# Marrying Top-k with Skyline Queries: Relaxing the Preference Input while Producing Output of Controllable Size

Kyriakos Mouratidis[†] Keming Li[‡] Bo Tang[‡§]
[†]School of Computing and Information Systems, Singapore Management University
[‡]Guangdong Provincial Key Laboratory of Brain-inspired Intelligent Computation,
Department of Computer Science and Engineering, Southern Univ. of Science and Technology
[§]Research Inst. of Trustworthy Autonomous Systems, Southern Univ. of Science and Technology
kyriakos@smu.edu.sg; likm2020@mail.sustech.edu.cn; tangb3@sustech.edu.cn

## ABSTRACT

The two most common paradigms to identify records of preference in a multi-objective setting rely either on dominance (e.g., the skyline operator) or on a utility function defined over the records' attributes (typically, using a top-$k$ query). Despite their proliferation, each of them has its own palpable drawbacks. Motivated by these drawbacks, we identify three hard requirements for practical decision support, namely, personalization, controllable output size, and flexibility in preference specification. With these requirements as a guide, we combine elements from both paradigms and propose two new operators, ORD and ORU. We perform a qualitative study to demonstrate how they work, and evaluate their performance against adaptations of previous work that mimic their output.

## CCS CONCEPTS

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

## KEYWORDS

Top-$k$ query; Skyline; Multi-dimensional datasets

**ACM Reference Format:**

Kyriakos Mouratidis, Keming Li, Bo Tang. 2021. Marrying Top-k with Skyline Queries: Relaxing the Preference Input while Producing Output of Controllable Size. In *Proceedings of the 2021 International Conference on Management of Data (SIGMOD'21), June 20–25, 2021, Virtual Event, China.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3448016.3457299

## 1 INTRODUCTION

In the era of ubiquitous access to the Internet, users are presented with numerous alternatives to cover their everyday needs. Choosing from the available alternatives generally entails the consideration of multiple, often conflicting aspects. Indeed, multi-objective optimization has been a traditional research topic [29, 44, 64], whose practical relevance has increased in the current, fully connected reality. For a large set of alternatives (i.e., $d$-dimensional records), there are two main paradigms to determine those of most interest to the user, namely, based on *dominance* or *ranking by utility.*

The first paradigm considers that a record *dominates* another if all its attributes are more preferable. The *skyline* includes the records that are not dominated [14], while the *k-skyband* those dominated by a maximum of $(k-1)$ others [58]. The dominance paradigm is intuitive to the user. On the downside, it has two major shortcomings: (i) it is not personalizable, reporting the same result for every user, and (ii) its output size (i.e., the number of reported records) is uncontrollable, and often overwhelming [12, 30].

*Personalization* (i.e., serving the specific preferences of an individual user) is a hard requirement for decision support, especially nowadays, when large amounts of personal information are available via smartphones, fitness trackers, online activities, etc. Regarding the output size, Hick's law, known since the 50's, suggests that controlling the number of results presented to the user is essential to the quality of the decision and to the user experience [34, 37]. That law has been used as a cornerstone in eCommerce applications, meta-search engines, etc [32, 35]. Dictating the output size is crucial also because of design considerations, such as display size, device capabilities, connection speed, etc. Hence, another hard requirement is for *output-size specified* (OSS) operators.

The second paradigm, ranking by utility, associates each record with a score via a (user-specific) function over the records' attributes. Most commonly, the utility function is a weighted sum, with user preferences expressed by $d$ per-attribute weights $w_i$ (together comprising a preference vector $\boldsymbol{w}$). This linear type of scoring has been the most proliferate since the inception of ranking by utility [23, 42], and is shown by user studies to effectively model the way humans assess tradeoffs in real-life multi-objective decisions [60].

Ranking by utility, in the form of a top-$k$ query, is both personalized and OSS. Its Achilles' heel, however, lies in specifying the "correct" weights, since a small change in $\boldsymbol{w}$ can drastically alter the top-$k$ result [39, 80]. Vector $\boldsymbol{w}$ is assumed to be either input directly by the user or somehow mined (e.g., via online behavior and review mining [41, 71], pairwise comparisons of example records [40, 60], or some other preference learning technique [24]). In the former case, a user cannot be reasonably expected to quantify with absolute precision the relative importance of the various attributes. In the latter, preference learning methods come with an understanding that the mined $\boldsymbol{w}$ is only an estimate. This shortcoming motivates a third hard requirement for practical decision support, which is *relaxed preference input*, i.e., some flexibility in the specified preferences.


