linear constraints, and all g_i's and h are continuous, monotone transforms such that the g_i's and h are all either *i)* non-decreasing, or *ii)* non-increasing. In the formal results we present, we distinguish the two cases by means of the Λ flag, where $\Lambda = 1$ in case *i*, $\Lambda = -1$ in case *ii*. Henceforth, we refer to the $g_i(t[A_i])$'s as the *marginal scores* of tuple t.

The class of MLW functions covers the majority of practically relevant cases when considering monotone scoring functions. For instance, they include commonly used (weighted) linear functions ($f^W(t) = \sum_{i=1}^{d} w_i t[A_i]$), but also, say, quadratic ($f^W(t) = \sqrt{\sum_{i=1}^{d} w_i t[A_i]^2}$) and harmonic ($f^W(t) = (\sum_{i=1}^{d} w_i t[A_i]^{-1})^{-1}$) means. Such a class is also very general, since the h and g_i's transforms account for arbitrary, monotone, non-linear extensions, e.g., $f^W(t) = \exp(\sum_{i=1}^{d} w_i \log^i(1 + t[A_i]))$.

Unless otherwise stated, $\mathcal{F}$ indicates a *homogeneous* family of MLW functions such that any two functions in $\mathcal{F}$ share the same set of transforms $h, g_1, \ldots, g_d$. At the end of Section 4.2, we discuss how non-homogeneous families can be dealt with in our framework.

4.1 Computing non-dominated tuples

In this section we introduce the main tools for determining the set of non-dominated tuples $\text{ND}(r; \mathcal{F})$, where $\mathcal{F}$ is a set of MLW functions.

Our first result shows that checking $\mathcal{F}$-dominance for MLW functions has polynomial complexity.

THEOREM 4.1 ($\mathcal{F}$-DOMINANCE TEST). *Let $\mathcal{F}$ be a set of MLW functions subject to a set $C = \{C_1, \ldots, C_c\}$ 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, $t \prec_{\mathcal{F}} s$ iff the following linear programming problem (LP) in the variables $W = (w_1, \ldots, w_d)$ has a non-negative solution:*

$$\begin{array}{ll} \text{minimize} & \Lambda \cdot \sum_{i=1}^{d} w_i (g_i(s[A_i]) - g_i(t[A_i])) \\ \text{subject to} & w_i \in [0,1] \qquad i \in \{1, \ldots, d\} \\ & \sum_{i=1}^{d} w_i = 1 \\ & \sum_{i=1}^{d} a_{ji} w_i \leq k_j \qquad j \in \{1, \ldots, c\}. \end{array} \tag{22}$$

Example 3.13 (cont.). In order to check whether $t_3 \prec_{\mathcal{F}} t_4$, according to Theorem 4.1 it suffices to check whether the following LP has a non-negative solution:

$$\begin{array}{ll} \text{minimize} & w_1(0.6 - 0.5) + w_2(0.15 - 0.2) \\ \text{subject to} & w_1, w_2 \in [0,1], w_1 + w_2 = 1, w_2 \leq w_1, \end{array}$$

which it does, e.g., by taking $w_1 = 1$ and $w_2 = 0$. ∎

Computing $\text{ND}(r; \mathcal{F})$ using Theorem 4.1 is likely to be time-consuming, since a different LP problem needs to be solved for each of the $O(N^2)$ $\mathcal{F}$-dominance tests implied by Definition 3.7, where N is the cardinality of the dataset. 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 (rather than once *per* tuple).

In order to compute $DR(t;\mathcal{F})$, a fundamental observation is that, for any set C of linear constraints on weights, $\mathcal{W}(C)$ is a convex polytope contained in the standard (or unit) $(d-1)$-simplex.[5] We have the following major result.

Theorem 4.2 ($\mathcal{F}$-dominance region). *Let $\mathcal{F}$ be a set of MLW functions subject to a set $C = \{C_1, \ldots, C_c\}$ 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 $\mathcal{W}(C)$. The dominance region $DR(t;\mathcal{F})$ of a tuple t under $\mathcal{F}$ is the locus of points s in $[0,1]^d$ defined by the q inequalities:*

$$\Lambda \cdot \sum_{i=1}^{d} w_i^{(\ell)} g_i(s[A_i]) \geq \Lambda \cdot \sum_{i=1}^{d} w_i^{(\ell)} g_i(t[A_i]), \quad \ell \in \{1, \ldots, q\}. \tag{23}$$

Example 3.13 (cont.). Since $C = \{w_1 \geq w_2\}$, and considering that $w_1 + w_2 = 1$ and $0 \leq w_2 \leq w_1 \leq 1$, the vertices of $\mathcal{W}(C)$ are $W^{(1)} = (1, 0)$ and $W^{(2)} = (\frac{1}{2}, \frac{1}{2})$. By Theorem 4.2, the dominance region $DR(t_3;\mathcal{F})$ of $t_3 = \langle 0.5, 0.2\rangle$ is characterized by the system of inequalities:

$$\{s[A_1] \geq 0.5, \quad \tfrac{1}{2}s[A_1] + \tfrac{1}{2}s[A_2] \geq \tfrac{1}{2}0.5 + \tfrac{1}{2}0.2\}. \tag{24}$$

Tuple $t_4 = \langle 0.6, 0.15\rangle$ satisfies (24) and thus $t_3 \prec_{\mathcal{F}} t_4$. For the dominance region $DR(t_1;\mathcal{F})$ of tuple $t_1 = \langle 0.3, 0.6\rangle$, the system becomes:

$$\{s[A_1] \geq 0.3, \quad \tfrac{1}{2}s[A_1] + \tfrac{1}{2}s[A_2] \geq \tfrac{1}{2}0.3 + \tfrac{1}{2}0.6\}. \tag{25}$$

Here, t_4 does not satisfy (25) and therefore $t_1 \not\prec_{\mathcal{F}} t_4$. ∎

As the above example and Figure 2a suggest, the "shape" of $DR(t;\mathcal{F})$ (modulo cropping in the $[0,1]^d$ hypercube) is independent of t. Indeed, in Inequalities (23), the left-hand sides of the inequalities stay the same and the right-hand sides are, for any given t, a constant.

Example 4.3. For a non-linear example, let $d = 3$ and consider the set $\mathcal{F}$ consisting of all MLW functions of the following form:

$$\sqrt{w_1 e^{t[A_1]} + w_2 \log(1 + t[A_2]) + w_3 t[A_3]^2}, \tag{26}$$

which complies with form (21) by posing $h(x) = \sqrt{x}$, $g_1(x) = e^x$, $g_2(x) = \log(1+x)$, $g_3(x) = x^2$, and subject to the set of constraints $C = \{w_1 + w_2 \geq w_3\}$. The vertices of $\mathcal{W}(C)$ are:

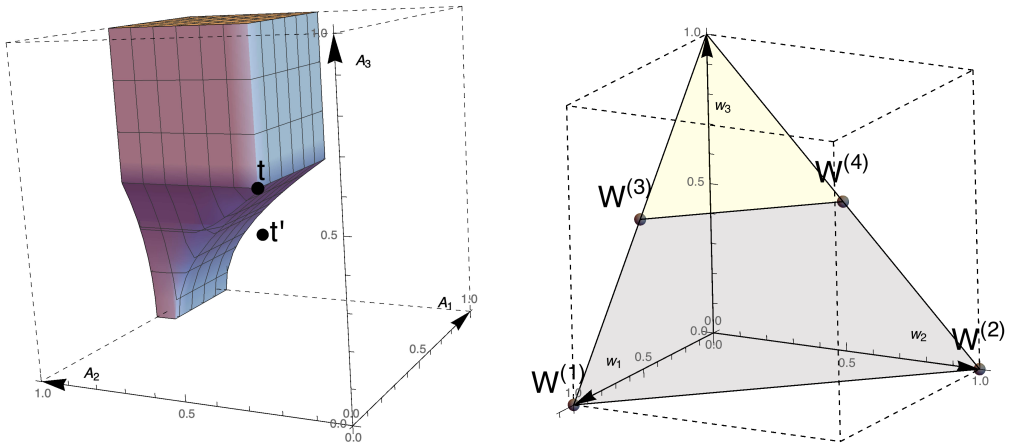
$$W^{(1)} = \langle 1, 0, 0\rangle, \quad W^{(2)} = \langle 0, 1, 0\rangle, \quad W^{(3)} = \langle \tfrac{1}{2}, 0, \tfrac{1}{2}\rangle, \quad W^{(4)} = \langle 0, \tfrac{1}{2}, \tfrac{1}{2}\rangle.$$

For $t = \langle \frac{1}{2}, \frac{1}{2}, \frac{1}{2}\rangle$, $DR(t;\mathcal{F})$ is characterized by:

$$\begin{aligned} \{ \quad & e^{s[A_1]} \geq e^{\frac{1}{2}}, \\ & \log(1 + s[A_2]) \geq \log(1 + \tfrac{1}{2}), \\ & \tfrac{1}{2}e^{s[A_1]} + \tfrac{1}{2}s[A_3]^2 \geq \tfrac{1}{2}e^{\frac{1}{2}} + \tfrac{1}{2}\tfrac{1}{2^2}, \\ & \tfrac{1}{2}\log(1 + s[A_2]) + \tfrac{1}{2}s[A_3]^2 \geq \tfrac{1}{2}\log(1 + \tfrac{1}{2}) + \tfrac{1}{2}\tfrac{1}{2^2} \quad \}. \end{aligned} \tag{27}$$

Therefore, tuple $t' = \langle 0.7, 0.5, 0.3\rangle$ is not $\mathcal{F}$-dominated by t, as the last inequality in (27) is not satisfied when $s = t'$. See Figure 3 for a graphical representation.

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

Fig. 3. Example 4.3 – tuples and weights in $[0,1]^d$, $d = 3$, $C = \{w_1 + w_2 \geq w_3\}$, $\mathcal{F}$ of form (26).

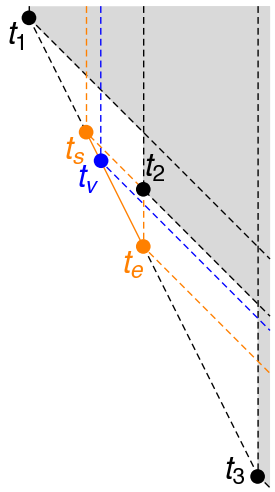


Fig. 4. Close-up of the tuples of Example 3.13. Tuple t_2 is not $\mathcal{F}$-dominated by either t_1 or t_3. However, t_2 is $\mathcal{F}$-dominated by the "virtual" tuple t_v, obtained as a convex combination of t_1 and t_3, so t_2 is not potentially optimal. Indeed, all convex combinations lying on the segment connecting t_s and t_e $\mathcal{F}$-dominate t_2.

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

4.2 Computing potentially optimal tuples

The fact that a tuple t is not $\mathcal{F}$-dominated is only a necessary condition for t to be potentially optimal, since there might anyhow be no function $f \in \mathcal{F}$ such that $f(t) < f(s)$ holds for all $s \neq t$.

An important property that suggests also a viable way to compute potentially optimal tuples is implied by the following proposition, which states that PO can be computed without considering

$\mathcal{F}$-dominated tuples. In particular, this allows computing PO in two phases, the first of which computes ND, as we shall discuss in detail in Section 5.

PROPOSITION 4.4. *Let $\mathcal{F}$ be a set of MLW functions subject to a set C of linear constraints on weights and such that h is strictly monotone. Then, for any instance r, we have* $\text{PO}(r;\mathcal{F}) = \text{PO}(\text{ND}(r;\mathcal{F});\mathcal{F})$.

The intuition behind the above result is that if $t \in \text{PO}(\text{ND}(r;\mathcal{F});\mathcal{F})$ then there is a function $f \in \mathcal{F}$ such that $f(t) < f(s)$, for all other tuples $s \in \text{ND}(r;\mathcal{F})$. In case for such f t ties with some tuple t' *not* in $\text{ND}(r;\mathcal{F})$, then, from the continuity of MLW functions, there exists another function f' "close" to f such that t remains the best tuple in $\text{ND}(r;\mathcal{F})$ and $f'(t) < f'(t')$.

From the very definition of potential optimality and Proposition 4.4, we have the following result, which holds provided that h is a strictly monotone function.

THEOREM 4.5 (PRIMAL PO TEST). *Let $\mathcal{F}$ be a set of MLW functions subject to a set $C = \{C_1, \ldots, C_c\}$ of linear constraints on weights, where $C_j = \sum_{i=1}^{d} a_{ji} w_i \leq k_j$ (for $j \in \{1, \ldots, c\}$), and such that h is strictly monotone. Let $\text{ND}(r;\mathcal{F}) = \{t_1, t_2, \ldots, t_\sigma, t\}$. Then, $t \in \text{PO}(r;\mathcal{F})$ iff the following linear programming problem (LP) in the variables $W = (w_1, \ldots, w_d)$ and ϕ has a strictly positive optimal solution:*

$$
\begin{array}{lll}
\text{maximize} & \phi & \\
\text{subject to} & \Lambda \cdot \sum_{i=1}^{d} w_i (g_i(t[A_i]) - g_i(t_j[A_i])) + \phi \leq 0 & j \in \{1, \ldots, \sigma\} \\
& \sum_{i=1}^{d} a_{ji} w_i \leq k_j & j \in \{1, \ldots, c\} \\
& w_i \in [0, 1] & i \in \{1, \ldots, d\} \\
& \sum_{i=1}^{d} w_i = 1. &
\end{array}
\tag{28}
$$

Note that, when the above problem has an optimal solution $\phi^* > 0$, then it is guaranteed that there exists a weight vector $W^* = (w_1^*, \ldots, w_d^*)$ such that $\sum_{i=1}^{d} w_i^* g_i(t[A_i]) < \sum_{i=1}^{d} w_i^* g_i(t_j[A_i])$ holds for all tuples t_j in $\text{ND}(r;\mathcal{F}) \setminus \{t\}$. Since h is strictly monotone, this proves that t is optimal for the function with weights W^*.

An alternative approach to the technique induced by Theorem 4.5 is based on the notion of *convex combination* of (the marginal scores of) a set of tuples.

Definition 4.6 (Convex combination). Given tuples $t_1, \ldots, t_n$, $n > 1$, and monotone transforms g_i, $1 \leq i \leq d$, a tuple s is a *convex combination* (through the g_i's) of (the marginal scores of) $t_1, \ldots, t_n$ if there exist $\alpha_1, \ldots, \alpha_n$ such that $\alpha_j \in [0, 1]$ for $1 \leq j \leq n$, $\sum_{j=1}^{n} \alpha_j = 1$, and

$$
g_i(s[A_i]) = \sum_{j=1}^{n} \alpha_j g_i(t_j[A_i]), \quad i \in \{1, \ldots, d\}.
$$

Indeed, as also shown in Example 3.13 for tuple t_2, there might be a "virtual" tuple obtained through a convex combination of other tuples in $\text{ND}(r;\mathcal{F})$ that $\mathcal{F}$-dominates (or coincides with) t.

Figure 4 offers a close-up of the scenario described in Example 3.13 and shows that, among the convex combinations of t_1 and t_3 (lying on the segment connecting t_1 and t_3) there are tuples, such as t_v (in blue), that $\mathcal{F}$-dominate t_2. Indeed, all the convex combinations of t_1 and t_3 lying on the segment connecting t_s and t_e (shown as a solid orange line) $\mathcal{F}$-dominate t_2. Note that, although in Example 3.13 the convex combinations of tuples lie on a segment, this is not the case when the monotone transforms g_i are not linear (see Appendix A).

We now make precise the intuition suggested by Figure 4 that the existence of a convex combination that $\mathcal{F}$-dominates a tuple t is a necessary and sufficient condition for t not being in $\text{PO}(r;\mathcal{F})$.

Theorem 4.7 (Dual po test). *Let $\mathcal{F}$ be a set of MLW functions subject to a set C of linear constraints on weights and such that h is strictly monotone. Let $W^{(1)}, \ldots, W^{(q)}$ be the vertices of $\mathcal{W}(C)$ and let $\text{ND}(r;\mathcal{F}) = \{t_1, t_2, \ldots, t_\sigma, t\}$. Then, $t \in \text{PO}(r;\mathcal{F})$ iff there is no convex combination s of $t_1, \ldots, t_\sigma$ such that $s \prec_{\mathcal{F}} t$, i.e., iff the following linear system in the variables $\alpha = (\alpha_1, \ldots, \alpha_\sigma)$ is unsatisfiable:*

$$\begin{gathered}\Lambda \cdot \sum_{i=1}^{d} w_i^{(\ell)} \left(\sum_{j=1}^{\sigma} \alpha_j g_i(t_j[A_i])\right) \leq \Lambda \cdot \sum_{i=1}^{d} w_i^{(\ell)} g_i(t[A_i]) \quad \ell \in \{1, \ldots, q\} \\ \alpha_j \in [0,1] \quad j \in \{1, \ldots, \sigma\} \\ \sum_{j=1}^{\sigma} \alpha_j = 1.\end{gathered} \tag{29}$$

On the left-hand side of Inequalities (29), the sum $\sum_{j=1}^{\sigma} \alpha_j g_i(t_j[A_i])$ occurs, which, according to Definition 4.6, corresponds to the i-th marginal score $g_i(s[A_i])$ of a convex combination s. Thus, Problem (29) is satisfiable if and only if such a convex combination s exists, such that s $\mathcal{F}$-dominates t. In Appendix D, we rigorously prove this based on the relationship between the *primal* formulation of a linear program (as shown in Theorem 4.5) and its *dual* [22]. The formulation of Theorem 4.7 is a modified version of the dual of Problem (28), in which we also exploit properties of the vertices of the polytope (not present in the primal).

When $d = 2$, the $\mathcal{F}$-dominance condition $s \prec_{\mathcal{F}} t$ required by Theorem 4.7 can be replaced by a dominance condition $s \prec t$: when starting from a set of mutually non-$\mathcal{F}$-dominating tuples, if a convex combination of two of them (say, t and u) $\mathcal{F}$-dominates another tuple v, then there must also be a (possibly different) convex combination of t and u that dominates v.

Proposition 4.8 (2d po test). *Let $\mathcal{F}$ be a set of MLW functions subject to a set C of linear constraints on weights and such that the g_i's and h are strictly monotone. Let t, u, and v be three tuples in $[0,1]^2$ such that i) $\text{ND}(\{t,u,v\};\mathcal{F}) = \{t,u,v\}$, and ii) there exists a convex combination of t and u that $\mathcal{F}$-dominates v. Then there exists a convex combination of t and u that dominates v.*

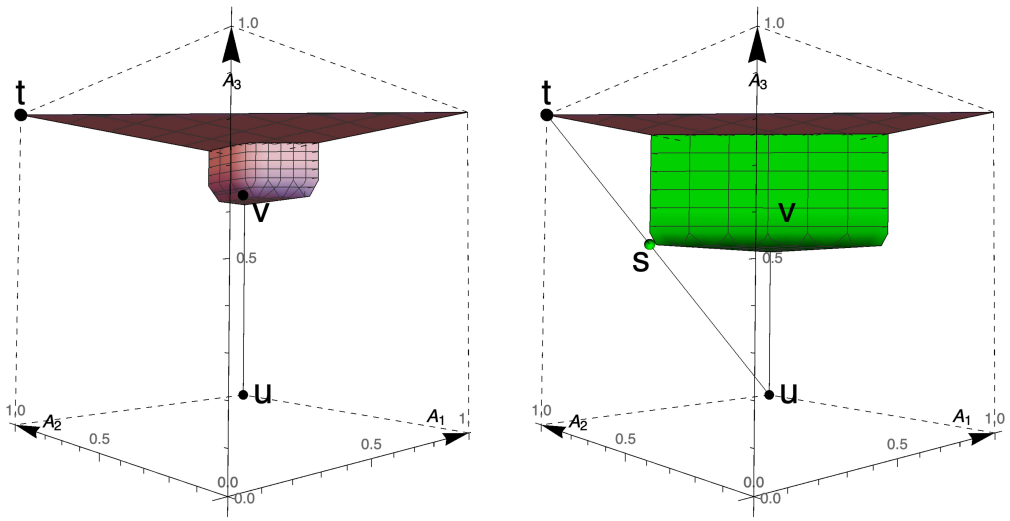
The result of Proposition 4.8 is, however, only of theoretical interest, since, testing dominance instead of $\mathcal{F}$-dominance amounts to changing the first q inequalities of System (29) with d inequalities; yet, when $d = 2$, the number of vertices q is 2, thus with no concrete reduction in the size of the problem. Proposition 4.8 does not apply to higher dimensions, as the following Example shows.

Example 4.9. Consider the tuples $t = \langle 0,1,1\rangle$, $u = \langle 1,1,0\rangle$, $v = \langle 0.8, 0.8, 0.8\rangle$, the set $\mathcal{F}$ of functions of the form $\sum_{i=1}^{3} w_i t[A_i]$ subject to the constraint $C = \{w_1 \geq w_2\}$. As Figure 5a shows, tuple u does not $\mathcal{F}$-dominate any point, while t only $\mathcal{F}$-dominates a triangular region at the top of the unit cube (shown in gray). Therefore, v (whose $\mathcal{F}$-dominance region is shown in pink) is not $\mathcal{F}$-dominated by either t or u, and, indeed, $\text{ND}(\{t,u,v\};\mathcal{F}) = \{t,u,v\}$. However, v is $\mathcal{F}$-dominated by some convex combination of t and u, e.g., $s = \langle 0.4, 1, 0.6\rangle$. Figure 5b shows that the $\mathcal{F}$-dominance region of s (shown in green) encloses v, and thus $v \notin \text{PO}(\{t,u,v\};\mathcal{F})$. Note that no convex combination s' of t and u can dominate v, since $s'[A_2] = 1$, while $v[A_2] = 0.8$, which shows that Proposition 4.8 only holds when $d = 2$.

All the results derived in this section can be immediately extended to the case in which $\mathcal{F}$ is a *finite* union of homogeneous sets of MLW functions, i.e., $\mathcal{F} = \mathcal{F}_1 \cup \ldots \cup \mathcal{F}_m$.

Proposition 3.17 guarantees that $\text{PO}(r;\mathcal{F})$ can be computed by taking the union of all the $\text{PO}(r;\mathcal{F}_i)$ sets, each of which can be obtained by either the method in Theorem 4.5 or 4.7.

Instead, $\text{ND}(r;\mathcal{F})$ *cannot* be computed by taking the union of the $\text{ND}(r;\mathcal{F}_i)$ sets. Rather, when comparing two tuples s and t, either using Theorem 4.1 or 4.2, one should apply the corresponding method for all the $\mathcal{F}_i$ sets and then conclude that $t \prec_{\mathcal{F}} s$ iff $t \prec_{\mathcal{F}_i} s$ for $1 \leq i \leq m$.



(a) $\mathcal{F}$-dominance regions for t (top gray triangle), u (empty), and v (in pink).

(b) $\mathcal{F}$-dominance region (in green) of a convex combination of t and u $\mathcal{F}$-dominating v.

Fig. 5. Example 4.9 – tuples and weights in $[0,1]^d$, $d = 3$, $C = \{w_1 \geq w_2\}$, $\mathcal{F}$ are weighted sums.

4.3 Non-tuple-distinguishing sets of MLW functions

The notion of $\mathcal{F}$-dominance was given in Definition 3.3 for tuple-distinguishing sets. When considering non-tuple-distinguishing (NTD) sets, an extra condition is needed (point *(ii)* in Definition 4.10 below, which implicitly holds when $\mathcal{F}$ is tuple-distinguishing).

Definition 4.10. Let $\mathcal{F}$ be a NTD set of monotone scoring functions. A tuple t *$\mathcal{F}$-dominates* another tuple $s \neq t$, denoted by $t \prec_{\mathcal{F}} s$, if *(i)* $\forall f \in \mathcal{F}.\ f(t) \leq f(s)$ and *(ii)* $\exists f \in \mathcal{F}.\ f(t) < f(s)$.

The notion of tuple-distinguishability for a set of MLW functions is tightly connected with the existence of d linearly independent weight vectors in $\mathcal{W}(C)$, as the following result shows.

Proposition 4.11. *Let $\mathcal{F}$ be a set of homogeneous MLW functions, such that the g_i's and h are strictly monotone, and subject to a set C of linear constraints on weights. Then, $\mathcal{F}$ is tuple-distinguishing iff d linearly independent weight vectors exist in $\mathcal{W}(C)$.*

Based on the above result, we can provide an effective way for checking whether a set $\mathcal{F}$ of MLW functions is tuple-distinguishing.

Theorem 4.12 (Tuple-distinguishability of MLW functions). *Let $\mathcal{F}$ be a set of homogeneous MLW functions such that the g_i's and h are strictly monotone, and subject to a set $C = \{C_1, \ldots, C_c\}$ 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, $\mathcal{F}$ is tuple-distinguishing iff there exists a weight vector $W^* = (w_1^*, \ldots, w_d^*) \in \mathcal{W}(C)$ such that $\sum_{i=1}^{d} a_{ji} w_i^* < k_j$ holds for $j \in \{1, \ldots, c\}$.*

According to Theorem 4.12, verifying tuple-distinguishability of a set of MLW functions amounts to checking feasibility of a linear program with constraints $\sum_{i=1}^{d} a_{ji} w_i < k_j, j \in \{1, \ldots, c\}$.

Example 4.13. Consider $d = 3$ and a set of MLW functions $\mathcal{F} = \{f^W(t) = \sum_{i=1}^{d} w_i t[A_i] \mid w_3 = w_1 + w_2\}$ (i.e., with the notation of Theorem 4.12, $C = \{-w_1 - w_2 + w_3 \leq 0, w_1 + w_2 - w_3 \leq 0\}$).

Table 3. Algorithmic variants for computing ND.

	ULP2	UVE2	SLP2	SVE2	SVE1/ SVE1F
sorting			✓	✓	✓
vertex enumeration		✓		✓	✓
1 phase (non SKY-based)					✓

Table 4. Algorithmic variants for computing PO.

	POPF2	PODF2	PODI1	POPI2	PODI2
dual LP test		✓	✓		✓
incremental			✓	✓	✓
ND-based	✓	✓		✓	✓

According to Theorem 4.12, $\mathcal{F}$ is NTD, because the system $\{-w_1 - w_2 + w_3 < 0, w_1 + w_2 - w_3 < 0\}$ is unsatisfiable. Indeed, consider $t_1 = \langle 0.5, 0.5, 0\rangle$ and $t_2 = \langle 0, 0, 0.5\rangle$. We have $f^W(t_1) = 0.5w_1 + 0.5w_2 = 0.5(w_1 + w_2) = 0.5w_3$ and $f^W(t_2) = 0.5w_3$, and thus t_1 and t_2 are undistinguishable by $\mathcal{F}$.

We now show how the Theorems in Section 4 have to be modified in order to correctly deal with NTD sets of functions.

Theorem 4.1 also holds when $\mathcal{F}$ is NTD if the solution to (22) is strictly positive.[6] If the solution to (22) is 0, an extra check is needed: $t \prec_{\mathcal{F}} s$ iff Problem (22) with s and t swapped provides a strictly negative solution. Indeed, if in both cases the solution is 0, then $\sum_{i=1}^{d} w_i g_i(s[A_i]) = \sum_{i=1}^{d} w_i g_i(t[A_i])$ for all $(w_1, \ldots, w_d) \in \mathcal{W}(C)$, thus s and t are indistinguishable.

Similarly, when $\mathcal{F}$ is NTD, Theorem 4.2 still applies, with the extra requirement that at least one of the Inequalities (23) is satisfied strictly. As a direct consequence of Definition 3.12, $DR(t;\mathcal{F})$ is a closed region if and only if $\mathcal{F}$ is tuple-distinguishing, since $\mathcal{F}$-dominance in this case requires only non-strict inequalities.

Theorem 4.5, for which $t \in \text{PO}(r;\mathcal{F})$ is determined using the very definition of potential optimality, requires no changes at all when $\mathcal{F}$ is NTD. The same holds also for Theorem 4.7.

5 ALGORITHMS

Based on different options for computing ND and PO, we consider several algorithmic alternatives, which are summarized in Tables 3 and 4.

5.1 Computing ND

Since ND is a subset of the skyline, it is in principle conceivable to first compute the skyline and then obtain the result from it. Alternatively, we can directly compute ND from the input dataset. Clearly, in the first case *any* skyline algorithm can be adopted. Among the many available alternatives, in this paper we consider the two well-known BNL [6] and SFS [11] algorithms, the latter performing a preliminary topological sort of the input data. Note that the alternative of whether to sort or not the input data is also practicable if a single-phase approach is pursued (i.e., without first computing the skyline). The available alternatives are further extended by considering how $\mathcal{F}$-dominance can be tested. The options we consider are described as follows.

[6] Here, as well as in the generalization of Theorem 4.2, we are obviously excluding the degenerate case in which the h transform is a constant function.

Algorithm 1: SLP2 and SVE2 for ND.

Input: *relation* r, *constraints* C, *family* $\mathcal{F}$ *of MLW functions.* Output: $\text{ND}(r;\mathcal{F})$.

(1) Prepare
(2) $\mathcal{S} := \text{SKY}(r)$ // *phase one*
(3) **for each** s **in** $\mathcal{S}$ // *candidate* $\mathcal{F}$*-dominated tuple*
(4) **if** SVE2 **then** compute left-hand sides of Inequalities (23)
(5) **for each** t **in** ND // *candidate* $\mathcal{F}$*-dominant tuple*
(6) **if** $t \prec_{\mathcal{F}} s$ **then continue** to line 3
(7) ND := ND $\cup \{s\}$
(8) **return** ND

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

Algorithm 2: SVE1 for ND.

Input: *relation* r, *constraints* C, *family* $\mathcal{F}$ *of MLW functions.* 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 (23)
(6) **for each** t **in** ND // *candidate* $\mathcal{F}$*-dominant tuple*
(7) **if** $t \prec_{\mathcal{F}} s$ **then continue** to line 2
(8) ND := ND $\cup \{s\}$
(9) **return** ND

Algorithm 3: SVE1F for ND.

Input: *relation* r, *constraints* C, *family* $\mathcal{F}$ *of MLW functions.* 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 (23)
(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) ND := ND $\cup \{s\}$
(7) **return** ND

Phases. First, we can 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 ND should be directly computed from the input dataset, without materializing the skyline (i.e., in one phase, "1" in the algorithm's name). As an optimization that exploits the fact that dominance entails $\mathcal{F}$-dominance (by Corollary 3.4), we still perform dominance tests, since they provide a faster way to safely

discard uninteresting tuples, while the more costly $\mathcal{F}$-dominance tests should be performed only if dominance does not hold.

Sorting. The second option regards whether to sort the dataset beforehand so as to produce a topological sort with respect to the $\mathcal{F}$-dominance relation, i.e., if tuple t precedes tuples s in the sorted input relation, then $s \not\prec_{\mathcal{F}} t$. In order to obtain such a topological sort, we can use as sorting function any weighted sum in which the weights satisfy the constraints C. In particular, we adopt as weights the coordinates of the centroid of the polytope $\mathcal{W}(C)$. 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. The alternatives for testing whether $s \prec_{\mathcal{F}} t$ are: *i)* solving an LP problem as in Theorem 4.1 (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 4.2 through vertex enumeration of the polytope $\mathcal{W}(C)$ ("VE" in the name).

So far, there are 8 alternatives. We start by describing the four 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 Sky(r); 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 $\mathcal{W}(C)$ 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 4.1 for SLP2 and via Theorem 4.2 for SVE2. In the latter case, it is useful to precompute the left-hand sides of Inequality (23) already at line 4.

The unsorted counterparts ULP2 and UVE2 are similar, but, without sorting, *i)* we cannot compute Sky(r) via SFS, and thus use the classical BNL algorithm, 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 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, are the harder $\mathcal{F}$-dominance tests (lines 6–7) 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 with respect to 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

Even for computing PO several alternatives are available. We can compute ND as an intermediate result or directly obtain the potentially optimal tuples from the input data, and the test of optimality can rely on two alternatives as well. Since optimality tests can become costly (especially for large problem sizes), we introduce a heuristic optimization technique that tries to discard non-optimal tuples by using an incremental, approximate strategy.

Phases. As in the algorithms for computing $\text{ND}(r;\mathcal{F})$, we can choose the number of "phases". A 2-phase algorithm (indicated by "2" in the algorithm's name) first computes $\text{ND}(r;\mathcal{F})$ and then filters out non-$\text{PO}(r;\mathcal{F})$ tuples from it, as Proposition 4.4 allows; a 1-phase algorithm ("1" in the name) discards non-$\text{PO}(r;\mathcal{F})$ tuples starting from the original relation r. In order to avoid any confusion with the terminology used for ND-variants, in the following we shall refer to 2-phase (resp., 1-phase) algorithms for computing PO as ND-*based* (resp., *non*-ND-*based*) algorithms.

PO test. Another choice regards the use of the primal (Theorem 4.5) or of the dual (Theorem 4.7) PO test. In the following, the algorithm's name will correspondingly contain the letter "P" (primal) or "D" (dual).

Incrementality. Testing whether a tuple t is potentially optimal with respect to σ other tuples requires solving an LP problem involving the marginal scores of all the $\sigma + 1$ tuples in the set, as indicated in the claims of Theorems 4.5 and 4.7. However, this task may be prohibitively time consuming when σ is large. Yet, we can solve an LP problem using only a subset of the σ other tuples: if the solution indicates that t is not potentially optimal, we can safely discard t. This allows us to try to solve LP problems of increasing sizes, with smaller sizes first, so that testing the full problem with all the other σ tuples will only be needed for those tuples that could not be discarded through smaller problem instances. The algorithm's name will contain the letter "I" (incremental) if this approach is adopted, or "F" (full) if the full-size instance is directly tested.

The eight resulting variants[7] can be divided into non-incremental (POPF1, PODF1, POPF2, PODF2) and incremental (POPI1, PODI1, POPI2, PODI2) algorithms.

Algorithm 4 describes the non-incremental approaches. The initial set of tuples we consider (PO) is prepared in the subprocedure `InitializePO` (line 1), where we either start with the tuples in r (for non-ND-based variants, line 5) or in $\text{ND}(r;\mathcal{F})$ (for an ND-based algorithm, line 6). All the algorithms we consider sort the starting set (be it r or $\text{ND}(r;\mathcal{F})$) in the same way as the algorithms for computing ND. Thanks to this, we can then sortedly enumerate candidate non-PO tuples from PO in reverse order (line 2), as the worst tuples with respect to the ordering are the most likely to be discarded. We then test whether t is non-potentially-optimal in PO through the `isNonPO` function: if so, we remove t from PO (line 3). If the adopted test uses the primal LP problem of Theorem 4.5 (line 7), then System (28) is solved with the tuples in PO; if the system does not have a positive solution, then $t \notin \text{PO}(r;\mathcal{F})$. Similarly, if the dual LP problem of Theorem 4.7 is used (line 8), then System (29) is solved in order to look for a convex combination of tuples in $\text{PO} \setminus \{t\}$ that $\mathcal{F}$-dominates t; if such a tuple is found, i.e., the system is satisfiable, then $t \notin \text{PO}(r;\mathcal{F})$. After all tuples are scanned, we return the remaining tuples (line 4).

Algorithm 5 describes the incremental variants. In order to reduce as early as possible the set of candidate potentially optimal tuples (PO), we adopt 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 LP problems; 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 8); *ii)* we test whether t is non-potentially-optimal with respect to the *first* $\tilde{\sigma}$ tuples in PO (lines 6 and 7), as they are the best with respect to the ordering and thus more likely to beat other tuples. After this early pruning, in the last round (enabled by line 4) all

[7] We prepend the letters "PO" to the algorithm's name.

Algorithm 4: PO: non-incremental computation.

Input: *relation r, constraints C, family $\mathcal{F}$ of MLW functions.* Output: $\text{PO}(r;\mathcal{F})$.

(1) `InitializePO`
(2) **for each** t **in** `PO` in reverse order *// candidate non-*PO *tuple*
(3) ⋮ **if** `isNonPO`$(t, \texttt{PO} \setminus \{t\})$ **then** `PO` := `PO` $\setminus \{t\}$
(4) **return** `PO`

Subprocedure: `InitializePO`

(5) **if** `phases` =1 **then** `Prepare`; `PO` := r // `Prepare` *as in Algorithm 1*
(6) **else** `PO` := $\text{ND}(r;\mathcal{F})$ // *this includes* `Prepare` *as in Algorithm 1*

Function: `isNonPO`. Input: *candidate tuple t, set of tuples T*. Output: **true** *iff* $t \notin \text{PO}(T \cup \{t\};\mathcal{F})$.

(7) `primal`: **return true** iff (28) on $T \cup \{t\}$ does not have a solution > 0
(8) `dual`: **return true** iff $\exists s.\ s \prec_{\mathcal{F}} t$, where s is a convex combination of the tuples in T

Algorithm 5: PO: incremental computation.

Input: *relation r, constraints C, family $\mathcal{F}$ of MLW functions.* Output: $\text{PO}(r;\mathcal{F})$.

(1) `InitializePO`
(2) $\tilde{\sigma}$:= 2; `lastRound` := **false**
(3) **while**(¬`lastRound`)
(4) ⋮ **if** $\tilde{\sigma} \geq |\texttt{PO}| - 1$ **then** `lastRound` := **true**
(5) ⋮ **for each** t **in** `PO` in reverse order *// candidate non-*PO *tuple*
(6) ⋮ ⋮ T := first $\min(\tilde{\sigma}, |\texttt{PO}| - 1)$ tuples in `PO` $\setminus \{t\}$
(7) ⋮ ⋮ **if** `isNonPO`(t, T) **then** `PO` := `PO` $\setminus \{t\}$
(8) ⋮ $\tilde{\sigma}$:= $\tilde{\sigma} \cdot 2$
(9) **return** `PO`

Table 5. Time complexity of algorithms for computing ND.

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

the remaining tuples are checked against all the other tuples still in PO, which is now a necessary and sufficient condition for pruning, as in Theorem 4.7.

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 with respect to 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 $O(\text{ve}(c))$ and $O(\text{lp}(x,y))$, 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

Table 6. Time complexity of algorithms for computing PO.

algorithm	complexity
POPF2	$C_{\text{ND}} + O(\lvert\text{ND}\rvert \cdot \text{lp}(\lvert\text{ND}\rvert + c, d))$
PODF2	$C_{\text{ND}} + O(\lvert\text{ND}\rvert \cdot \text{lp}(q, \lvert\text{ND}\rvert))$
PODI1	$O(N \cdot \log N \cdot \text{lp}(q, N))$
POPI2	$C_{\text{ND}} + O(\lvert\text{ND}\rvert \cdot \log \lvert\text{ND}\rvert \cdot \text{lp}(\lvert\text{ND}\rvert + c, d))$
PODI2	$C_{\text{ND}} + O(\lvert\text{ND}\rvert \cdot \log \lvert\text{ND}\rvert \cdot \text{lp}(q, \lvert\text{ND}\rvert))$

in general and it is not known whether for the special case of bounded polytopes (like $\mathcal{W}(C)$) 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 $O(c^{\lfloor d/2 \rfloor})$ (see [26] and references therein).

Tables 5 summarizes our results for ND. For the 2-phase algorithms, the complexity of the first phase is that of the corresponding skyline algorithm [24]. In the second phase, ULP2 performs at most $O(\lvert\text{SKY}\rvert^2)$ $\mathcal{F}$-dominance tests, each of which costs $O(\text{lp}(c, d))$.[8] On the other hand, SVE2 first enumerates the vertices of $\mathcal{W}(C)$, which costs $O(\text{ve}(c))$, and then performs at most $\lvert\text{SKY}\rvert \cdot \lvert\text{ND}\rvert$ $\mathcal{F}$-dominance tests using Theorem 4.2, each of which costs $O(q)$, where q is the number of vertices of $\mathcal{W}(C)$. The worst-case complexity of SVE1 and SVE1F is the same, since, besides enumerating vertices of $\mathcal{W}(C)$ and sorting the dataset, both execute $O(N \cdot \lvert\text{ND}\rvert)$ $\mathcal{F}$-dominance tests. From the comparison between SVE2 and SVE1 (and SVE1F), we argue that the larger the skyline, the more SVE2 will be penalized.

For all variants that compute $\text{PO}(r; \mathcal{F})$ starting from $\text{ND}(r; \mathcal{F})$, we include the complexity of computing ND and indicate it as C_{ND} in the results summarized in Table 6. The PODI2 algorithm will execute the loop at most $O(\log \lvert\text{ND}\rvert)$ times, which happens in the very unlikely case in which most of the tuples in ND \ PO appear before those in PO in the ordering; at each iteration, PODI2 will execute at most $\lvert\text{ND}\rvert$ $\mathcal{F}$-dominance tests, the cost of which is bounded by $O(\text{lp}(q, \lvert\text{ND}\rvert))$, from which the result follows. Analogously, POPI2 incurs a cost of $O(\text{lp}(\lvert\text{ND}\rvert + c, d))$ for checking $\mathcal{F}$-dominance with the primal test, hence the result. The non-incremental variants PODF2 and POPF2 save a factor $O(\log \lvert\text{ND}\rvert)$ from the corresponding incremental versions (in the worst case), since there is no outer loop varying $\tilde{\sigma}$. The complexity of PODI1 is the same as that of PODI2, in which ND is replaced by N, without the component due to the computation of ND.

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. The largest LP problem encountered by 2-phase variants for computing PO will be a matrix of size $O(q \times \lvert\text{ND}\rvert)$. This grows to $O(q \times N)$ for 1-phase variants.

6 EXPERIMENTS

In this section we aim to assess the efficiency of the various algorithmic alternatives for computing F-skylines, and to understand how F-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. As for the family of scoring functions in use, we

[8] When considering the number of constraints in an LP problem, we disregard those imposing non-negativity of variables, as these are commonly implicitly assumed by LP solvers.

consider weighted power means M_p^W, defined as follows:

$$M_p^W(t) = \left(\sum_{i=1}^{d} w_i t[A_i]^p \right)^{1/p}, \quad p \neq 0 \tag{30}$$

$$M_0^W(t) = \prod_{i=1}^{d} t[A_i]^{w_i}. \tag{31}$$

Note that, when $p < 0$, $M_p^W(t)$ is defined only when each $t[A_i] > 0$, for $1 \leq i \leq d$. Furthermore, when $p = 0$, we obtain the *geometric mean*, which can be expressed as $M_0^W(t) = \exp(\sum_{i=1}^{d} w_i \log t[A_i])$ (which indeed shows that even M_0^W is a MLW function), provided that $t[A_i] > 0$, for $1 \leq i \leq d$. We shall therefore restrict to tuples from $(0, 1]^d$ when using weighted power means with $p \leq 0$. More details on weighted power means are given in Appendix A.

The relevant parameters are shown in Table 7, with defaults in bold.

In this section, we focus on the main distinctive traits of our algorithms, and therefore omit the evaluation of experimental settings and datasets that turn out to be less challenging. We discuss all such cases in detail in Appendix C.

Table 7. Operating parameters for performance evaluation (defaults, when available, are in bold).

Full name	Tested values
Distribution	synthetic: ANT, UNI, COR; real: NBA, HOU, GAU
Synthetic dataset size (N)	10K, 50K, **100K**, 500K, 1M, 5M, 10M
# of dimensions (d)	2, 4, **6**, 8, 10
# of constraints (c)	1, 2, 3, 4, 5 (default: $\mathbf{d/2}$)
Parameter of M_p^W mean (p)	[-5,5] with step 0.5 (default: **1**)

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 [6]. For any value of d and N mentioned in Table 7, we produced three d-dimensional datasets of size N with values in the $[0, 1]$ interval: one of these datasets (UNI) has values distributed uniformly in $[0, 1]$; another one (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; the last dataset (COR) has correlated values. Like other less challenging datasets, we shall only report detailed results about COR in Appendix C.

The real datasets analyzed here are three publicly available datasets, two of which are commonly 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.[9] 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 6 offers a representation of the notions of F-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

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

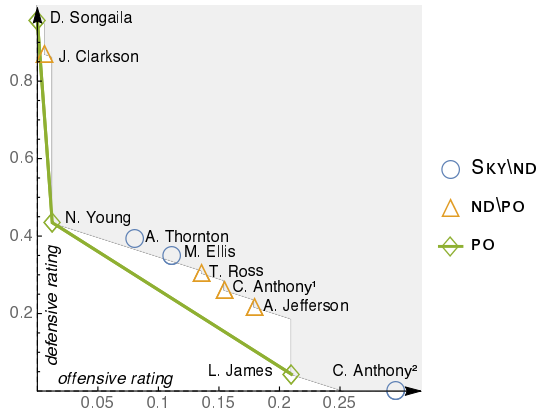


Fig. 6. NBA dataset, $d = 2$, $w_1 \geq w_2$: SKY, ND and PO.

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 $-45°$.

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.

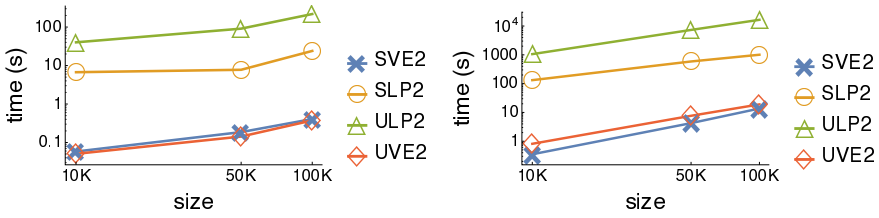
The third dataset (EMP) contains information regarding salary and benefits paid to City employees since fiscal year 2013 in San Francisco[10] and consists of 291, 825 multi-dimensional points including 6 numeric attributes describing several forms of compensation (salary, overtime, other salaries, retirement, dental health, and other benefits). The EMP dataset is an example of naturally clustered data. In the interest of space, in this section, both HOU and EMP are only mentioned in a table reporting aggregated results (Table 8), while more detailed results regarding these datasets are found in Appendix C.

For our experiments, we consider one of the most common types of constraints on weights: *weak rankings* (see, e.g., [17] for an overview of useful constraints). In particular, for any number c of constraints mentioned in Table 7 (with $c < d$), we consider the following set: $\{w_i \geq w_{i+1} | i \in \{1, \ldots, c\}\}$. Note that, due to Theorem 4.12, the set of functions obtained by applying such constraints to a family of weighted M_p^W means is tuple-distinguishing for any value of p. In the following, we only show results for constraints in the form of weak rankings, whereas in Appendix C we shall also consider *interval constraints* of the form $\bar{w}_i(1 - \varepsilon) \leq w_i \leq \bar{w}_i(1 + \varepsilon)$, for $1 \leq i \leq d$, indicating uncertainty in the weight around a value $\bar{w}_i$. Results for interval constraints show indeed trends similar to weak rankings, as concisely summarized in Section 6.3. Moreover, we study the effect of varying p on weighted M_p^W means.

6.2 Results on efficiency

We assess efficiency of the different algorithms for computing ND and PO 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

[10] Available at https://data.world/data-society/employee-compensation-in-sf.



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

Fig. 7. Performance of 2-phase algorithms for ND on smaller-scale datasets as the size N varies.

computing PO, we only report the execution time. The data structures storing partial results were implemented as plain arrays, with no particular indexing structure built on the fly.

All LP problems were solved by integrating our system with the linear programming tool lp_solve.[11] In the worst-case scenario ($d = 10$, $c = 5$), solving the LP problem in Theorem 4.1 required $0.24ms$ on average. In the same scenario, it took $3.7ms$ on average to check the PO test of Theorem 4.7, and $4.1ms$ for the PO test of Theorem 4.5.

Vertex enumeration, relevant for Theorems 4.2 and 4.7, was performed with the `lrs` tool,[12] which never required more than $18ms$ to complete. We remind that vertex enumeration needs to be done only once during the entire computation of ND and PO.

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 7 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 4.2, 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.

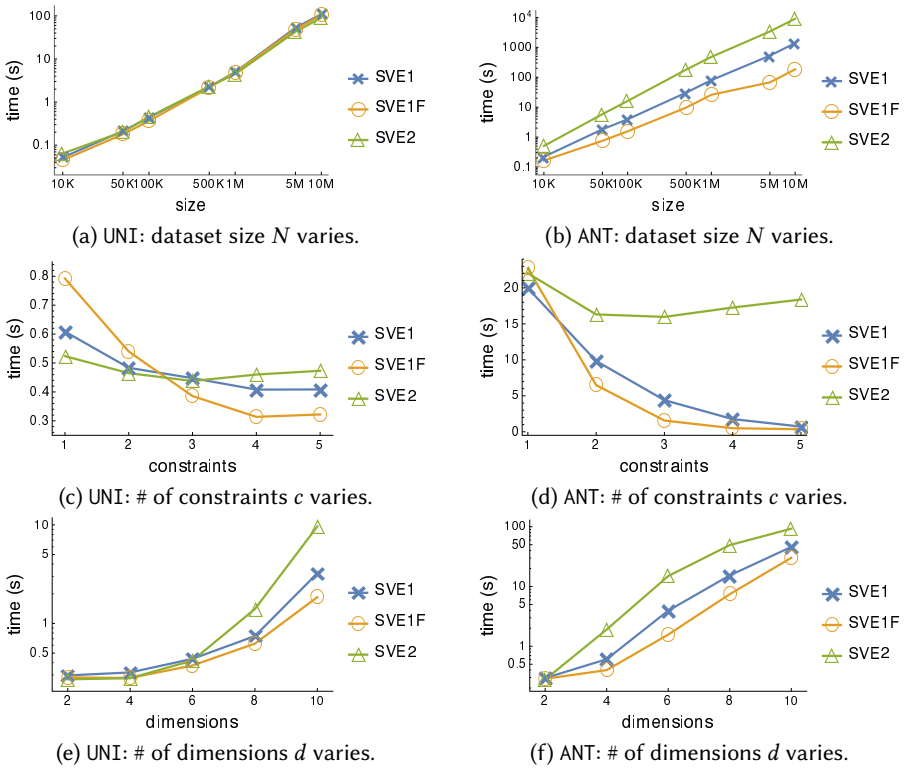
Although comparable for less challenging datasets (UNI, Figure 7a), SVE2 prevails over UVE2 in the more difficult cases (ANT, Figure 7b). This is in line with what others observed when comparing sorted (i.e., SFS) vs. unsorted (i.e., BNL) skyline algorithms, which are used in the first phase of SVE2 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 (Algorithms 4 and 5), in the following we 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 10M tuples (see Figures 8a and 8b). 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 9a and 9b for the ANT dataset. We observe that SVE1F performs at least 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 8c and 8d we vary the number of constraints from $c = 1$ to $c = 5$. As the figures show, the

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

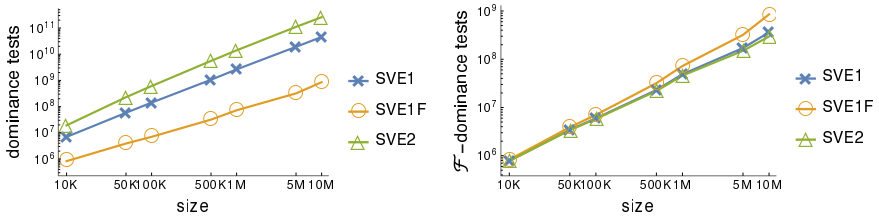
[12] http://cgm.cs.mcgill.ca/~avis/C/lrs.html.

.



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

Fig. 8. Performance for computing ND. Distribution: UNI on the left; ANT on the right.

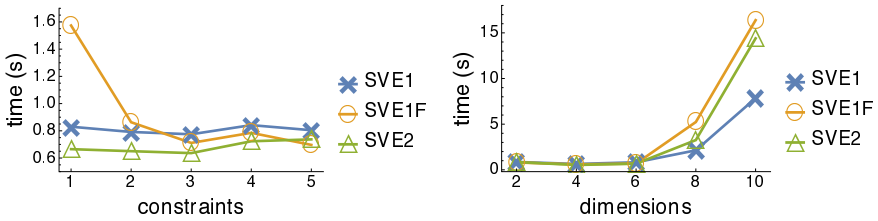


(a) ANT: dataset size N varies.
(b) ANT: dataset size N varies.

Fig. 9. Dominance (a) and $\mathcal{F}$-dominance (b) tests for computing ND on ANT.

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 8e and 8f. 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



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

Fig. 10. Performance for computing ND on the real dataset NBA.

(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 10a 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. As in the other datasets, 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; $\mathcal{F}$) with respect to SKY(NBA) is larger than for the synthetic datasets UNI and ANT (see Table 8), 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 10b, 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 $d \in \{8, 10\}$.

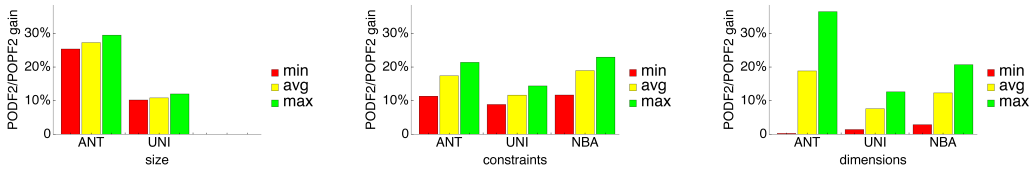
Table 8. Cardinalities of SKY, ND, PO with default values; in brackets, the ratio with the cardinality of SKY.

Dataset	SKY	ND	PO
ANT	26637	2616 (9.8%)	271 (1%)
UNI	2626	445 (16.9%)	122 (4.6%)
COR	31	13 (41.9%)	9 (29.0%)
NBA	1137	264 (23.2%)	32 (2.8%)
HOU	49	26 (53%)	19 (38.7%)
EMP	49	12 (24.5%)	5 (10.2%)

Computing PO.

We first assess ND-based algorithms for computing PO on all datasets, with default values on all but one parameter among size (N), constraints (c), and dimensions (d). For this set of experiments, we measure the time required to compute PO starting from the ND set.

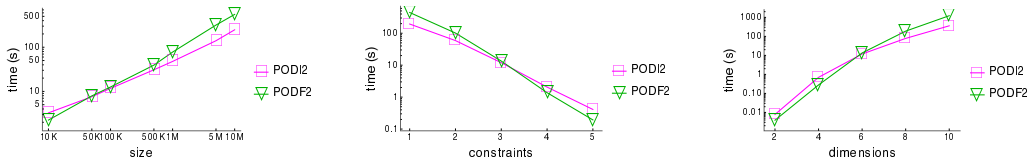
Figure 11 compares POPF2 with PODF2, with the latter always outperforming the former. The figure shows the minimum, average and maximum gain (in execution time) of PODF2 with respect to POPF2 in each dataset and for each varying parameter. Generally the gain is larger for more challenging instances, i.e., for higher values of N or d or lower values of c, and the effect is more visible for ANT than for UNI. For instance, with an ANT dataset of $N = 1M$ tuples, POPF2 finds the result in $105.1s$ while PODF2 requires $77.6s$, with a 29.5% gain. The widest range is found when d varies: with $d = 2$ the gain is unsubstantial (POPF2 and PODF2 are essentially tied at $0.0039s$ in ANT), whereas with $d = 10$ the gain is notable ($1923.1s$ for POPF2 vs. $1221.5s$ for PODF2, with a 36.5% gain). In all tested scenarios, the average gain is beyond 10%, with the sole exception of the UNI dataset with varying d, which turns out to be "easier" than the other datasets even for high values



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

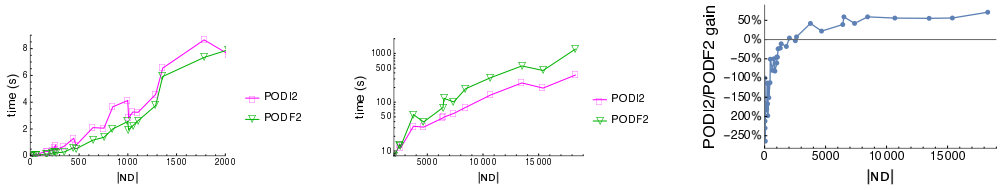
Fig. 11. Min, max and avg gains of PODF2 with respect to POPF2 while varying (a) size, (b) constraints, (c) dimensions.

of d, thus resulting in slightly lower gains (averaging 7.5%). Similar effects (not shown here for the sake of brevity) can be observed between POPI2 and PODI2, with PODI2 outperforming POPI2 in all tested scenarios. This shows that checking potential optimality through the dual PO test is generally faster than through the primal PO test. Indeed, this can be explained by observing that, unlike the primal PO test, the dual test just consists of a feasibility check, which can often fail early in the solution search performed by LP solvers. For this reason, we shall disregard methods based on the primal PO test in the following, and will now focus on the comparison between PODF2 and PODI2.



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

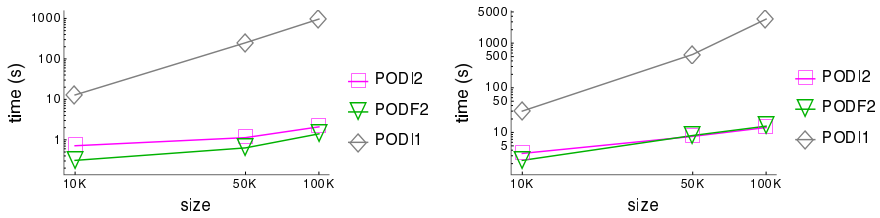
Fig. 12. Performance for computing PO on anti-correlated distribution.



(a) Smaller |ND| sizes: PODF2 prevails. (b) Larger |ND| sizes: PODI2 prevails. (c) PODI2/PODF2 gain as |ND| varies.

Fig. 13. Performance for computing PO as |ND| varies (on all distributions).

Figure 12 shows the time required by PODI2 and PODF2 to compute PO starting from the ND set on the ANT dataset. The incrementality of PODI2 starts to pay off as soon as the problem instance becomes sufficiently challenging. In our tests, we observed that PODI2 prevails in the ANT dataset already with default parameter values, and more so for $N > 100K$, $c < 3$ and $d > 6$. The number of LP problems solved by PODF2 is exactly |ND|. Although such a number is generally lower than the number of LP problems solved by PODI2, the size of the problems varies substantially. Indeed, the sufficient condition used by PODI2 for early pruning proves very effective for larger instances.



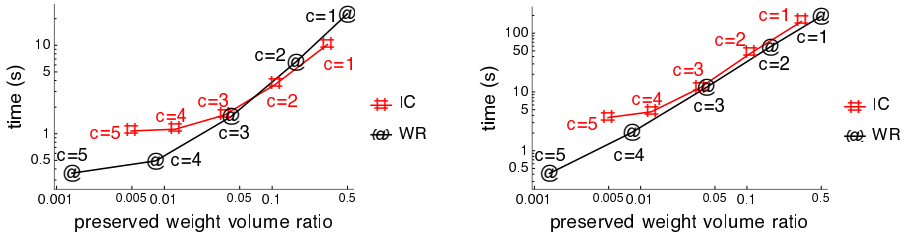
(a) Uniform distribution (UNI). (b) Anti-correlated distribution (ANT).

Fig. 14. Comparison between 1-phase and 2-phase algorithms for PO on smaller-scale datasets as the size N varies.

For instance, in the ANT dataset with default parameter values, ND consists of 2,616 tuples, 271 of which are in PO; PODI2 removes 96% of the non-potentially-optimal tuples before the last round, and 43% in the first five (very quick) rounds, where convex 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 solving a single instance of Problem (29) with a combination of $\sigma = 2,615$ tuples takes on average approximately 26 times as much as solving an instance with $\sigma = 366$ tuples, therefore the gains become apparent. Overall, PODI2 completes in 11s, whereas PODF2, which directly computes PO using LP Problem (29), requires 13s. The other datasets show similar trends, confirming a correlation between execution times and |ND|, independently of the distribution. This can be clearly observed in Figures 13a and 13b, which report the execution times of PODI2 and PODF2 for all values of |ND| occurring with all tested parameter values, merging observed results from all dataset distributions. In our experiments, PODF2 outperformed PODI2 whenever $|\text{ND}| < 2,000$ tuples, whereas PODI2 prevailed for larger values of |ND|. Figure 13c shows the relative gain of PODI2 over PODF2 as |ND| varies and indicates that, although PODF2 is much better than PODI2 for smaller values of |ND|, the difference in execution times is small in absolute terms (always less than 2s). On the contrary, PODI2 was up to 70% better than PODF2 for larger instances, showing also huge gains in absolute terms (more than 800s faster than PODF2 for the largest instance).

Given the intrinsic higher complexity of computing PO with respect to ND, it can be argued that the times incurred by PODI2 and PODF2 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 $d \leq 6$.

Non-ND-based methods require performing a large number of PO tests. In the case of non-incremental variants, the size of the LP problems to be solved is unfeasibly large even for the smallest instances we consider (when $N = 10K$, the first Problem (29) solved by PODF1 has already σ=9,999 variables, while Problem (28) solved by POPF1 has more than σ=9,999 constraints). As for ND-based methods, the dual PO test runs faster than the primal PO test. Therefore, the only sensible non-ND-based variant to consider is PODI1, and we compare it with PODI2 and PODF2 on smaller-scale synthetic datasets with varying size up to 100K tuples. Figure 14 reports the execution times incurred by the various algorithms (for PODI2 and PODF2 we now also include the time spent for computing ND through SVE1F) and shows that PODI1 is clearly outperformed by ND-based algorithms by at least one order of magnitude. This is due to the fact that PODI1 incurs a much larger number of LP problems to be solved. For instance, on the ANT dataset with $N = 10^5$, the number of LP problems solved by the various algorithms is: 843,872 for PODI1, 16,325 for PODI2, and 2,616 for PODF2.



(a) Computing ND with SVE1F on ANT. (b) Computing PO with PODI2 on ANT.

Fig. 15. Correlation between execution times and percentage of preserved volume in the space of weights as the number of constraints varies, both for weak rankings (WR) and for interval constraints (IC).

6.3 Other types of constraints

As in the case of weak rankings, our experiments with interval constraints detailed in Appendix C suggest that performance of our algorithms clearly depends on the *selectivity* of the constraints, i.e., the ratio between the (hyper-)volume of the polytope $\mathcal{W}(C)$ determined by the $|C| = c$ constraints in the space of weights and the volume of the $(d-1)$-simplex. Figure 15 shows the execution times for ND (as computed by SVE1F, Figure 15a) and PO (as computed by PODI2, Figure 15b) with the ANT dataset as the number of constraints c varies, both for weak rankings (WR) and for interval constraints (IC). The percentages of preserved volumes are not identical across the two kinds of constraints as c varies (for instance, when $c = 1$, the percentage is 50% for WR and 32.6% for IC, while, when $c = 5$, we have 0.13% for WR and 0.49% for IC). Still, the measured execution times tend to lie on the same line, independently of the kind of constraints (apart perhaps for smaller percentages of preserved volume, where small fluctuations in the measured times are proportionally more relevant). This suggests that performance is indeed independent of the kind of constraints and that the main impacting factor is the percentage of preserved volume.

6.4 Partitioning the dataset for parallel processing

The aim of this subsection is to explore the potentialities of parallel processing for computing ND and PO. To this end, we exploit Proposition 3.22 and consider a simulated environment with $m \geq 1$ processors with shared memory, in which the i-th processor is in charge of processing a partition r_i of the whole dataset r, such that all r_i's are (approximately) of the same size. The r_i partitions are determined by assigning tuples to processors in a round-robin fashion. We perform this set of experiments only with the most challenging datasets, namely ANT with $N = 10M$ (and default parameter values otherwise) and ANT with $d = 10$ (default values otherwise). For computing ND, we consider the SVE1F variant, which is the best one on the considered datasets; similarly, we consider PODI2 for computing PO.

According to Equation (19) in Proposition 3.22, when $r = r_1 \cup r_2 \cup \ldots \cup r_m$, $\text{ND}(r; \mathcal{F})$ can be computed in two steps:

(1) first, determine the so-called "local" $\text{ND}(r_i; \mathcal{F})$'s, for $1 \leq i \leq m$;
(2) then, apply ND on the union of the local $\text{ND}(r_i; \mathcal{F})$'s in order to obtain the "final" $\text{ND}(r, \mathcal{F})$.

Figure 16a shows results for the case $N = 10M$. In this Figure, each stacked bar shows in red the maximum of the times spent by the m processors in computing the corresponding local ND, while the pink part accounts for the cost of the second step ("final ND"), in which a single processor is in

charge of computing the final result.[13] The overall execution time is almost inversely proportional to the number m of partitions, with a remarkable speedup of 4.92 with $m = 8$ and of 5.76 with $m = 16$. The cost of the second step tends to increase with m, since, as Figure 19a shows, the sum of cardinalities of the local $\text{ND}(r_i; \mathcal{F})$'s ("merged ND's") grows with m, as expected.

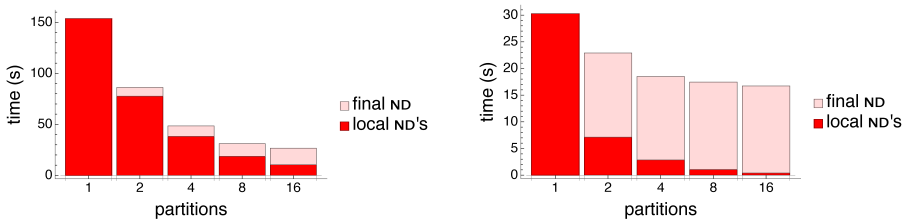
The situation for the case $d = 10$ is shown in Figure 16b, from which it is evident that parallelization has a reduced benefit if compared with the previous case. Even if the cost of the first step has a trend similar to the case $N = 10M$, computing the final result becomes the heavier component already when $m = 2$. Indeed, even if ND has a comparable cardinality in the two datasets, when $N = 10M$ it only represents 0.13% of the dataset, whereas, for $d = 10$, 18.29% of the tuples are in ND.

In the second step, we experimented with several optimization opportunities, in particular an m-way merge of the local ND's and a "partition-aware" $\mathcal{F}$-dominance test. When combining the local ND's, which are already locally sorted as a result of the computation, one can avoid sorting their union from the scratch by applying a simpler merge algorithm. Additionally, $\mathcal{F}$-dominance tests can be avoided altogether between all those pairs of tuples coming from the same partition (on which such tests have already been applied). Although these optimizations tends to reduce the overall execution time, their effect is negligible compared to the other cost factors.

For the computation of PO, we initially analyze two alternative strategies, whose different behaviors are shown in Figure 17 for the case $m = 4$. The first strategy, labeled "4/1" in the figure, first computes the local ND's (whose computation time is reported as the red part of the corresponding stacked bar), then it directly applies PODI2 on the union of such sets of tuples (i.e., first computing the final $\text{ND}(r; \mathcal{F})$, whose time is reported as the pink component, and then $\text{PO}(r; \mathcal{F})$, shown in green). Conversely, the second strategy, labeled "4/4" in the figure, also parallelizes the computation of the local PO's (green part of the stacked bar), which is followed by the computation of the final $\text{PO}(r; \mathcal{F})$ (light green part), thus reflecting the strategy induced by Equation (20) in Proposition 3.22. Note that, this second strategy, compared to the first one, largely reduces the number of tuples from which the final PO result has to be derived. Indeed, for both datasets, the second strategy largely outperforms the first one, which shows no visible improvement with respect to the non-parallelized baseline (labeled "1/1") when $d = 10$. Similar results occur also for other values of m.

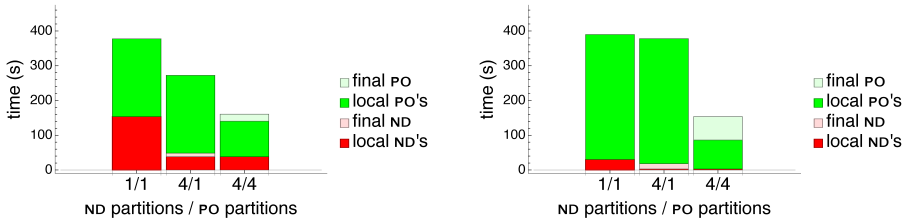
Figure 18 shows how our second strategy behaves with a variable number of processors. For both datasets, the availability of multiple processors results in significant improvements, attaining a maximum speedup of 3.3 for $m = 16$ when $N = 10M$ and a maximum of 2.9 for $m = 8$ when $d = 10$. However, as was the case with ND, the dataset with $N = 10M$ is less heavily impacted by the (non-parallelized) computation of the final result, shown as the light-green components of the bars, never exceeding 50% of the total time. Conversely, such a component becomes prevalent in the dataset with $d = 10$, attaining up to 90% of the total time when $m = 16$. This phenomenon can be explained by considering the fact that the cardinalities of the inputs (i.e., the "merged PO's") to the final PO computation are roughly the same in the two scenarios (see Figure 19), but the complexity of the problem is much lower when $d = 6$ (which is the case of the dataset with $N = 10M$) than when $d = 10$. Since the cardinality of the merged PO's grows with m, the benefits of parallelization are partially neutralized by the increased burden of the final PO computation for larger values of m.

[13] Note that, for the non-partitioned case $m = 1$, the concepts of "local" and "final" ND coincide, and we have labeled the bar as "local" for a more immediate comparison with the other cases. A similar observation applies also to the other figures in this subsection.



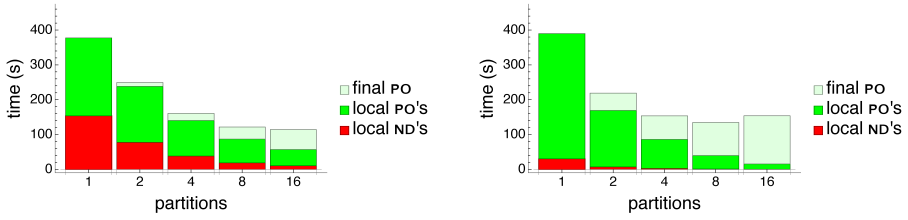
(a) Size $N = 10M$. (b) # of dimensions $d = 10$.

Fig. 16. Performance for computing ND with SVE1F while varying number of partitions for parallel processing with default parameter values except for (a) size $N = 10M$, (b) dimensions $d = 10$.



(a) Size $N = 10M$. (b) # of dimensions $d = 10$.

Fig. 17. Performance for computing PO with PODI2, comparing no partitioning (left-most bars), partitioning with 4 partitions for ND (central bars) and for both ND and PO (right-most bars). Default parameter values except for (a) size $N = 10M$, (b) dimensions $d = 10$.



(a) Size $N = 10M$. (b) # of dimensions $d = 10$.

Fig. 18. Performance for computing PO with PODI2 while varying number of partitions for parallel processing with default parameter values except for (a) size $N = 10M$, (b) dimensions $d = 10$.

6.5 Summary of experimental findings on performance

We conclude by summarizing the main findings from the previous subsections.

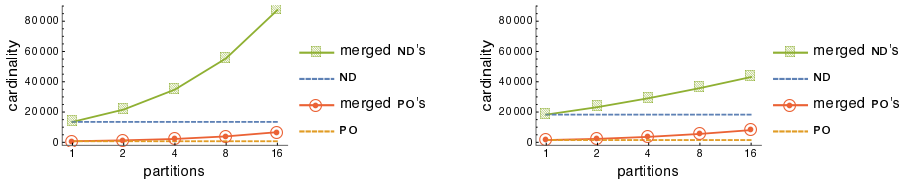
Computation of ND

(1) sorting the input dataset is always beneficial;

(2) the Vertex Enumeration (VE) strategy largely outperforms the approach based on Linear Programming (LP);

(3) computing ND starting from the skyline (2-phase approach) does not pay off in challenging scenarios.

(4) if the set of constraints has a reasonably high selectivity (e.g., $c \geq 3$ in our scenarios), interleaving dominance and $\mathcal{F}$-dominance tests (as in SVE1F) leads to better performance than first checking dominance on all tuples and then $\mathcal{F}$-dominance on the remaining tuples.



(a) Size $N = 10M$. (b) # of dimensions $d = 10$.

Fig. 19. Cardinalities of the union of the local ND's and PO's as the number of partitions varies. Default parameter values except for (a) size $N = 10M$, (b) dimensions $d = 10$.

Computation of PO

(1) computing PO starting from ND (ND-based variants) is always the best alternative;
(2) using the dual PO test of Theorem 4.7 leads to better performances than the primal test of Theorem 4.5, with the maximum gain exceeding 30%;
(3) the variants based on the incremental approach are preferable for challenging problem instances, as measured by the cardinality of ND.

From these observations, we can derive the following guidelines for the choice of the most appropriate algorithms, depending on the scenario at hand:

- generally, *the best variant for computing* ND *on challenging datasets is* SVE1F, which, for default parameter values ($d = 6$, $c = 3$, $N = 100K$, $p = 1$) requires 1.54s on the ANT dataset;
- however, if we consider *challenging datasets and constraints with low selectivity then the best variant is* SVE1, which prevails over SVE1F on the ANT dataset when $c = 1$ (default parameter values otherwise) by a 12% margin;
- for less challenging scenarios, SVE2 usually prevails (see, e.g., Figure 10a and Appendix C);
- *the best variant for computing* PO *in scenarios in which* ND *is large is* PODI2, which executes in 12.69s on the ANT dataset with default parameter values;
- for smaller values of |ND|, PODF2 generally prevails over PODI2, although, in this case, the performance gains are not as significant.

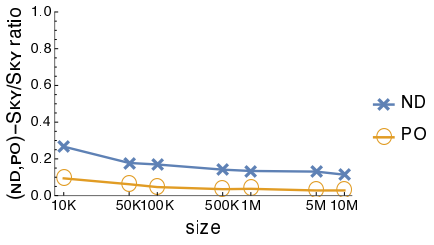
6.6 Comparison with skyline queries

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 8 in Section 6.2).

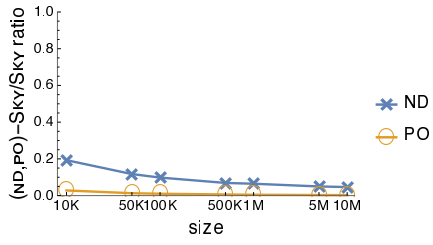
Figures 20 and 21 show this ratio in several scenarios for the ANT, UNI and NBA datasets. The results indicate that F-skylines are always much more effective than skylines, and PO is more effective than ND, as expected. Table 8 shows that, for default values of the parameters, the effectiveness of F-skylines is already remarkable, with the largest reductions in ANT (9.8% of the skyline points are in ND and only 1% in PO).

Figures 20a and 20b 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 $N = 10M$ points; in this setting, the skyline has 289812 points, 13490 of which (4.6%) are retained by ND, and only 710 (0.2%) 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 [23].

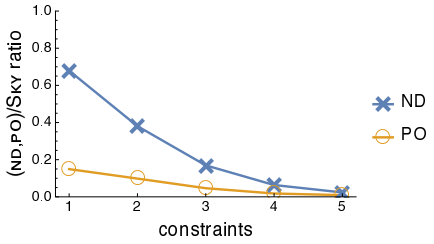
Figures 20c, 20d, and 21a show that the effectiveness of F-skylines steadily improves as the number of constraints c increases. This is mainly due to the fact that each constraint reduces the



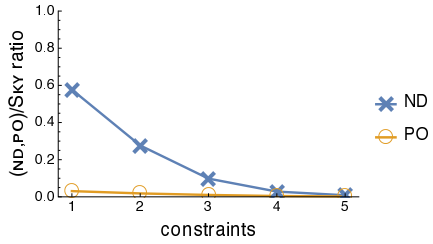
(a) UNI: dataset size N varies.



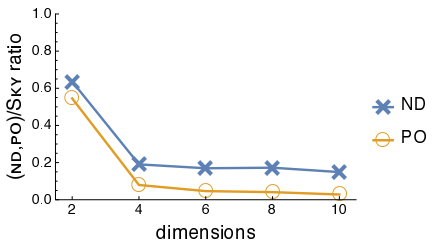
(b) ANT: dataset size N varies.



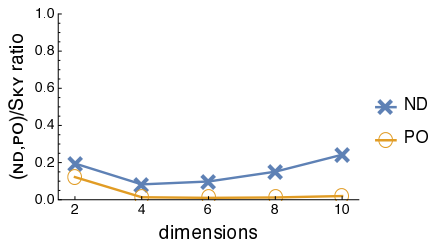
(c) UNI: # of constraints c varies.



(d) ANT: # of constraints c varies.

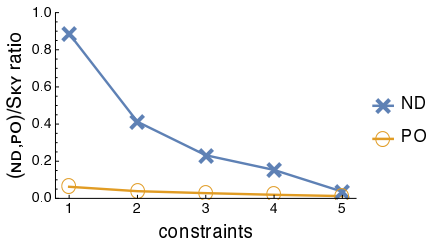


(e) UNI: # of dimensions d varies.

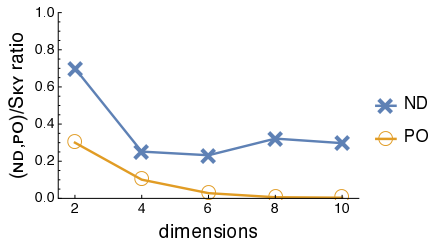


(f) ANT: # of dimensions d varies.

Fig. 20. Cardinality ratio between F-skylines (PO, ND) and SKY: UNI on the left; ANT on the right.



(a) NBA: # of constraints c varies.



(b) NBA: # of dimensions d varies.

Fig. 21. Cardinality ratio between F-skylines (PO, ND) and SKY for the NBA dataset.

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 3.14. The experiments also show that PO proves very effective even with few constraints, with the most dramatic improvement with respect to 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 (hyper-)volume of the space of weights $\mathcal{W}(C)$ caused by constraints and the reduction of tuples caused by F-skylines. Figure 22 confirms a clear correlation between the ratio of points in ND and the ratio of preserved

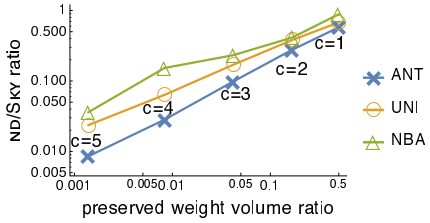
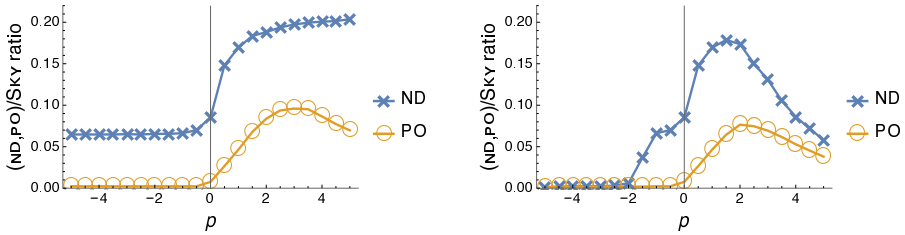


Fig. 22. Correlation between ND/SKY cardinality ratio and percentage of preserved volume in the space of weights as the number of constraints c varies.

weight volume, i.e., the ratio between the (hyper-)volume of the polytope $\mathcal{W}(C)$ determined by the $|C| = c$ 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 20e, 20f, and 21b. 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.



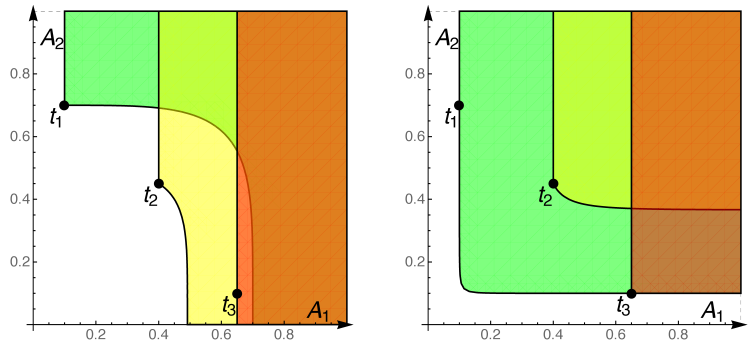
(a) UNI. (b) UNI (with non-zero weights).

Fig. 23. Cardinality ratio between F-skylines (PO, ND) and SKY as parameter p of M_p^W means varies; (a): weak rankings; (b): weak rankings with non-zero weights.

Finally, Figure 23a shows the effect of parameter p of M_p^W means (see Equations (30) and (31)). We only show the results for the UNI dataset, since ANT and NBA have similar behaviors. The |ND|/|SKY| ratio grows steadily as p grows, and then saturates for high values of p ($p \approx 4$). This asymptotic behavior can be explained by observing that the shape of the $\mathcal{F}$-dominance regions tends to stabilize for large values of $|p|$ (see also Figure 24). As for PO, the |PO|/|SKY| ratio is minimal for negative values of p (where less than 10 tuples are in PO). For $p > 0$, the ratio grows with p until a maximum is reached for a value of $p \approx 3$, after which it rapidly decreases.

Note that when all the weights are non-zero and $p \to \infty$, the score of a tuple t tends to $\max\{t[A_1], \ldots, t[A_d]\}$ (or min when $p \to -\infty$), in which case both ND and PO would tend to the set with just the tuple (possibly tied) with the best (i.e., lowest) maximum (minimum) attribute value. However, our default constraints allow some of the weights to be 0, therefore preventing this behavior. Figure 23b shows what happens when additional constraints imposing all weights to be non-zero are included: for high values of $|p|$, both |PO| and |ND| tend to one single tuple, as expected.

Since in Figure 23 the cardinality of SKY is clearly independent of p, the graphs also suggest that, in the general case, the cardinalities of ND and PO do not display a monotonic behavior when varying p. Indeed, as further discussed in Appendix A, this depends on the fact that there is no inclusion between the $\mathcal{F}$-dominance regions of a tuple when p varie



(a) $p = 5$ (b) $p = -5$

Fig. 24. $\mathcal{F}$-dominance regions for three tuples when $p = 5$ (left) and $p = -5$ (right).

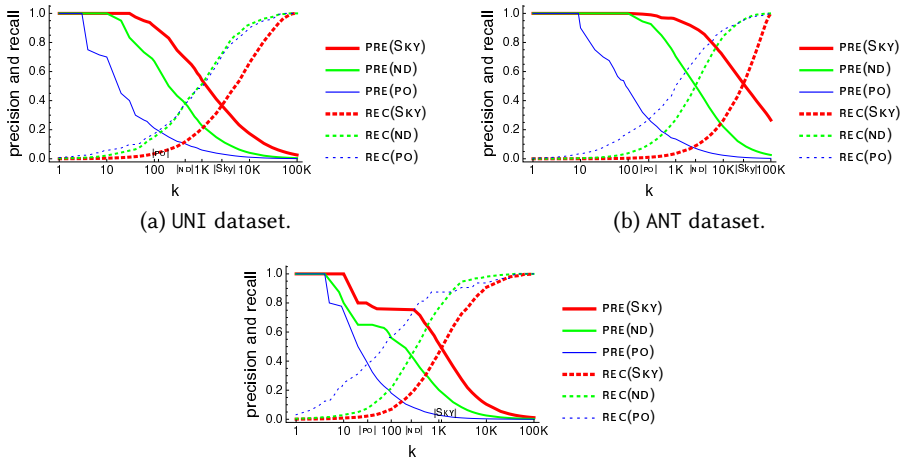
Dataset	SKY by SFS	ND by SVE1F	PO by PODI2
ANT	11.17 (98%)	1.54	12.69
UNI	0.34 (94%)	0.36	1.41
NBA	0.58 (98%)	0.65	1.74

Table 9. Execution times (in seconds) of SKY, ND, PO with default values; in brackets, the percentage with respect to the execution time of SVE2.

In order to provide more intuition on the behavior of the |ND|/|SKY| ratio (and, consequently, on the |PO|/|SKY| ratio) when zero-weights are possible, we refer to Figure 24, showing the $\mathcal{F}$-dominance regions for three tuples with constraints $\{w_1 \geq w_2\}$ with $p = 5$ (Figure 24a) and $p = -5$ (Figure 24b).

With respect to the reference linear case $p = 1$, when p grows, the $\mathcal{F}$-dominance region of a tuple t below the main diagonal (such as t_3) tends to include all tuples s such that $s[A_1] \geq t[A_1]$ (i.e., only the first coordinate matters), and this region strictly includes the $\mathcal{F}$-dominance region of t when $p = 1$. In this case, we therefore expect tuple t_3 to $\mathcal{F}$-dominate more tuples than when $p = 1$. On the other hand, for a tuple t above the main diagonal (such as t_1 and t_2) a somewhat opposite effect may occur, in that the volume of the $\mathcal{F}$-dominance region may reduce depending on how far the tuple is from the diagonal. For instance, this reduction can be appreciated for t_1 (far from the diagonal), but not for t_2 (close to the diagonal). A rather different situation occurs when $p \to -\infty$: now, the $\mathcal{F}$-dominance region of tuples below the main diagonal reduces to the minimum possible (i.e., their dominance region), whereas tuples above the diagonal tend to enlarge their $\mathcal{F}$-dominance region, again depending on how far they are from the diagonal. The different phenomenon occurring when $p \to \infty$ and $p \to -\infty$ explains why the asymptotic values of |ND| are likely to be different in practice.

We conclude this set of experiments on p by observing that, for large values of $|p|$ (such as $|p| > 5$), numerical instability problems are encountered by the LP solver, thus making the results unreliable for those values. However, this is not an actual limitation, since, when $|p| = 5$, the ND and PO operators already exhibit a stable behavior, which is not expected to change for larger values of $|p|$, as confirmed by the fact that the $\mathcal{F}$-dominance regions observed in Figure 24 for $|p| = 5$ are not going to change significantly for larger values of $|p|$.



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

(c) NBA dataset.

Fig. 25. Precision and recall of top-k queries with respect to SKY, ND, and PO. Centroid of the $\mathcal{W}(C)$ as weight vector; default parameter values.

We now compare F-skylines and skylines in terms of execution times. Table 9 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 9 for the ANT case. Table 9 also shows that computing PO incurs only a moderate overhead for all the considered datasets with default parameter values.

Overall, this confirms that F-skyline operators can be safely used in place of standard skylines, especially considering their increased flexibility in controlling the size of the result.

6.7 Comparison with ranking queries

We first remark that, to the best of our knowledge, a detailed comparative analysis between the results of ranking (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. In the following we provide some intuition on the relationship between results yielded by our operators and those of 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 with respect to a scoring function f. The precision of $\mathcal{T}_k(r; f)$ with respect to 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.

Clearly, if there is a small overlap between top-k and skyline results, we expect a similar or smaller overlap with F-skylines. Moreover, the overlap will largely depend on the specific scoring function f, and, in the case of F-skylines, also on the family of scoring functions $\mathcal{F}$. Intuitively, the more f is "dissimilar" to $\mathcal{F}$, the more F-skyline and top-k query results will differ.

Figure 25 shows precision and recall when f is a weighted sum with the centroid of the polytope $\mathcal{W}(C)$ 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, PRE(PO) $\leq$ PRE(ND) $\leq$ PRE(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 F-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 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 F-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 $\mathcal{W}(C)$ leads to lower precision as well as recall.

Similarly to skyline queries, F-skylines pay the increased capability of returning interesting results at the price of a higher computational overhead with respect to ranking queries. In all our experiments, in which the top-k algorithm was implemented using a max-heap along the lines of [8], ranking queries required only a fraction of the execution time of SVE1F (between 0.1% and 7%), as expected.

7 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.

Among the methods aiming to extend skyline queries with user preferences, we mention p-skylines and trade-off skylines, which 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). P-skyline (or Prioritized skyline) queries [32] 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 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 [31]. Trade-off skylines [30] consider *qualitative* trade-offs, i.e., no numerical considerations are present. While this allows extending the applicability of their algorithms to categorical attributes, the price to be paid is an increased computational complexity, which makes them hardly applicable to datasets as large as those we consider in this paper.

Several techniques have been proposed for reducing the skyline size, a recent survey of which can be found in [29]. Notice, however, that none of these methods is conceived to accommodate user preferences as we do with F-skylines. Distance-based representative skylines [43] 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) [28] 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 [44] 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.

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., [34, 45]. Along these lines, [42] studies representative orderings (such as the most probable ordering) and their stability with respect to a change of parameter values, by assuming that the set of parameters (weights) is a random variable with a uniform distribution. Also partially related to the present work is the notion of skyline over probabilistic data [3, 36], which we discuss as an orthogonal line of research in Section 8.

When the scoring function f is linear both in the weights *and* in the attribute values (i.e., f is a weighted arithmetic mean), the top-1 result is guaranteed to lie on the convex hull of the dataset.[14] This is exploited by the *Onion* technique for indexing purposes [9]. However, for general MLW functions and constraints, which we consider in this paper, the Onion technique cannot be adopted. This stems from the observation that, in general, neither ND nor PO are subsets of the convex hull.

F-skylines have a small overlap with *dynamic skylines* [35], but important differences exist. In a dynamic skyline query, one considers a *finite* set of q monotone functions, $\Phi = \{\phi_1, \ldots, \phi_q\}$, defined on the tuple attributes and, possibly, on user-provided *query points*, and then establishes that tuple t Φ-dominates tuple s iff $\phi_i(t) \leq \phi_i(s)$ holds for $i \in \{1, \ldots, q\}$, with at least one inequality being strict. A remarkable application of dynamic skylines is on spatial datasets [40], in which the q functions are the distances of a tuple t from q query points. In the extreme case in which no user-defined query points are given (i.e., Φ only consists of monotone functions of the tuple attributes), the dynamic skyline of a relation r coincides with $\text{ND}(r; \Phi)$, as it is immediate to verify. However, in general, the two types of queries are incomparable, since, on one hand, dynamic skylines can rely on external query points and, on the other hand, F-skylines can deal with an *infinite* family of scoring functions (and constraints on them). It is interesting to observe that our approach can also be applied so as to extend dynamic skylines with user preferences, as we have done with standard skylines. To see this, consider for instance $\Phi = \{\phi_1, \phi_2\}$ and the set of (functions of) functions $\mathcal{F} = \{w_1\phi_1 + w_2\phi_2 | w_1 \geq w_2\}$. Then, one can define ND and PO as in the classical (static) case, thus returning, respectively, the set of non-$\mathcal{F}$-dominated tuples and the set of tuples that are optimal for at least one combination of ϕ_1 and ϕ_2 respecting the constraint.

In this paper we have assumed, for the sake of tuple-distinguishability, that for each attribute A_i in $\mathcal{A} = \{A_1, \ldots, A_d\}$ there exists at least one function in $\mathcal{F}$ that depends on A_i. When this is not the case, we fall into the same scenario considered by *subspace skylines* [37], which compute the skyline, $\text{SKY}_{\mathcal{B}}(r)$, by considering each proper (non-empty) subset $\mathcal{B} \subset \mathcal{A}$ of the attributes. Let $\text{MF}_{\mathcal{B}}$ be the set of all monotone scoring functions defined on $\mathcal{B}$. Then, $\text{SKY}_{\mathcal{B}}(r) = \text{ND}(r; \text{MF}_{\mathcal{B}})$, whereas $\text{PO}(r; \text{MF}_{\mathcal{B}})$, due to the loss of tuple-distinguishability in $\mathcal{B}$, omits all the tuples t that in the $\mathcal{B}$ subspace collapse with some other tuple. In all these cases, as well as in the more general case in which a subset of MLW functions $\mathcal{F} \subseteq \text{MF}_{\mathcal{B}}$ is used, $\text{ND}(r; \mathcal{F})$ and $\text{PO}(r; \mathcal{F})$ can be computed exactly as described in Section 4.3.

Since their introduction in [12], the very concept of $\mathcal{F}$-dominance and the operators relying on it have inspired several subsequent works aiming to extract interesting results from large datasets.

In [33], the authors consider the extension of our ND and PO operators to computing the set of tuples that are $\mathcal{F}$-dominated by less than k tuples (ND_k) and the set of tuples that are in the top-k result for some function in $\mathcal{F}$ (PO_k), respectively. The computation of ND_k assumes that the dataset is indexed by an R-tree, and their algorithm is a variant of the BBS algorithm used to

[14] The convex hull is the smallest convex polytope that encloses all tuples in a dataset [15].

compute the k-skyband (the generalization of the skyline to the case $k \geq 1$): the index is traversed by considering a priority queue whose entries are ordered using, as we do, a linear scoring function with weights equal to the coordinates of the centroid of $\mathcal{W}(C)$, and $\mathcal{F}$-dominance is checked by an analog of our test based on the vertices of the $\mathcal{W}(C)$ polytope. When $k = 1$ their algorithm for computing PO_1 reduces to the PO variant described in this paper based on the primal LP test without incrementality (i.e., POPF2), which we have shown to yield inferior performance with respect to the dual-based approach with incrementality (i.e., PODI2).

In [13] the concept of $\mathcal{F}$-dominance has been exploited to derive a novel, *instance-optimal* algorithm [19], called FSA, for solving multi-source, distributed top-k queries, when only constraints on weights (rather than a specific scoring function) are available. In [13] it is also shown that FSA generalizes and includes as special cases two well-known algorithms for computing top-k queries in a distributed scenario, i.e., Fagin's Algorithm [18] and the Threshold Algorithm [19].

There are numerous application scenarios that share the common goal of extracting the most relevant objects from a large set of alternatives. For instance, this is the case with recommender systems as well as with *feature selection* approaches, in which, for avoiding the so-called curse of dimensionality, the aim is to select the most relevant features for, e.g., training a classifier. Assuming that *d filters* are available that assign a relevance score to the N features under consideration, [4] describes the application of $\mathcal{F}$-dominance to the problem of feature selection, with (range-tolerance) interval constraints on weights. Initial experiments involving several UCI datasets and filters show the benefits of this approach with respect to other methods based on classical top-k selection.

This paper substantially extends the work in [12], in which the concept of $\mathcal{F}$-dominance was originally introduced, together with the ND and PO operators. In particular, besides adding rigorous proofs to all our formal results, here we provide a more detailed analysis of the fundamental properties of the operators (Section 3.1), several novel properties extending the applicability of F-skylines (Section 3.2), along with numerous examples illustrating the properties at hand. While [12] only focused on scoring functions in the form of weighted L_p norms, here we consider a much more general class, which we call MLW functions (Section 4), in which attribute values can be subject to arbitrary, possibly different, monotone transforms. We show that all our formal results generalize to this larger class, including a novel theorem providing an alternative way of computing potentially optimal tuples. We also provide a formal justification to ND-based algorithms for computing PO (which was only implicitly assumed in [12]). The case of non-tuple-distinguishing sets of functions was not considered at all in [12], whereas here we provide a comprehensive analysis, in Section 4.3 for the case of infinite sets of scoring functions, which is the main focus of our work, and in Appendix B for the general case in which the set can also be finite. The computation of PO in [12] was based on a single algorithm, whereas here we introduce and experimentally evaluate eight new different variants. Our experiments focus on a practically relevant subset of MLW functions, i.e., weighted power means, whose distinctive properties (only sketched in [12]) are studied in Appendix A. The experiments cover a wide variety of scenarios that were not considered in the original conference paper, such as larger datasets (up to 10M tuples) and different data distributions (also including correlated data and clustered data, for both real and synthetic variants). We experiment with a different type of constraints (interval constraints), showing, in Section 6.3, that the selectivity of the constraints, independently of their kind, is the factor with the heaviest impact on performance. In Appendix C, we also compare the different kinds of constraints from a geometric point of view. Appendix C also discusses how performance is affected by non-linearity in the scoring function. As a novel contribution, in Section 6.4, we assess the impact of the availability of multiple processors for parallelizing the computation of ND and PO, along the lines of Proposition 3.22. The detailed comparisons with dynamic skylines and subspaces skylines offered in this section are also new.

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 F-skylines do, is therefore a viable way to alleviate the problem. To this end, a large body of techniques have been investigated in Multi-Attribute Decision Theory within the so-called *preference programming* [39]. In general, preference programming methods deal with the problem of assisting decision makers when their preferences are incomplete. F-skyline algorithms represent, as far as we know, the first successful attempt to apply a preference programming method on large datasets.

8 FINAL DISCUSSION

In this paper, we have introduced the notion of flexible skyline queries (F-skylines), aiming to extend the skyline framework with user preferences expressed by means of constraints on the weights of scoring functions. We have done so through the novel concept of $\mathcal{F}$-dominance, i.e., dominance with respect to a *family* of scoring functions $\mathcal{F}$. We have then introduced two F-skyline operators, respectively implementing the notions of non-dominated (ND) and potentially optimal (PO) tuples with respect to $\mathcal{F}$.

We have shown that both ND and PO are very effective in focusing on tuples of interest, even in very large datasets, often leading to considerably smaller result sizes than standard skylines. Overall, we have developed several variants for efficiently computing both ND and PO, and characterized their behaviors in a variety of scenarios.

The concepts and methods we have presented in this paper can be extended along several directions, which we briefly describe in the following.

For the sake of general applicability, in this paper we have considered algorithms that sequentially scan the input dataset, possibly after pre-sorting it. Clearly, the notion of $\mathcal{F}$-dominance could certainly be used in conjunction with more refined optimization techniques, such as those characterizing the LESS [24] and SaLSa [2] algorithms, as well as ad hoc algorithms for parallel environments, such as [5] – developed for classical skyline queries –, which would improve the efficiency of the variants based on sorting of the dataset. Similarly, one might extend index-based approaches, such as the BBS algorithm [35], for supporting F-skyline queries.

A different line of investigation would be to apply the notion of $\mathcal{F}$-dominance to variants of skyline queries. Among them, we mention reverse skyline queries [16] and skylines for probabilistic data [3, 36]. In all these cases, the possibility of working with a specific family of scoring functions could reveal interesting properties of the data.

As observed in Section 7, the notion of F-skyline can also be applied when the coordinates of interest are dynamically defined, as is the case with dynamic skylines. An in-depth analysis of this enlarged scenario, which greatly extends the applicability of our techniques beyond the family of MLW functions considered in this paper, is another interesting line of research.

Our F-skylines, much in the same way as ranking queries, are tailored to deal with numeric attributes. Clearly, by mapping categorical attributes into numerical domains, our techniques would still be applicable. Alternatively, one could imagine a mixed scenario, in which numerical and categorical data coexist, and in which F-skyline concepts are applied only to the numerical part, while standard dominance criteria are applied to the categorical part (which, however, needs to consist of ordinal attributes). The extension of our techniques to this case, which would require a fusion of the two parts, is also an open problem.

Furthermore, the application of F-skylines to different computer science areas could be worth investigating, as exemplified by the approach described in [4].

According to our framework, the specification of the family of scoring functions $\mathcal{F}$ to use can be given through constraints on weights and the marginal scores of tuples (given by our g_i functions

in (21)). It is apparent (as well as agreed upon in the literature [42]) that providing precise weight values is much harder than specifying constraints on weights. Although definitely important, the issue of providing the "right" weight constraints is completely orthogonal to our work. This task is tightly related to the issue of *preference elicitation*, which is covered by a large body of research in the Decision Theory field. Similarly, methods for choosing the "right" marginal scores abound in the Decision Theory literature (see, e.g., [41]), and have also been incorporated in some preference programming methods [39]. An interesting development of our work would be that of extending F-skyline algorithms to the more challenging scenario in which both weights *and* marginal scores are partially specified.

Finally, we observe that, when the result of an F-skyline operator is too large, one could consider using one of the methods developed for controlling the cardinality of the skyline, which were described in the related work section and can be applied orthogonally to F-skylines. For instance, the concept underlying distance-based representative skylines [43] can also immediately be applied to F-skylines. An interesting research issue would be to devise efficient algorithms in which the final result is obtained without first computing the full F-skyline result.

ELECTRONIC APPENDIX

The electronic appendix to this article can be accessed in the ACM Digital Library.

REFERENCES

[1] E. Cabral Balreira, Olga Kosheleva, and Vladik Kreinovich. 2014. Algorithmics of Checking whether a Mapping Is Injective, Surjective, and/or Bijective. In *Constraint Programming and Decision Making*. 1–7. https://doi.org/10.1007/978-3-319-04280-0_1

[2] Ilaria Bartolini, Paolo Ciaccia, and Marco Patella. 2008. Efficient sort-based skyline evaluation. *ACM Trans. Database Syst.* 33, 4 (2008), 31:1–31:49.

[3] Ilaria Bartolini, Paolo Ciaccia, and Marco Patella. 2014. Domination in the Probabilistic World: Computing Skylines for Arbitrary Correlations and Ranking Semantics. *ACM Trans. Database Syst.* 39, 2 (2014), 14:1–14:45.

[4] Marcos V. N. Bedo, Paolo Ciaccia, Davide Martinenghi, and Daniel de Oliveira. 2019. A k-Skyband Approach for Feature Selection. In *Similarity Search and Applications - 12th International Conference, SISAP 2019, Newark, NJ, USA, October 2-4, 2019, Proceedings*. 160–168. https://doi.org/10.1007/978-3-030-32047-8_15

[5] Kenneth S. Bøgh, Sean Chester, and Ira Assent. 2016. SkyAlign: a portable, work-efficient skyline algorithm for multicore and GPU architectures. *VLDB J.* 25, 6 (2016), 817–841. https://doi.org/10.1007/s00778-016-0438-1

[6] 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

[7] L.E.J. Brouwer. 1912/3. Invariantz des n-dimensionalen Gebiets. *Math. Ann.* 71, 2 (1912/3), 305–313.

[8] Michael J. Carey and Donald Kossmann. 1997. On Saying "Enough Already!" in SQL. In *SIGMOD 1997, Proceedings ACM SIGMOD International Conference on Management of Data, May 13-15, 1997, Tucson, Arizona, USA*. 219–230. https://doi.org/10.1145/253260.253302

[9] 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

[10] Jan Chomicki, Paolo Ciaccia, and Niccolo' Meneghetti. 2013. Skyline queries, front and back. *SIGMOD Record* 42, 3 (2013), 6–18. https://doi.org/10.1145/2536669.2536671

[11] Jan Chomicki, Parke Godfrey, Jarek Gryz, and Dongming Liang. 2003. Skyline with Presorting. In *ICDE*. 717–719. https://doi.org/10.1109/ICDE.2003.1260846

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

[13] 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

[14] Eleonora Ciceri, Piero Fraternali, Davide Martinenghi, and Marco Tagliasacchi. 2016. Crowdsourcing for Top-K Query Processing over Uncertain Data. *TKDE* 28, 1 (2016), 41–53. https://doi.org/10.1109/TKDE.2015.2462357

[15] Mark de Berg, Otfried Cheong, Marc J. van Kreveld, and Mark H. Overmars. 2008. *Computational geometry: algorithms and applications, 3rd Edition*. Springer. http://www.worldcat.org/oclc/227584184

[16] Evangelos Dellis and Bernhard Seeger. 2007. Efficient Computation of Reverse Skyline Queries. In *VLDB*. ACM, 291–302.

[17] Yun Seong Eum, Kyung Sam Park, and Soung Hie Kim. 2001. Establishing dominance and potential optimality in multi-criteria analysis with imprecise weight and value. *Computers & Operations Research* 28 (2001), 397–409.

[18] Ronald Fagin. 1996. Combining Fuzzy Information from Multiple Systems. In *PODS*. 216–226. https://doi.org/10.1145/237661.237715

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

[20] Alex Alves Freitas. 2004. A critical review of multi-objective optimization in data mining: a position paper. *SIGKDD Explorations* 6, 2 (2004), 77–86. https://doi.org/10.1145/1046456.1046467

[21] Volker Gaede and Oliver Günther. 1998. Multidimensional Access Methods. *ACM Comput. Surv.* 30, 2 (1998), 170–231. https://doi.org/10.1145/280277.280279

[22] Bernd Gärtner and Jiří Matoušek. 2006. *Understanding and Using Linear Programming*. Springer. 81–104 pages.

[23] 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

[24] Parke Godfrey, Ryan Shipley, and Jarek Gryz. 2007. Algorithms and analyses for maximal vector computation. *VLDB J.* 16, 1 (2007), 5–28. https://doi.org/10.1007/s00778-006-0029-7

[25] 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

[26] Volker Kaibel and Marc E. Pfetsch. 2003. Some Algorithmic Problems in Polytope Theory. In *Algebra, Geometry, and Software Systems [outcome of a Dagstuhl seminar]*. 23–47.

[27] Kenneth A. Kaufman and Ryszard S. Michalski. 1999. Learning from Inconsistent and Noisy Data: The AQ18 Approach. In *ISMIS (Lecture Notes in Computer Science)*, Vol. 1609. Springer, 411–419.

[28] 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

[29] 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

[30] Christoph Lofi, Ulrich Güntzer, and Wolf-Tilo Balke. 2010. Efficient computation of trade-off skylines. In *EDBT 2010, 13th International Conference on Extending Database Technology, Lausanne, Switzerland, March 22-26, 2010, Proceedings*. 597–608. https://doi.org/10.1145/1739041.1739112

[31] Niccolo' Meneghetti, Denis Mindolin, Paolo Ciaccia, and Jan Chomicki. 2015. Output-sensitive Evaluation of Prioritized Skyline Queries. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, Melbourne, Victoria, Australia, May 31 - June 4, 2015*. 1955–1967. https://doi.org/10.1145/2723372.2723736

[32] Denis Mindolin and Jan Chomicki. 2011. Preference elicitation in prioritized skyline queries. *VLDB J.* 20, 2 (2011), 157–182. https://doi.org/10.1007/s00778-011-0227-9

[33] 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

[34] 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

[35] Dimitris Papadias, Yufei Tao, Greg Fu, and Bernhard Seeger. 2005. Progressive skyline computation in database systems. *TODS* 30, 1 (2005), 41–82. https://doi.org/10.1145/1061318.1061320

[36] Jian Pei, Bin Jiang, Xuemin Lin, and Yidong Yuan. 2007. Probabilistic Skylines on Uncertain Data. In *VLDB*. ACM, 15–26.

[37] Jian Pei, Yidong Yuan, Xuemin Lin, Wen Jin, Martin Ester, Qing Liu, Wei Wang, Yufei Tao, Jeffrey Xu Yu, and Qing Zhang. 2006. Towards multidimensional subspace skyline analysis. *TODS* 31, 4 (2006), 1335–1381. https://doi.org/10.1145/1189774

[38] Li Qian, Jinyang Gao, and H. V. Jagadish. 2015. Learning User Preferences By Adaptive Pairwise Comparison. *PVLDB* 8, 11 (2015), 1322–1333. https://doi.org/10.14778/2809974.2809992

[39] Ahti Salo and Raimo P. Hämäläinen. 2010. Preference Programming – Multicriteria Weighting Models under Incomplete Information. In *Handbook of Multicriteria Analysis*. Springer, Chapter 4, 167–187.

[40] Mehdi Sharifzadeh and Cyrus Shahabi. 2006. The Spatial Skyline Queries. In *VLDB*. 751–762.

[41] Yannis Siskos, Evangelos Grigoroudis, and Nikolaos F. Matsatsinis. 2005. *UTA Methods.* Springer New York, 297–334. https://doi.org/10.1007/0-387-23081-5_8

[42] Mohamed A. Soliman, Ihab F. Ilyas, Davide Martinenghi, and Marco Tagliasacchi. 2011. Ranking with uncertain scoring functions: semantics and sensitivity measures. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2011, Athens, Greece, June 12-16, 2011.* 805–816. https://doi.org/10.1145/1989323.1989408

[43] 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

[44] 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

[45] Jilian Zhang, Kyriakos Mouratidis, and HweeHwa Pang. 2014. Global immutable region computation. In *International Conference on Management of Data, SIGMOD 2014, Snowbird, UT, USA, June 22-27, 2014.* 1151–1162. https://doi.org/10.1145/2588555.2610508

ELECTRONIC APPENDIX

In this online appendix, the four sections report, respectively, a detailed analysis of the behavior of weighted power means (Section A), an extension of our results to cover all cases in which the set of scoring functions fails to distinguish all possible tuples (Section B), additional experiments (Section C), and all the proofs of the claims included in the main body of the paper (Section D).

A THE CASE OF WEIGHTED POWER MEANS

In this appendix, we analyze in more detail the relevant case of weighted power means, which, as defined in Section 6, are MLW functions of the following form:

$$M_p^W(t) = \left(\sum_{i=1}^{d} w_i t[A_i]^p\right)^{1/p}, \quad p \neq 0$$

$$M_0^W(t) = \prod_{i=1}^{d} t[A_i]^{w_i}.$$

We define the family $\mathcal{M}_p$ of weighted power means for some fixed value of p as follows:

$$\mathcal{M}_p = \{M_p^W \mid W \in \mathcal{W}\}. \tag{32}$$

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

Theorem A.1. *For every $p \in \mathbb{R}$ and every r, $\text{ND}(r; \mathcal{M}_p) = \text{SKY}(r)$.*

Proof. By Proposition 3.11 it follows that $\text{ND}(r; \mathcal{M}_p) \subseteq \text{SKY}(r)$. Now we also need to show that every tuple $t \in \text{SKY}(r)$ also belongs to $\text{ND}(r; \mathcal{M}_p)$. We know that $\nexists s \in r.\ s \prec t$, and assume, by contradiction, that $\exists s \in r.\ s \prec_{\mathcal{M}_p} t$, with $s \neq t$. Therefore, $\forall f \in \mathcal{M}_p.\ f(s) \leq f(t)$. Now, let us indicate with $W^{(i)} = (w_1^{(i)}, \ldots, w_d^{(i)}) \in \mathcal{W}$ the weight vector such that $w_i^{(i)} = 1$ and, for $j \neq i$, $w_j^{(i)} = 0$. Since $s \prec_{\mathcal{M}_p} t$, we must have $M_p^{W^{(i)}}(s) \leq M_p^{W^{(i)}}(t)$ for $1 \leq i \leq d$, i.e., we must have $s[A_i] \leq t[A_i]$ for $1 \leq i \leq d$, which entails $s \prec t$ since $t \neq s$. Contradiction. □

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

Theorem A.2. *For every $p \in \mathbb{R}$, there exists a relation r such that*

$$\text{PO}(r; \mathcal{M}_p) \subset \text{SKY}(r). \tag{33}$$

Proof. Consider an instance $r = \{t_1, t_2, t\}$, with $t_1 = \langle 1, 0\rangle$, $t_2 = \langle 0, 1\rangle$, $t = \langle 1-\epsilon, 1-\epsilon\rangle$, where $0 < \epsilon < 1$. Clearly, $\text{SKY}(r) = r$, since no tuple in r is dominated by any other tuple in r. Now, for every $p \in \mathbb{R}$, we can choose a value of ϵ such that $t \notin \text{PO}(r; \mathcal{M}_p)$. Indeed, for t to be in $\text{PO}(r; \mathcal{M}_p)$, there needs to be a vector of weights $W = (w_1, w_2) \in \mathcal{W}$ such that $M_p^W(t) < M_p^W(t_1)$ and $M_p^W(t) < M_p^W(t_2)$. When $p \neq 0$, this means:

$$((1-\epsilon)^p w_1 + (1-\epsilon)^p w_2)^{1/p} < w_1^{1/p},$$
$$((1-\epsilon)^p w_1 + (1-\epsilon)^p w_2)^{1/p} < w_2^{1/p},$$

which reduces to $w_2 \frac{(1-\epsilon)^p}{1-(1-\epsilon)^p} < w_1 < w_2 \frac{1-(1-\epsilon)^p}{(1-\epsilon)^p}$. But this condition can never hold if $\frac{(1-\epsilon)^p}{1-(1-\epsilon)^p} \geq \frac{1-(1-\epsilon)^p}{(1-\epsilon)^p}$, which happens for $\epsilon \leq 1 - \frac{1}{2^{1/p}}$.

When $p = 0$, the condition becomes $(1-\epsilon)^{w_1} + (1-\epsilon)^{w_2} < 0$, which is impossible. □

Yet, a containment relationship holds for PO for increasing values of p.

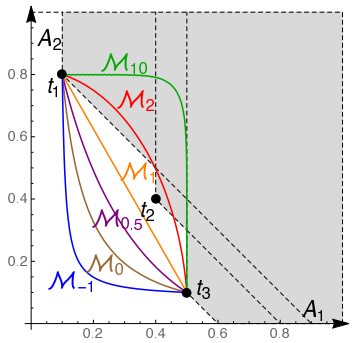


Fig. 26. Tuples from Example A.4. $\mathcal{M}_1^C$-dominance regions, with $C = \{w_1 \geq w_2\}$, are in gray. Tuple t_2 is potentially optimal for the $\mathcal{M}_p^C$ families with $p \geq 2$ but not with $p \leq 1$. The colored curves connecting t_1 and t_3 are the loci of points obtained as convex combinations of t_1 and t_3 for various values of p.

Theorem A.3. *Let $p' > p$. Then, for every r we have:*

$$\text{PO}(r; \mathcal{M}_p) \subseteq \text{PO}(r; \mathcal{M}_{p'}). \tag{34}$$

Proof. Let t be a tuple in $\text{PO}(r; \mathcal{M}_p)$. Then t minimizes an M_p^W function for some weight vector $W = (w_1, \ldots, w_d)$. Consider the region S_p (which includes only t among the tuples in r) defined by the inequalities $\sum_{i=1}^d w_i x_i^p \leq \sum_{i=1}^d w_i t[A_i]^p$ and $x_i \geq 0$ (for $i \in \{1, \ldots, d\}$). The region $S_{p'}$ defined by $\sum_{i=1}^d w'_i x_i^{p'} \leq \sum_{i=1}^d w'_i t[A_i]^{p'}$ and $x_i \geq 0$ (for $i \in \{1, \ldots, d\}$), with $W' = (w'_1, \ldots, w'_d)$ chosen so that the boundary of $S_{p'}$ is tangent in t to the boundary of S_p, is strictly enclosed in S_p. This implies that $S_{p'} \cap r = \{t\}$, thus $M_{p'}^{W'}(t) < M_{p'}^{W'}(s)$ holds for all $s \in r, s \neq t$. Then, $t \in \text{PO}(r; \mathcal{M}_{p'})$. □

Now we analyze how the presence of constraints can influence the above results, which were stated for the case in which no constraints on the weights are applied. For the sake of clarity, let us denote by $\mathcal{M}_p^C$ the subset of $\mathcal{M}_p$ functions that satisfy the constraints C.

The constrained counterpart of Theorem A.1 can be stated as $\text{ND}(r; \mathcal{M}_p^C) \subseteq \text{SKY}(r)$, which trivially follows from Proposition 3.11. Clearly, Theorem A.2 continues to hold also for constrained weighted $\mathcal{M}_p$ means, since $\text{PO}(r; \mathcal{M}_p^C) \subseteq \text{PO}(r; \mathcal{M}_p)$, by Proposition 3.14.

As for Theorem A.3, we start by analyzing, in the next example, the effect of p on the convex combinations that ultimately determine potential optimality of a tuple.

Example A.4. Consider constraint $C = \{w_1 \geq w_2\}$, tuples $t_1 = \langle 0.1, 0.8 \rangle$, $t_2 = \langle 0.4, 0.4 \rangle$, $t_3 = \langle 0.5, 0.1 \rangle$, and instance $r = \{t_1, t_2, t_3\}$, shown in Figure 26, where the gray area represents the union of the $\mathcal{M}_1^C$-dominance regions for all the tuples in r. The orange line connecting t_1 and t_3 represents the set of points that convexly combine t_1 and t_3 when the considered scoring functions are in $\mathcal{M}_1$, i.e., those points expressible as $\langle \alpha t_1[A_1] + (1-\alpha) t_3[A_1], \alpha t_1[A_2] + (1-\alpha) t_3[A_2] \rangle$ with $\alpha \in [0, 1]$. Therefore, such points are simply determined as the segment connecting t_1 and t_3. If any point on this segment $\mathcal{M}_1^C$-dominates t_2 (as is apparent from the figure), then t_2 cannot be in $\text{PO}(r; \mathcal{M}_1^C)$.

Let us now consider the $\mathcal{M}_2^C$ family. The red arc connecting t_1 and t_3 represents the points t that convexly combine t_1 and t_3 via functions in $\mathcal{M}_2$, i.e., those points expressible as $\langle \sqrt{\alpha t_1[A_1]^2 + (1-\alpha) t_3[A_1]^2}, \sqrt{\alpha t_1[A_2]^2 + (1-\alpha) t_3[A_2]^2} \rangle$ with $\alpha \in [0, 1]$. Such a shape is obtained by imposing $f(t) = f(t_1) = f(t_3)$, where f is a function in $\mathcal{M}_2^C$, and thus corresponds to an

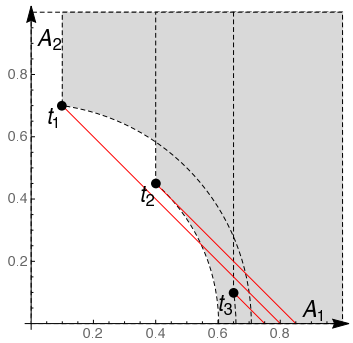


Fig. 27. Tuples from Example A.5. $\mathcal{M}_2^C$-dominance regions are in gray, with $d = 2$ and $C = \{w_1 \geq w_2\}$; red lines indicate the bottom-left part of $\mathcal{M}_1^C$-dominance regions.

ellipse centered in $\langle 0, 0\rangle$ and passing through t_1 and t_3. In this case, no point on the red arc $\mathcal{M}_2^C$-dominates t_2, and therefore, for the particular choice of weights determining the scoring function represented by the red arc, t_2 is the optimal tuple in r, and thus $t_2 \in \text{PO}(r; \mathcal{M}_2^C)$. Note that, since $d = 2$, by Proposition 4.8, we could have reached the same conclusion by simply checking that no point on the red arc dominates (instead of $\mathcal{M}_2^C$-dominates) t_2.

We can observe that t_2 continues to be potentially optimal for larger values of p, as shown in the figure for $p = 10$ (green arc), i.e., $t_2 \in \text{PO}(r; \mathcal{M}_p^C)$ for $p \geq 2$.

Based on similar considerations, we observe that, with values of p smaller than 1, t_2 can never be potentially optimal, as shown for $p = 0.5$ (purple arc), $p = 0$ (brown arc), and $p = -1$ (blue arc), i.e., $t_2 \notin \text{PO}(r; \mathcal{M}_p^C)$ for $p \leq 1$.

Example A.4 seems to suggest that Theorem A.3 also holds for constrained weighted power means. However, the following example shows that the inclusion holds neither for ND nor for PO when varying p.

Example A.5. Consider constraint $C = \{w_1 \geq w_2\}$, tuples $t_1 = \langle 0.1, 0.7\rangle$, $t_2 = \langle 0.4, 0.45\rangle$, $t_3 = \langle 0.65, 0.1\rangle$, and instance $r = \{t_1, t_2, t_3\}$, shown in Figure 27. The constrained $\mathcal{M}_2^C$-dominance regions of t_1, t_2 and t_3 are shown in gray, with dashed borders. The bottom-left part of such regions is an arc of a circle centered in $\langle 0, 0\rangle$ connecting the horizontal axis with the tuple. Such a shape derives from the definition of $\mathcal{F}$-dominance region when Equation (30) (for $p = 2$) is plugged in Equation (11) and the limit case $w_1 = w_2$ is considered, i.e., $\sqrt{0.5 \cdot t[A_1]^2 + 0.5 \cdot t[A_2]^2} \leq \sqrt{0.5 \cdot s[A_1]^2 + 0.5 \cdot s[A_2]^2}$, where t is the considered tuple and s is a generic tuple $\mathcal{M}_2^C$-dominated by t. Similarly, when the family of functions is $\mathcal{M}_1^C$, the bottom-left part of the $\mathcal{M}_1^C$-dominance regions is a segment whose slope is $-45°$ (shown in red in the figure).

Clearly, we have $\text{ND}(r; \mathcal{M}_2^C) = \{t_1, t_2\} \neq \text{ND}(r; \mathcal{M}_1^C) = \{t_1, t_3\}$. This shows that, generally, there is no inclusion between ND for constrained $\mathcal{M}_p^C$ families with increasing values of p. Indeed, there is no such inclusion between the corresponding $\mathcal{M}_p^C$-dominance regions of a tuple. Consider, e.g., tuple t_1: its $\mathcal{M}_1^C$-dominance region neither includes nor is included in the $\mathcal{M}_2^C$-dominance region, i.e., $DR(t_1, \mathcal{M}_2^C) \not\subseteq DR(t_1, \mathcal{M}_1^C)$ and $DR(t_1, \mathcal{M}_1^C) \not\subseteq DR(t_1, \mathcal{M}_2^C)$. Ditto for PO, since in this example $\text{ND}(r; \mathcal{M}_p^C) = \text{PO}(r; \mathcal{M}_p^C)$, with $p = 1, 2$, since there are only 2 tuples in ND and thus no convex combination can be formed to $\mathcal{M}_p^C$-dominate one such tuple.

B NON-TUPLE-DISTINGUISHING SETS OF FUNCTIONS

For the sake of completeness, in this appendix we analyze the case of non-tuple-distinguishing (NTD) sets of functions, including the case of finite sets (the focus on infinite sets was covered in Section 4.3).

Example 3.5 showed that $\mathcal{F}$-dominance between tuples may not be preserved when considering a larger set of functions. Note, however, that $\mathcal{F}$-dominance may be lost also when reducing $\mathcal{F}$, if the smaller set becomes NTD.

Example B.1. Consider tuples $t = \langle 0, 0\rangle$ and $s = \langle 0, 1\rangle$. Clearly, $t \prec_{\mathsf{MF}} s$, since the condition in Definition 3.3 holds. Consider now the monotone scoring function $f(x, y) = \min\{x, y\}$ and the set $\mathcal{F} = \{f\}$, which is NTD. Although the first condition of Definition 4.10 holds, the second one does not, since $f(t) = f(s) = 0$; therefore, $t \prec_{\mathcal{F}} s$ does not hold.

A direct consequence of the notion of tuple-distinguishability is stated in Corollary B.2 below.

Corollary B.2. *Let $\mathcal{F}$ be a* NTD *set of scoring functions. Then no subset of $\mathcal{F}$ is tuple-distinguishing.*

We observe that, when $r \neq \emptyset$, $\textsc{Sky}(r)$ and $\textsc{nd}(r; \mathcal{F})$ are never empty whereas $\textsc{po}(r; \mathcal{F})$ may only be empty if $\mathcal{F}$ is NTD.

Example B.3. Consider tuples $t = \langle 0, 1\rangle$ and $s = \langle 1, 0\rangle$, instance $r = \{t, s\}$, function $f(x, y) = x + y$ and set $\mathcal{F} = \{f\}$. Then $\textsc{po}(r; \mathcal{F}) = \emptyset$, since $f(t) = f(s) = 1$. Indeed, $\mathcal{F}$ is NTD.

We observe that, if $\mathcal{F} = \{f\}$, then both $\textsc{nd}(r; \mathcal{F})$ and $\textsc{po}(r; \mathcal{F})$ are a sort of "top-1" operator with respect to scoring function f, and return the best tuple according to f in r, if it exists. However, if there are ties, $\textsc{nd}(r; \mathcal{F})$ returns *all* the tied top-1 tuples in r, whereas $\textsc{po}(r; \mathcal{F}) = \emptyset$.

Besides the cases covered by Theorem 4.12, non-tuple-distinguishability may also arise if one considers a *finite* set of functions.

Proposition B.4. *Let $\mathcal{F}$ be a set of monotone scoring functions. If $|\mathcal{F}| < d$ and each function in $\mathcal{F}$ is continuous, then $\mathcal{F}$ is* NTD.

Proof. To prove the claim, let us represent the $m < d$ continuous functions in $\mathcal{F}$ by a single continuous m-valued function f from $[0, 1]^d$ to $\mathbb{R}^m$. Set $\mathcal{F}$ is tuple-distinguishing iff f is injective. We show that f cannot be both continuous and injective. Recall that a subset D of $\mathbb{R}^d$ is called open if, given any point x in D, there exists a real number $\epsilon > 0$ such that, given any point y in $\mathbb{R}^d$ whose Euclidean distance from x is smaller than ϵ, y also belongs to D. Indeed, $[0, 1]^d$ contains an open subset D of $\mathbb{R}^d$ (for instance, $(0, 1)^d$). The restriction of f from D to $\mathbb{R}^m$ remains continuous and injective. Let us now extend f to a new function F from D to $\mathbb{R}^d = \mathbb{R}^m \times \mathbb{R}^{d-m}$ by adding a point c in $\mathbb{R}^{d-m}$, so that the image $F(p)$ of a point p in D through F becomes $F(p) = (f(p), c)$ (i.e., the m values in the image $f(p)$ of p through f plus $d - m$ fixed values from point c). F remains continuous and injective. The domain invariance theorem [7] states that the image of a non-empty open subset D of $\mathbb{R}^d$ through a continuous and injective function from D to $\mathbb{R}^d$ is an open set, while $F(D) = f(D) \times \{c\}$ is evidently not open. □

Example B.5. Consider $d = 3$, functions $f_1(x, y, z) = x + y + z$ and $f_2(x, y, z) = x + y + 2z$, and the set $\mathcal{F} = \{f_1, f_2\}$. Set $\mathcal{F}$ is NTD by Proposition B.4, since $|\mathcal{F}| = 2 < 3 = d$, and all the functions in $\mathcal{F}$ are continuous. Indeed, given tuples $t = \langle 0.2, 0.5, 0.5\rangle$ and $u = \langle 0.4, 0.3, 0.5\rangle$, we have $f_1(t) = f_1(u) = 1.2$ and $f_2(t) = f_2(u) = 1.7$.

When $|\mathcal{F}| < d$, the requirement in Proposition B.4 that the functions are continuous is essential for the result to hold. Indeed, if one drops the hypothesis of continuity, then a *single* monotone function would be sufficient to achieve tuple-distinguishability for *any* value of d.

Example B.6. Let $d = 2$ and, for any tuple t, consider the scoring function f obtained by interleaving the bits of the binary representations of $t[A_1]$ and $t[A_2]$, which leads to the well-known Z-order [21]. For instance, when $t = \langle 0.75, 0.25 \rangle$, the binary representations are, respectively, 0.11_2 and 0.01_2, and interleaving the bits yields 0.1011_2, thus $f(t) = 0.6875$. Clearly, f is not continuous, yet it is monotone and injective (thus tuple-distinguishing). The same procedure can be applied to any value of d.

The condition $|\mathcal{F}| \geq d$ is not sufficient to avoid NTD sets. For instance, let $d = 2$ and consider the set $\mathcal{F} = \{f_1, f_2\}$, with $f_1(x, y) = x + y$ and $f_2(x, y) = x \cdot y$. The set $\mathcal{F}$ is NTD since there are tuples, such as $t = \langle 0.2, 0.5 \rangle$ and $u = \langle 0.5, 0.2 \rangle$, having the same score for both f_1 and f_2 ($f_1(t) = f_1(u) = 0.7$ and $f_2(t) = f_2(u) = 0.1$).

Verifying that a finite set of monotone (and continuous) scoring functions $\mathcal{F}$ is tuple-distinguishing appears to be a challenging problem to solve, in the general case, as it amounts to checking injectivity of a multi-dimensional mapping, which is known to be hard [1].

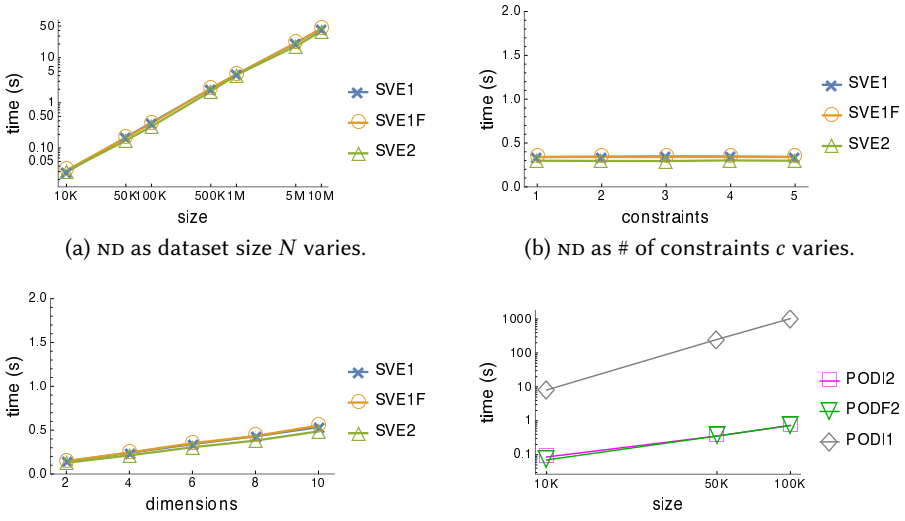
C ADDITIONAL EXPERIMENTS

In this appendix we discuss additional experiments regarding aspects that were not considered in full detail in the main body of the paper. In particular, we show performance results for computing ND and PO in a number of new scenarios and datasets. New datasets include correlated data, both synthetic (COR) and real (HOU), as well as clustered data, both synthetic (GAU) and real (EMP). Moreover, we study the effect of adopting different kinds of constraints on performance for all the main datasets analyzed in the main body of the paper. Finally, we assess how performance is affected by non-linearity in the scoring function by observing the behavior as the p parameter of M_p^W means changes.

Performances of SVE1F, SVE1 and SVE2 for computing ND are reported in Figure 28a for correlated data (COR) as dataset size N varies, with little observable differences in the execution times among the various algorithms. Indeed, all algorithms spend most of the time for sorting the dataset, while the final result is found very quickly as soon as the dataset is sorted. With default parameters we have sub-second execution times. A closer analysis as the number of constraints c (Figure 28b) or as the number of dimensions d (Figure 28c) varies reveals that SVE2 is around 10% faster than the other alternatives, confirming the observation made in Section 6 that SVE2 usually prevails in less challenging datasets. Figure 28d compares three algorithms for computing PO, namely PODI2, PODF2 and PODI1, on small-sized versions of COR (up to $100K$), and shows that, while we always obtain sub-second times with PODI2 and PODF2 with these parameter configurations, the execution times are already unacceptably high for PODI1, thus confirming what was observed in Section 6.

Figure 29a shows performance for computing ND on HOU as the number of constraints c varies. The trends are similar to those obtained with synthetic data, with SVE1 and SVE1F nearly tied for all values of c at around 1s and SVE2 performing around 20% better. The analysis for PO, reported in Figure 29b, shows that the difference in performance between PODI2 and PODF2 is less than 2% for all values of c.

A similar analysis is conducted on the EMP real dataset as the number of constraints c varies (Figure 30). While the dataset proves nearly twice more challenging than HOU (being more than twice as large), with worst times around 2s for most algorithms, the relative differences between the different algorithms are unchanged: SVE1 and SVE1F are essentially tied, whereas SVE2 is around 15% faster; PODI2 and PODF2 are less than 2% apart for all values of c.



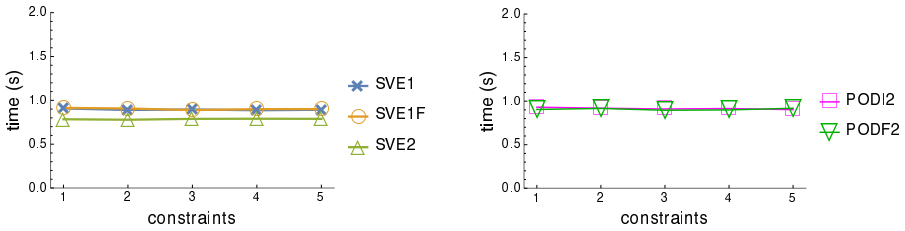
(a) ND as dataset size N varies.

(b) ND as # of constraints c varies.

(c) ND as # of dimensions d varies.

(d) PO on smaller-scale datasets as size N varies.

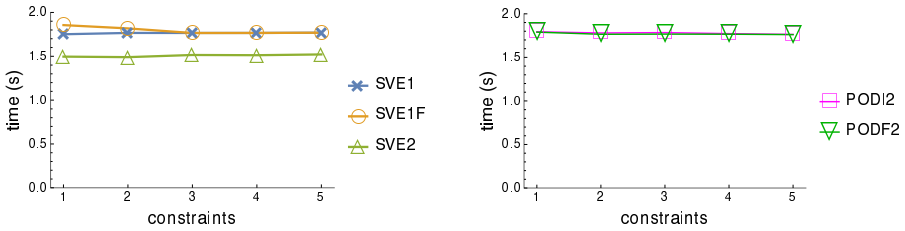
Fig. 28. Performance for computing ND and PO on correlated datasets (COR).



(a) Computing ND.

(b) Computing PO.

Fig. 29. Performance for computing ND and PO on the real dataset HOU as the number of constraints c varies.

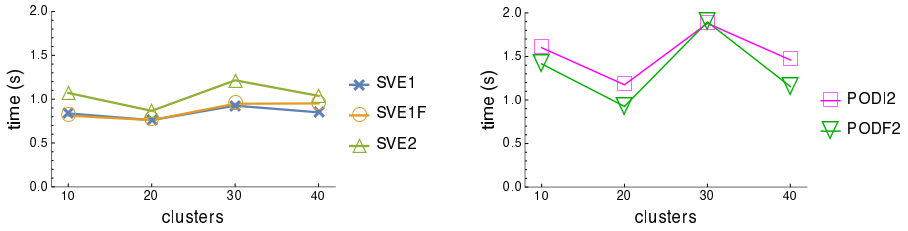


(a) Computing ND.

(b) Computing PO.

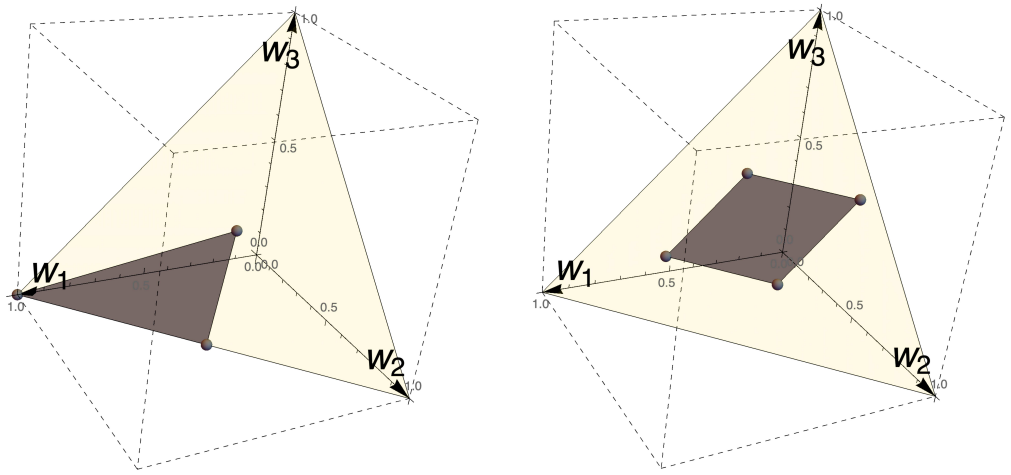
Fig. 30. Performance for computing ND and PO on the real dataset EMP as the number of constraints c varies.

In order to analyze clustered data, we have generated synthetic datasets, referred to here as GAU, with a varying number of Gaussian clusters (10, 20, 30, and 40) and default parameter values otherwise. Our experiments on performance for computing ND and PO on GAU are shown in Figure 31. Execution times vary as the number of clusters varies, with a close correspondence to the number



(a) Computing ND. (b) Computing PO.

Fig. 31. Performance for computing ND and PO on the clustered dataset GAU as the number of clusters varies.



(a) $C = \{w_1 \geq w_2, w_2 \geq w_3\}$ (weak rankings).

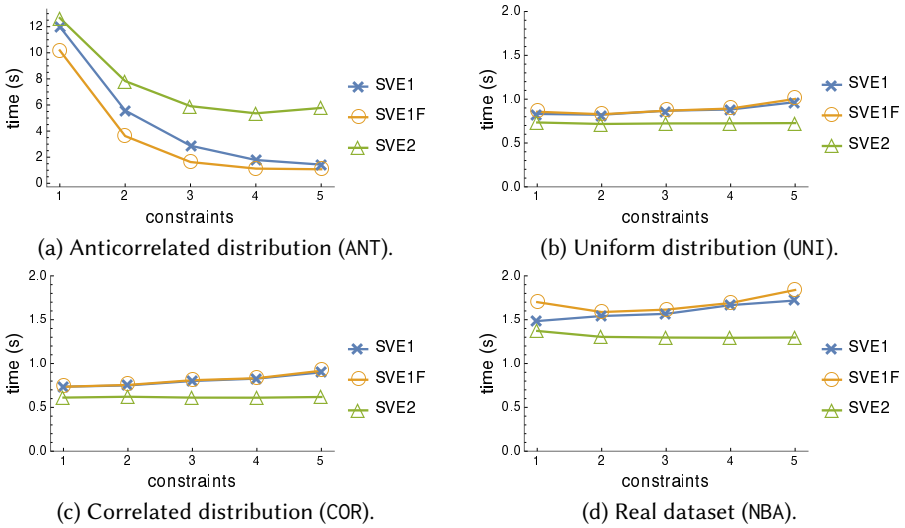
(b) $C = \{w_2, w_3 \in [\bar{w}(1-\epsilon), \bar{w}(1+\epsilon)]\}$ (interval constraints), with $\bar{w} = \frac{1}{d}$ and ϵ chosen so as to equal the area of the gray polygon in Figure 32a.

Fig. 32. Polytopes $\mathcal{W}(C)$ (in gray) on the 2-simplex (in yellow) from Example C.1, with $d = 3$. In both examples, the area of the polytope is 1/6 of the area of the simplex.

of tuples in the result set. Indeed, in the tested instances, the cardinalities of both ND and PO are highest with 30 clusters (with, respectively, 786 and 58 tuples), which in turn correspond to the highest execution times for all algorithms. Similarly, the lowest cardinalities occur with 20 clusters (with, respectively, 382 and 44 tuples), corresponding to the fastest executions. The spread around the average execution time for each algorithm is within ±20% for ND and ±30% for PO.

Before introducing our experiments on interval constraints, we first illustrate their differences with respect to constraints in the form of weak rankings. To this end, we offer a geometric illustration for $d = 3$ in the next example.

Example C.1. Consider the 2-simplex in $\mathbb{R}^3$, shown in yellow in Figures 32a and 32b. Figure 32a also shows, in gray, the shape of the polytope (polygon, in this case) $\mathcal{W}(C)$ when the set of constraints is $C = \{w_1 \geq w_2, w_2 \geq w_3\}$, i.e., two weak rankings constraints. In this case, $\mathcal{W}(C)$ has vertices $(1, 0, 0)$, $(\frac{1}{2}, \frac{1}{2}, 0)$ and $(\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$, and area $\frac{1}{4\sqrt{3}}$, i.e., $\frac{1}{6}$ of the area of the simplex ($\frac{\sqrt{3}}{2}$).



(a) Anticorrelated distribution (ANT). (b) Uniform distribution (UNI).

(c) Correlated distribution (COR). (d) Real dataset (NBA).

Fig. 33. Performance for computing ND with interval constraints as their number varies (default parameter values).

Similarly, Figure 32b shows $\mathcal{W}(C)$ when the set of constraints C consists of two interval constraints:

$$\bar{w}(1-\epsilon) \leq w_2 \leq \bar{w}(1+\epsilon)$$
$$\bar{w}(1-\epsilon) \leq w_3 \leq \bar{w}(1+\epsilon),$$

where $\bar{w} = \frac{1}{d} = \frac{1}{3}$ and ϵ is chosen so that the area of $\mathcal{W}(C)$ is also $\frac{1}{6}$ of the area of the simplex, by analogy with the previous case. This is obtained by setting $\epsilon = \frac{\sqrt{3}}{4}$. The vertices of $\mathcal{W}(C)$ in this case are: $\langle \frac{1}{3}, \frac{1+\epsilon}{3}, \frac{1-\epsilon}{3} \rangle$, $\langle \frac{1+2\epsilon}{3}, \frac{1-\epsilon}{3}, \frac{1-\epsilon}{3} \rangle$, $\langle \frac{1}{3}, \frac{1-\epsilon}{3}, \frac{1+\epsilon}{3} \rangle$, and $\langle \frac{1-2\epsilon}{3}, \frac{1+\epsilon}{3}, \frac{1+\epsilon}{3} \rangle$.

Figure 33 shows performance of computing ND under interval constraints for the ANT (Figure 33a), UNI (Figure 33b), COR (Figure 33c), and NBA (Figure 33d) datasets as the number c of constraints varies (default parameter values otherwise for synthetic datasets). We set $\bar{w} = \frac{1}{d} = \frac{1}{6}$ and $\epsilon = 40\%$. The results show that SVE1F is the preferred choice for more challenging scenarios (ANT) and that, in such cases, times decrease as the number c of constraints grows, since pruning is very effective in these cases. For less challenging cases, SVE2 is the best choice, and pruning is not significant, so that times do not necessarily decrease as c grows.

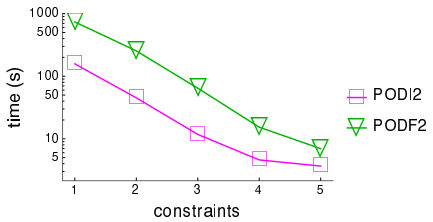
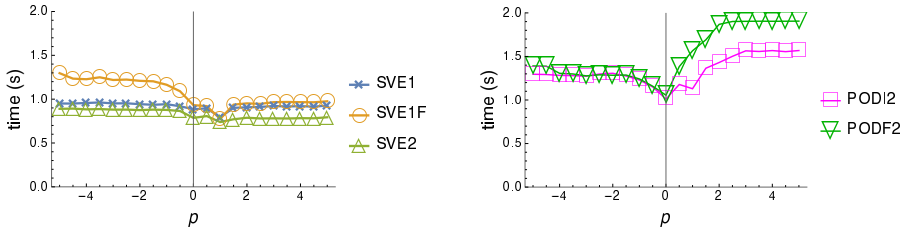


Fig. 34. Performance for computing PO on ANT with interval constraints as their number varies (default parameter values).



(a) Computing ND. (b) Computing PO.

Fig. 35. Performance for computing ND and PO on UNI as p varies (default parameter values).

Figure 34 shows performance of computing PO under interval constraints for the ANT dataset as c varies (default parameter values otherwise for synthetic datasets and, again, $\bar{w} = \frac{1}{d} = \frac{1}{6}$ and $\epsilon = 40\%$). The gains of PODI2 with respect to PODF2 are significant, and more so as c decreases. For instance, when $c = 1$, PODF2 requires 738s while PODI2 only requires 157s.

Figure 35 shows the effect of the p parameter of M_p^W means on execution times for computing ND and PO on the UNI dataset. Generally, computing ND reaches a minimum execution time when $p = 1$, while performance slightly deteriorates as p diverges from 1. This is likely due to the fact that the number of dominance and $\mathcal{F}$-dominance tests, for all algorithms, minimizes between 0 and 1 and increases for larger values of $|p|$; also, our implementation does not use power functions when $p = 1$, thus possibly saving time. A similar effect is observed for the computation of PO, only with a minimum execution time around $p = 0$. A possible explanation is that the number of LP tests performed by both algorithms drops considerably when moving from $p > 0$ to $p = 0$, while it increases as p grows and it remains more or less stable when $p < 0$.

D PROOFS

This appendix reports all the proofs of the claims included in the main body of the paper.

Proposition 3.11. *For any set $\mathcal{F}$ of monotone scoring functions, the following relationships hold:*

$$\text{PO}(r;\mathcal{F}) \subseteq \text{ND}(r;\mathcal{F}) \subseteq \text{SKY}(r). \tag{10}$$

Proof. To prove (10), take any tuple $t \in \text{PO}(r;\mathcal{F})$. According to Definition 3.8, there exists a scoring function $f \in \mathcal{F}$ such that $f(t)$ is lower than the score of any other tuple in r. Therefore, there cannot be any tuple in r that either dominates or $\mathcal{F}$-dominates t in the sense of Definitions 2.1 and 3.3, because for at least function f, t would achieve a better score. Therefore,

- t is not dominated, thus $t \in \text{SKY}(r)$ according to Definition 2.1, i.e., $\text{PO}(r;\mathcal{F}) \subseteq \text{SKY}(r)$.
- t is not $\mathcal{F}$-dominated, thus $t \in \text{ND}(r;\mathcal{F})$ according to Definition 3.7, i.e., $\text{PO}(r;\mathcal{F}) \subseteq \text{ND}(r;\mathcal{F})$.

To prove that $\text{ND}(r;\mathcal{F}) \subseteq \text{SKY}(r)$, we first observe that $\text{ND}(r;\mathcal{F}) \subseteq \text{ND}(r;\text{MF})$ follows from Equation (7), since $\mathcal{F} \subseteq \text{MF}$ entails $\prec_{\text{MF}} \subseteq \prec_{\mathcal{F}}$. By transitivity with Equation (9), we obtain $\text{ND}(r;\mathcal{F}) \subseteq \text{SKY}(r)$. □

Proposition 3.14. ND *and* PO *are monotone operators with respect to the set of scoring functions, i.e., for any two sets $\mathcal{F}_1$ and $\mathcal{F}_2$ of monotone scoring functions such that $\mathcal{F}_1 \subseteq \mathcal{F}_2$, the following relationships hold:*

$$\text{PO}(r;\mathcal{F}_1) \subseteq \text{PO}(r;\mathcal{F}_2), \tag{12}$$

$$\text{ND}(r;\mathcal{F}_1) \subseteq \text{ND}(r;\mathcal{F}_2). \tag{13}$$

Proof. Inequality (12) is a direct consequence of Definition 3.8. Inequality (13) follows from Definition 3.7 by observing that $\mathcal{F}_1 \subseteq \mathcal{F}_2$ entails $\prec_{\mathcal{F}_2} \subseteq \prec_{\mathcal{F}_1}$ (as stated in Corollary 3.4). □

Proposition 3.17. *Let $\mathcal{F} = \mathcal{F}_1 \cup \ldots \cup \mathcal{F}_m$, where each $\mathcal{F}_i$, $1 \leq i \leq m$, is a set of monotone scoring functions. Then:*

$$\text{ND}(r;\mathcal{F}) \supseteq \text{ND}(r;\mathcal{F}_1) \cup \ldots \cup \text{ND}(r;\mathcal{F}_m) \tag{15}$$

$$\text{PO}(r;\mathcal{F}) = \text{PO}(r;\mathcal{F}_1) \cup \ldots \cup \text{PO}(r;\mathcal{F}_m) \tag{16}$$

Proof. By Proposition 3.14, each disjunct in the right-hand side of Equation (15) is contained in the left-hand side, hence the claim.

Analogously, the right-hand side of Equation (16) is contained in its left-hand side. Additionally, let $t \in \text{PO}(r;\mathcal{F})$. Then, $\exists f \in \mathcal{F}$ such that $f(t) < f(s)\ \forall s \in r \setminus \{t\}$. Since $\mathcal{F} = \cup_{i=1}^{m} \mathcal{F}_i$, there exists $\mathcal{F}_i$ such that $f \in \mathcal{F}_i$. Hence $t \in \text{PO}(r;\mathcal{F}_i)$, thus proving the claim. □

Proposition 3.21. *Let r and r' be two instances over R with $r \subset r'$. For any set of monotone scoring functions $\mathcal{F}$, the following holds:*

- $\text{ND}(r';\mathcal{F}) \cap r \subseteq \text{ND}(r;\mathcal{F})$,
- $\text{PO}(r';\mathcal{F}) \cap r \subseteq \text{PO}(r;\mathcal{F})$.

Proof. Let $t \in r$. Since $t \in \text{ND}(r';\mathcal{F})$, there is no other tuple $t' \in r'$ that $\mathcal{F}$-dominates t. Since $r \subseteq r'$, it follows that no tuple in r $\mathcal{F}$-dominates t.

For the second part, since $t \in \text{PO}(r';\mathcal{F})$, there exists a scoring function $f \in \mathcal{F}$ such that, for all tuples $t' \in r'$, $f(t) < f(t')$ holds. Since $r \subseteq r'$, t is potentially optimal also in r. □

Proposition 3.22. *Let $r = r_1 \cup \ldots \cup r_m$, where r_i, $1 \leq i \leq m$, are relations over R, and $\mathcal{F}$ be a set of monotone scoring functions. Then:*

$$\text{ND}(r;\mathcal{F}) = \text{ND}(\text{ND}(r_1;\mathcal{F}) \cup \ldots \cup \text{ND}(r_m;\mathcal{F});\mathcal{F}), \tag{19}$$

$$\text{PO}(r;\mathcal{F}) = \text{PO}(\text{PO}(r_1;\mathcal{F}) \cup \ldots \cup \text{PO}(r_m;\mathcal{F});\mathcal{F}). \tag{20}$$

Proof. Let $r' = \text{ND}(r_1;\mathcal{F}) \cup \ldots \cup \text{ND}(r_m;\mathcal{F})$; clearly, $r' \subseteq r$. Every tuple $t \in \text{ND}(r;\mathcal{F})$ must also belong to $\text{ND}(r';\mathcal{F})$; indeed, t must belong to one of $r_1, \ldots, r_m$, say r_i, and therefore $t \in \text{ND}(r_i;\mathcal{F})$, since no tuple in r (and a fortiori in $r_i \subseteq r$) $\mathcal{F}$-dominates t. So, $t \in r'$ and then $t \in \text{ND}(r';\mathcal{F})$ since no tuple in r (and a fortiori in $r' \subseteq r$) $\mathcal{F}$-dominates t. For the other direction, consider a tuple $s \in \text{ND}(r';\mathcal{F})$. Then s is not $\mathcal{F}$-dominated by any tuple in r'. Note that s cannot be $\mathcal{F}$-dominated by any tuple in $r \setminus r'$ either, since every tuple in $r \setminus r'$ is $\mathcal{F}$-dominated by some tuple in r' and $\mathcal{F}$-dominance is transitive (Corollary 3.6), so $s \in \text{ND}(r;\mathcal{F})$.

Similarly, let $r' = \text{PO}(r_1;\mathcal{F}) \cup \ldots \cup \text{PO}(r_m;\mathcal{F})$; clearly, $r' \subseteq r$. If $t \in \text{PO}(r;\mathcal{F})$ then $\exists f \in \mathcal{F}$ such that $f(t) < f(t')$ for every other tuple $t' \in r$, and, a fortiori, also for every other tuple $t' \in r' \subseteq r$, so $t \in \text{PO}(r';\mathcal{F})$. For the other direction, consider a tuple $s \in \text{PO}(r';\mathcal{F})$. Then $\exists f \in \mathcal{F}$ such that $f(s) < f(s')$ for every other tuple $s' \in r'$. Note that $f(s) < f(s')$ also holds for every other tuple $s' \in r$, since every tuple in $r \setminus r'$ is worse than some tuple in r' according to f and, by transitivity, than s, so $s \in \text{ND}(r;\mathcal{F})$. □

Theorem 4.1 ($\mathcal{F}$-dominance test). *Let $\mathcal{F}$ be a set of MLW functions subject to a set $C = \{C_1, \ldots, C_c\}$ 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, $t \prec_{\mathcal{F}} s$ iff the following linear programming problem (LP) in the variables $W = (w_1, \ldots, w_d)$ has a*

non-negative solution:

$$\begin{array}{lll} \text{minimize} & \Lambda \cdot \sum_{i=1}^{d} w_i(g_i(s[A_i]) - g_i(t[A_i])) & \\ \text{subject to} & w_i \in [0,1] \qquad i \in \{1,\ldots,d\} & \\ & \sum_{i=1}^{d} w_i = 1 & \\ & \sum_{i=1}^{d} a_{ji} w_i \leq k_j \qquad j \in \{1,\ldots,c\}. & \end{array} \tag{22}$$

Proof. The following expression

$$\sum_{i=1}^{d} w_i g_i(s[A_i]) - \sum_{i=1}^{d} w_i g_i(t[A_i]) \tag{35}$$

reduces to the objective of System (22). The result immediately follows from the definition of $\mathcal{F}$-dominance, since, for any function $f^W \in \mathcal{F}$, the difference $f^W(s) - f^W(t)$ is

$$h(\sum_{i=1}^{d} w_i g_i(s[A_i])) - h(\sum_{i=1}^{d} w_i g_i(t[A_i])), \tag{36}$$

and (36) has the same sign as (35). □

Theorem 4.2 ($\mathcal{F}$-dominance region). *Let $\mathcal{F}$ be a set of MLW functions subject to a set $C = \{C_1,\ldots,C_c\}$ 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 $\mathcal{W}(C)$. The dominance region $DR(t;\mathcal{F})$ of a tuple t under $\mathcal{F}$ is the locus of points s in $[0,1]^d$ defined by the q inequalities:*

$$\Lambda \cdot \sum_{i=1}^{d} w_i^{(\ell)} g_i(s[A_i]) \geq \Lambda \cdot \sum_{i=1}^{d} w_i^{(\ell)} g_i(t[A_i]), \quad \ell \in \{1,\ldots,q\}. \tag{23}$$

Proof. We assume $\Lambda = 1$, the case $\Lambda = -1$ being analogous. Since $\mathcal{W}(C)$ is convex, any $W \in \mathcal{W}(C)$ can be written as a convex combination of the vertices, i.e., $W = \sum_{\ell=1}^{q} b_\ell W^{(\ell)}$, with $b_\ell \geq 0$, $\ell \in \{1,\ldots,q\}$, and $\sum_{\ell=1}^{q} b_\ell = 1$. For $\ell \in \{1,\ldots,q\}$, let us multiply both members of the ℓ-th Inequality (23) by b_ℓ and sum member-wise all the resulting q inequalities. We obtain:

$$\sum_{\ell=1}^{q} b_\ell \sum_{i=1}^{d} w_i^{(\ell)} g_i(s[A_i]) \geq \sum_{\ell=1}^{q} b_\ell \sum_{i=1}^{d} w_i^{(\ell)} g_i(t[A_i]),$$

which holds if and only if the following holds

$$\sum_{i=1}^{d} \sum_{\ell=1}^{q} b_\ell w_i^{(\ell)} g_i(s[A_i]) \geq \sum_{i=1}^{d} \sum_{\ell=1}^{q} b_\ell w_i^{(\ell)} g_i(t[A_i]), \text{ i.e.,}$$

$$\sum_{i=1}^{d} w_i g_i(s[A_i]) \geq \sum_{i=1}^{d} w_i g_i(t[A_i]). \tag{37}$$

Now, since h is monotone, Inequality (37) holds if and only if the following holds:

$$h\left(\sum_{i=1}^{d} w_i g_i(s[A_i])\right) \geq h\left(\sum_{i=1}^{d} w_i g_i(t[A_i])\right),$$

i.e., if and only if $t \prec_{\mathcal{F}} s$ holds, which is what we wanted to prove. □

Proposition 4.4. *Let $\mathcal{F}$ be a set of MLW functions subject to a set C of linear constraints on weights and such that h is strictly monotone. Then, for any instance r, we have* $\text{PO}(r;\mathcal{F}) = \text{PO}(\text{ND}(r;\mathcal{F});\mathcal{F})$.

Proof. If $t \in \text{PO}(r;\mathcal{F})$, there exists $f \in \mathcal{F}$ such that $f(t) < f(s)$, $\forall s \in r \setminus \{t\}$, and, a fortiori, also $\forall s \in \text{ND}(r;\mathcal{F}) \setminus \{t\}$, since $\text{ND}(r;\mathcal{F}) \subseteq r$. Since $t \in \text{ND}(r;\mathcal{F})$ by Proposition 3.11, then $t \in \text{PO}(\text{ND}(r;\mathcal{F});\mathcal{F})$.

If $t \in \text{PO}(\text{ND}(r;\mathcal{F});\mathcal{F})$, $\exists f \in \mathcal{F}$ such that $f(t) < f(s)$, $\forall s \in \text{ND}(r;\mathcal{F}) \setminus \{t\}$. Now, if $t \notin \text{PO}(r;\mathcal{F})$ then there would exist a tuple $t' \in r \setminus \text{ND}(r;\mathcal{F})$, such that $f(t') \leq f(t)$. Since the case $f(t') < f(t)$ cannot occur (otherwise t' would not be $\mathcal{F}$-dominated, thus a contradiction), we have $f(t') = f(t)$. Note that this implies that t' is $\mathcal{F}$-dominated by t and by no other tuple in $\text{ND}(r;\mathcal{F})$ (since $f(t') = f(t) < f(s)$, $\forall s \in \text{ND}(r;\mathcal{F}) \setminus \{t\}$).

Let W be the weight vector of function f. From the assumption of continuity it follows that for any arbitrarily small value of $\varepsilon > 0$, there exists a neighborhood of W in $\mathcal{W}(C)$ such that, for any weight vector W' in the neighborhood, we have $|f^{W'}(t) - f(t)| < \varepsilon$. Then, by choosing ε small enough, t will remain the best tuple in $\text{ND}(r;\mathcal{F})$, i.e., $f^{W'}(t) < f^{W'}(s)$, $\forall s \in \text{ND}(r;\mathcal{F}) \setminus \{t\}$. Finally, since t $\mathcal{F}$-dominates t' and h is strictly monotone, there exists W' in the neighborhood of W such that $f^{W'}(t) < f^{W'}(t')$, which completes the proof. □

Theorem 4.5 (Primal PO test). *Let $\mathcal{F}$ be a set of MLW functions subject to a set $C = \{C_1, \ldots, C_c\}$ of linear constraints on weights, where $C_j = \sum_{i=1}^{d} a_{ji} w_i \leq k_j$ (for $j \in \{1, \ldots, c\}$), and such that h is strictly monotone. Let $\text{ND}(r;\mathcal{F}) = \{t_1, t_2, \ldots, t_\sigma, t\}$. Then, $t \in \text{PO}(r;\mathcal{F})$ iff the following linear programming problem (LP) in the variables $W = (w_1, \ldots, w_d)$ and ϕ has a strictly positive optimal solution:*

$$
\begin{array}{lll}
\text{maximize} & \phi & \\
\text{subject to} & \Lambda \cdot \sum_{i=1}^{d} w_i (g_i(t[A_i]) - g_i(t_j[A_i])) + \phi \leq 0 & j \in \{1, \ldots, \sigma\} \\
& \sum_{i=1}^{d} a_{ji} w_i \leq k_j & j \in \{1, \ldots, c\} \\
& w_i \in [0, 1] & i \in \{1, \ldots, d\} \\
& \sum_{i=1}^{d} w_i = 1. &
\end{array}
\tag{28}
$$

Proof. We assume $\Lambda = 1$, the case $\Lambda = -1$ being analogous. If the above LP problem has optimal solution $\phi^* > 0$, with $(w_1^*, \ldots, w_d^*)$ being the corresponding weight vector, from the first σ constraints we derive $\sum_{i=1}^{d} w_i^* g_i(t[A_i]) < \sum_{i=1}^{d} w_i^* g_i(t_j[A_i])$, $j \in \{1, \ldots, \sigma\}$, i.e. $t \in \text{PO}(r;\mathcal{F})$. On the other hand, $\phi^* \leq 0$ implies that $t \notin \text{PO}(r;\mathcal{F})$. Notice that the problem is always feasible, since we assumed that C is not contradictory, i.e., $\mathcal{W}(C) \neq \emptyset$, and thus the last $c + d + 1$ constraints are satisfiable, and ϕ is not constrained (i.e., $\phi \in \mathbb{R}$), thus also the first σ constraints are satisfiable. □

Theorem 4.7 (Dual PO test). *Let $\mathcal{F}$ be a set of MLW functions subject to a set C of linear constraints on weights and such that h is strictly monotone. Let $W^{(1)}, \ldots, W^{(q)}$ be the vertices of $\mathcal{W}(C)$ and let $\text{ND}(r;\mathcal{F}) = \{t_1, t_2, \ldots, t_\sigma, t\}$. Then, $t \in \text{PO}(r;\mathcal{F})$ iff there is no convex combination s of $t_1, \ldots, t_\sigma$ such that $s \prec_{\mathcal{F}} t$, i.e., iff the following linear system in the variables $\alpha = (\alpha_1, \ldots, \alpha_\sigma)$ is unsatisfiable:*

$$
\begin{array}{lll}
\Lambda \cdot \sum_{i=1}^{d} w_i^{(\ell)} (\sum_{j=1}^{\sigma} \alpha_j g_i(t_j[A_i])) \leq \Lambda \cdot \sum_{i=1}^{d} w_i^{(\ell)} g_i(t[A_i]) & \ell \in \{1, \ldots, q\} \\
\alpha_j \in [0, 1] & j \in \{1, \ldots, \sigma\} \\
\sum_{j=1}^{\sigma} \alpha_j = 1. &
\end{array}
\tag{29}
$$

Proof. We assume $\Lambda = 1$, the case $\Lambda = -1$ being analogous.

Only-if part. Let $t \in \text{PO}(r;\mathcal{F})$ and assume that the above system is satisfiable with $\alpha^* = (\alpha_1^*, \ldots, \alpha_\sigma^*)$. Since $t \in \text{PO}(r;\mathcal{F})$, there must exist $W \in \mathcal{W}(C)$ such that t's score is better than those of $t_1, \ldots, t_\sigma$:

$$h\left(\sum_{i=1}^{d} w_i g_i(t[A_i])\right) < h\left(\sum_{i=1}^{d} w_i g_i(t_j[A_i])\right), \quad j \in \{1, \ldots, \sigma\}. \tag{38}$$

Since h is strictly monotone, (38) holds if and only if the following holds:

$$\sum_{i=1}^{d} w_i g_i(t[A_i]) < \sum_{i=1}^{d} w_i g_i(t_j[A_i]), \;\; j \in \{1, \ldots, \sigma\}. \tag{39}$$

Let us multiply both members of the j-th Inequality (39) by α_j^*, for $1 \le j \le \sigma$, and sum member-wise all the resulting σ inequalities. We obtain:

$$\begin{aligned} \sum_{j=1}^{\sigma} \alpha_j^* \sum_{i=1}^{d} w_i g_i(t[A_i]) < \sum_{j=1}^{\sigma} \alpha_j^* \sum_{i=1}^{d} w_i g_i(t_j[A_i]), \text{ i.e.,} \\ \sum_{i=1}^{d} w_i g_i(t[A_i]) < \sum_{i=1}^{d} w_i \sum_{j=1}^{\sigma} \alpha_j^* g_i(t_j[A_i]). \end{aligned} \tag{40}$$

Since α^* satisfies system (29), we also have:

$$\sum_{i=1}^{d} w_i^{(\ell)} \sum_{j=1}^{\sigma} \alpha_j^* g_i(t_j[A_i]) \le \sum_{i=1}^{d} w_i^{(\ell)} g_i(t[A_i]), \quad \ell \in \{1, \ldots, q\} \tag{41}$$

As in the proof of Theorem 4.2, we exploit the fact that any $W \in \mathcal{W}(C)$ can be written as $W = \sum_{\ell=1}^{q} b_\ell W^{(\ell)}$. For $\ell \in \{1, \ldots, q\}$, let us multiply both members of the ℓ-th Inequality (41) by b_ℓ and sum member-wise all the resulting q inequalities. We obtain:

$$\sum_{\ell=1}^{q} b_\ell \sum_{i=1}^{d} w_i^{(\ell)} \sum_{j=1}^{\sigma} \alpha_j^* g_i(t_j[A_i]) \le \sum_{\ell=1}^{q} b_\ell \sum_{i=1}^{d} w_i^{(\ell)} g_i(t[A_i]),$$

which holds if and only if the following holds

$$\sum_{i=1}^{d} \sum_{\ell=1}^{q} b_\ell w_i^{(\ell)} \sum_{j=1}^{\sigma} \alpha_j^* g_i(t_j[A_i]) \le \sum_{i=1}^{d} \sum_{\ell=1}^{q} b_\ell w_i^{(\ell)} g_i(t[A_i]),$$

which in turn becomes:

$$\sum_{i=1}^{d} w_i \sum_{j=1}^{\sigma} \alpha_j^* g_i(t_j[A_i]) \le \sum_{i=1}^{d} w_i g_i(t[A_i]). \tag{42}$$

Inequalities (42) and (40) are in contradiction, hence if $t \in \text{PO}(r;\mathcal{F})$ then system (29) must be unsatisfiable.

If part. The proof that the condition in the theorem is also sufficient is based on the relation existing between a solution to an LP problem and its *dual* version.

Consider LP problem (28) in the variables $W = (w_1, \ldots, w_d)$ and ϕ, and recall that $t \in \text{PO}(r;\mathcal{F})$ iff the LP problem has optimal solution $\phi^* > 0$.

In order to simplify the notation, let us introduce the shorthands $x_i = g_i(t[A_i]), i \in \{1, \ldots, d\}$, and $y_{ji} = g_i(t_j[A_i]), i \in \{1, \ldots, d\}, j \in \{1, \ldots, \sigma\}$. Then, the first σ constraints in (28) become $\sum_{i=1}^{d} w_i(x_i - y_{ji}) + \phi \le 0, j \in \{1, \ldots, \sigma\}$.

Let the set of constraints C be $\{C_1, \ldots, C_c\}$, where $C_j = \sum_{i=1}^{d} a_{ji} w_i \leq k_j$ (for $j \in \{1, \ldots, c\}$). The dual version of LP problem (28) can then be written as follows (see, e.g., [22]):

$$\begin{array}{ll} \text{minimize} & \sum_{j=1}^{c} k_j \beta_j + \gamma \\ \text{subject to} & \sum_{j=1}^{\sigma} \alpha_j (x_i - y_{ji}) + \sum_{j=1}^{c} \beta_j a_{ji} + \gamma \geq 0 \quad i \in \{1, \ldots, d\} \\ & \alpha_j \in [0, 1] \qquad j \in \{1, \ldots, \sigma\} \\ & \sum_{j=1}^{\sigma} \alpha_j = 1 \\ & \beta_j \geq 0 \qquad j \in \{1, \ldots, c\}, \end{array} \tag{43}$$

where the α_j variables, $j \in \{1, \ldots, \sigma\}$, correspond to the first σ constraints of the original LP problem (28), the β_j variables, $j \in \{1, \ldots, c\}$, to the c constraints on the weights, and the γ variable originates from the constraint $\sum_{i=1}^{d} w_i = 1$.

Assume $t \notin \text{PO}(r; \mathcal{F})$. Then, $\phi^* \leq 0$ (System (28) cannot be infeasible, since we assumed $\mathcal{W}(C) \neq \emptyset$). By the duality theorem of LP, System (43) is also feasible and has an optimal solution $(\alpha_1^*, \ldots, \alpha_\sigma^*, \beta_1^*, \ldots, \beta_c^*, \gamma^*)$ such that $\phi^* = \sum_{j=1}^{c} k_j \beta_j^* + \gamma^*$. Let $y_i^* = \sum_{j=1}^{\sigma} \alpha_j^* y_{ji}$, $i \in \{1, \ldots, d\}$. Since $\sum_{j=1}^{\sigma} \alpha_j^* = 1$, the following inequalities, derived from the first d constraints of (43), hold:

$$x_i - y_i^* + \sum_{j=1}^{c} \beta_j^* a_{ji} + \gamma^* \geq 0, \qquad i \in \{1, \ldots, d\}. \tag{44}$$

Let us now multiply each of these inequalities by the i-th component of an arbitrary weight vector $W = (w_1, \ldots, w_d) \in \mathcal{W}(C)$ and sum member-wise the obtained inequalities. We have:

$$\sum_{i=1}^{d} w_i (x_i - y_i^*) + \sum_{i=1}^{d} w_i \sum_{j=1}^{c} \beta_j^* a_{ji} + \gamma^* \geq 0, \tag{45}$$

which can be rewritten as:

$$\sum_{i=1}^{d} w_i (x_i - y_i^*) + \sum_{j=1}^{c} \beta_j^* \sum_{i=1}^{d} w_i a_{ji} + \gamma^* \geq 0 \tag{46}$$

From the original LP problem we know that $\sum_{i=1}^{d} w_i a_{ji} \leq k_j$, $j \in \{1, \ldots, c\}$. Hence:

$$\sum_{i=1}^{d} w_i (x_i - y_i^*) + \sum_{j=1}^{c} \beta_j^* k_j + \gamma^* \geq 0. \tag{47}$$

Since $\sum_{j=1}^{c} \beta_j^* k_j + \gamma^* = \phi^* \leq 0$, we have $\sum_{i=1}^{d} w_i (x_i - y_i^*) \geq 0$, i.e.:

$$\sum_{i=1}^{d} w_i g_i(t[A_i]) \geq \sum_{i=1}^{d} w_i \sum_{j=1}^{\sigma} \alpha_j^* g_i(t_j[A_i]). \tag{48}$$

Since W is an arbitrary weight vector in $\mathcal{W}(C)$, the above inequality also holds for all the vertices $W^{(1)}, \ldots, W^{(q)}$ of $\mathcal{W}(C)$. Therefore, System (29) is satisfiable, which proves the result. □

Proposition 4.8 (2D PO TEST). *Let $\mathcal{F}$ be a set of MLW functions subject to a set C of linear constraints on weights and such that the g_i's and h are strictly monotone. Let t, u, and v be three tuples in $[0, 1]^2$ such that i) $\text{ND}(\{t, u, v\}; \mathcal{F}) = \{t, u, v\}$, and ii) there exists a convex combination of t and u that $\mathcal{F}$-dominates v. Then there exists a convex combination of t and u that dominates v.*

Proof. We assume $\Lambda = 1$, the case $\Lambda = -1$ being analogous. Let us denote $g_i(t[A_i]) = t_i$ for $i = 1, 2$ and let $f(t, w) = wt_1 + (1 - w)t_2$, and similarly for u, v, and a generic point s. Notice that, since h is assumed to be strictly monotone, it can be safely ignored for the purpose of checking $\mathcal{F}$-dominance and dominance, as in Theorem 4.7. We observe that, for the polytope of admissible weights to be convex, it can only have two (possibly coinciding) vertices $W^{(1)} = \langle \overline{w}, 1 - \overline{w} \rangle$ and $W^{(2)} = \langle \underline{w}, 1 - \underline{w} \rangle$, corresponding to a constraint of the form $\underline{w} \le w_1 \le \overline{w}$, or else convexity would be lost. Since $t \not\prec_{\mathcal{F}} v$ and $v \not\prec_{\mathcal{F}} t$, tuple t must be worse than u on one of the vertices and better on the other; similarly for u and v (but with the opposite vertices, or else every convex combination of t and u would be worse than v on a vertex). The following (49) shows one of the two possible choices (the other one is symmetric and thus not shown):

$$f(t, \underline{w}) > f(v, \underline{w}), \quad f(u, \overline{w}) > f(v, \overline{w}), \quad f(t, \overline{w}) < f(v, \overline{w}), \quad f(u, \underline{w}) < f(v, \underline{w}) \tag{49}$$

By transitivity of inequalities in (49), we have

$$f(t, \overline{w}) < f(u, \overline{w}), \quad f(t, \underline{w}) > f(u, \underline{w}) \tag{50}$$

Then, by Bolzano's theorem, there must be a weight value w^* in $[\underline{w}, \overline{w}]$ such that

$$f(t, w^*) = f(u, w^*) \tag{51}$$

Since there exists a convex combination of t and u that $\mathcal{F}$-dominates v, the following holds for some $\alpha \in [0, 1]$:

$$\begin{aligned} \alpha f(t, w^*) + (1 - \alpha) f(u, w^*) \le f(v, w^*), \text{ hence, by (51),} \\ f(t, w^*) \le f(v, w^*) \end{aligned} \tag{52}$$

The score on w^* of any convex combination s of t and u can be expressed as follows, with $\beta \in [0, 1]$:

$$\begin{aligned} f(s, w^*) = \beta f(t, w^*) + (1 - \beta) f(u, w^*), \text{ and, by (51),} \\ f(s, w^*) = f(t, w^*). \end{aligned} \tag{53}$$

Consider, in particular, a convex combination s such that $s[A_1] = v[A_1]$. Then, the marginal score s_2 is obtained by setting $s_1 = v_1$ in (53):

$$\begin{aligned} w^* v_1 + (1 - w^*) s_2 &= w^* t_1 + (1 - w^*) t_2 \\ s_2 &= t_2 + (t_1 - v_1) \frac{w^*}{1 - w^*} \end{aligned} \tag{54}$$

In order to prove the clam, it suffices to show that $s_2 \le v_2$, since the g_i's are strictly monotone, which implies $s[A_2] \le v[A_2]$:

$$\begin{aligned} t_2 + (t_1 - v_1) \frac{w^*}{1 - w^*} \le v_2, \text{ i.e.,} \\ w^* t_1 + (1 - w^*) t_2 \le w^* v_1 + (1 - w^*) v_2 \end{aligned} \tag{55}$$

where (55) is the same as (52). □

Proposition 4.11. *Let $\mathcal{F}$ be a set of homogeneous MLW functions, such that the g_i's and h are strictly monotone, and subject to a set C of linear constraints on weights. Then, $\mathcal{F}$ is tuple-distinguishing iff d linearly independent weight vectors exist in $\mathcal{W}(C)$.*

Proof. If part. Let $\{W^{(1)}, \ldots, W^{(d)}\}$ be a set of d linearly independent weight vectors, and let f_j be the function corresponding to $W^{(j)}$ ($j = 1, \ldots, d$). Consider an arbitrary tuple t with scores $f_1(t), \ldots, f_d(t)$, and the following system of d equations in the unknowns $X = (x_1, \ldots, x_d)$:

$$h\left(\sum_{i=1}^{d} w_i^{(j)} x_i\right) = f_j(t), \quad j \in \{1, \ldots, d\}, \text{ i.e., by strict monotonicity of } h,$$

$$\sum_{i=1}^{d} w_i^{(j)} x_i = \sum_{i=1}^{d} w_i^{(j)} g_i(t[A_i]), \quad j \in \{1, \ldots, d\}.$$

Since the d weight vectors are linearly independent, the rank of the matrix of coefficients is d, and thus the above system admits as unique solution $x_i = g_i(t[A_i])$, $i \in \{1, \ldots, d\}$. Since all the g_i's are strictly monotone, there is no tuple $u \neq t$ such that $f_1(t) = f_1(u), \ldots, f_d(t) = f_d(u)$, thus $\mathcal{F}$ is tuple-distinguishing.

Only-if part. Let $b < d$ be the maximum number of linearly independent weight vectors in $\mathcal{W}(C)$. Let $\{W^{(1)}, \ldots, W^{(b)}\}$ be one of such sets, and $\{f_1, \ldots, f_b\}$ be the corresponding functions. Then, for every weight vector $W \in \mathcal{W}(C)$ there exist real numbers $\beta_1, \ldots, \beta_b$ satisfying $W = \sum_{j=1}^{b} \beta_j W^{(j)}$. Let f be the function corresponding to W, and let t be an arbitrary tuple. Then we have:

$$h^{-1}(f(t)) = \sum_{i=1}^{d} w_i g_i(t[A_i]) = \sum_{i=1}^{d} \sum_{j=1}^{b} \beta_j w_i^{(j)} g_i(t[A_i]) = \sum_{j=1}^{b} \beta_j \sum_{i=1}^{d} w_i^{(j)} g_i(t[A_i]) = \sum_{j=1}^{b} \beta_j h^{-1}(f_j(t)),$$

i.e.,

$$f(t) = h\left(\sum_{j=1}^{b} \beta_j h^{-1}(f_j(t))\right). \tag{56}$$

By Proposition B.4, the set $\{f_1, \ldots, f_b\}$ is not tuple-distinguishing, since $b < d$. Therefore, there exist two tuples t and u such that $f_1(t) = f_1(u), \ldots, f_b(t) = f_b(u)$. Then, by Equation (56), $f(t) = f(u)$ also holds for each function $f \in \mathcal{F}$, thus proving that $\mathcal{F}$ is NTD. □

THEOREM 4.12 (TUPLE-DISTINGUISHABILITY OF MLW FUNCTIONS). *Let $\mathcal{F}$ be a set of homogeneous MLW functions such that the g_i's and h are strictly monotone, and subject to a set $C = \{C_1, \ldots, C_c\}$ 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, $\mathcal{F}$ is tuple-distinguishing iff there exists a weight vector $W^* = (w_1^*, \ldots, w_d^*) \in \mathcal{W}(C)$ such that $\sum_{i=1}^{d} a_{ji} w_i^* < k_j$ holds for $j \in \{1, \ldots, c\}$.*

Proof. If part. The existence of a weight vector $W^* = (w_1^*, \ldots, w_d^*) \in \mathcal{W}(C)$ such that $\sum_{i=1}^{d} a_{ji} w_i^* < k_j$ holds for $j \in \{1, \ldots, c\}$ amounts to saying that $\mathcal{W}(C)$ has the same dimensionality as the standard $(d-1)$-simplex (since its interior is not empty), which implies that $\mathcal{W}(C)$ includes a set of d linearly independent vectors (like the $(d-1)$-simplex). From the *If* part of the proof of Proposition 4.11 it then follows that $\mathcal{F}$ is tuple-distinguishing.

Only-if part. When the system of strict inequalities $\sum_{i=1}^{d} a_{ji} w_i < k_j$ ($j \in \{1, \ldots, c\}$) has no solution, the dimensionality of $\mathcal{W}(C)$ is strictly less than that of the standard $(d-1)$-simplex, which means that no set of d linearly independent weight vectors exists in $\mathcal{W}(C)$. The result then follows from the *Only-if* part of the proof of Proposition 4.11. □