*SIGMOD '21, June 20–25, 2021, Virtual Event, China*

ACM ISBN 978-1-4503-8343-1/21/06…$15.00
https://doi.org/10.1145/3448016.3457299

In this paper, we aim to bring together the strong points of both paradigms (dominance-based and ranking by utility), while avoiding their drawbacks. In particular, we propose two operators that uphold all the three hard requirements we have established, namely, (i) being OSS, (ii) being personalized, and (iii) having a relaxed preference input. To achieve personalization, we employ linear scoring, due to its demonstrated effectiveness in modeling human decision making [60]. However, we consider the input $\boldsymbol{w}$ as a best-effort estimate. We therefore relax it, by incrementally expanding it equally on all directions in the preference domain. Conceptually, at the original $\boldsymbol{w}$, this corresponds to ranking by utility (e.g., a top-$k$ query at $\boldsymbol{w}$). As the expansion radius grows, it gradually shifts towards standard dominance, including in the output additional records that cater to alternative preferences, similar to $\boldsymbol{w}$. The stopping radius is indirectly (yet strictly) determined by the desired output size $m$.

Research on both standard paradigms has considered their individual weaknesses, but no existing work satisfies all three hard requirements. The skyline literature includes formulations that control the output size by loosening the definition of dominance [15, 45, 72], identifying representatives [33, 46, 49, 65], or considering subspaces [16, 69]. For example, Lin et al. [49] report the $m$ skyline members that dominate the most non-skyline records, while Chan et al. [16] shortlist the $m$ records that belong to the most subspace skylines. These definitions aim to produce the most competitive or the most representative skyline records in a general sense, without a specific user in mind, thus lacking personalization.

Centered more on utility, studies on regret-minimizing sets report $m$ representative records from the dataset. Typically, they define the regret ratio as the relative difference between the utility of the top-scoring record in the selected subset and the top-scorer in the entire dataset. Their objective is to minimize the aggregate (usually, the maximum) regret ratio across every possible utility function [57, 74], i.e., the reported subset is meant to satisfy as it best can all possible users, without an intent for personalization.

Two recent studies, [20] and [54], attempt to relax the preference input in ranking by utility. That is, they assume that the preference input is a convex polytope $R$ instead of a vector $\boldsymbol{w}$. Concordantly, they report the records that could be among the $k$ most preferable for any $\boldsymbol{w} \in R$ (for $k = 1$ and $k \geq 1$ in [20] and [54], respectively). Unfortunately, these methods are not OSS and, worse yet, come without even an estimate of the output size, i.e., the user/application is in the dark on whether $R$ is too large or too small to produce, even approximately, the required number of records. Furthermore, these approaches may remove the need for a particular $\boldsymbol{w}$, but require specifying a polytope $R$ in the preference domain. Deciding $R$ is left to the user or application, a choice that becomes tougher considering that the dynamics in the preference domain are hard to gauge. In usability terms, specifying the output size $m$ is arguably more tangible and more relatable to the user/application than specifying a polytope in the preference domain.

Our operators, ORD and ORU, satisfy all three hard requirements. They expand the preference input $\boldsymbol{w}$ in a similar way, however, they retain a stronger flavor of either paradigm each. ORD employs an adaptive notion of dominance that is guided by $m$, while ORU sticks closer to ranking by utility. Hard requirements aside, practicality also demands responsiveness and scalability. We make geometric observations and establish propositions that lead to efficient processing. Our algorithms are orders of magnitude faster than adaptations of previous work, which can merely simulate the ORD/ORU output, and still, without OSS guarantees.

| Operator | Personalized | OSS | Flexible Input |
|---|---|---|---|
| Skyline/$k$-Skyband | ✘ | ✘ | ✔ |
| Top-$k$ | ✔ | ✔ | ✘ |
| OSS skylines | ✘ | ✔ | ✔ |
| Regret-minimizing sets | ✘ | ✔ | ✔ |
| Fixed-region techniques | ✔ | ✘ | ✔ |
| Proposed (ORD and ORU) | ✔ | ✔ | ✔ |

**Table 1: Multi-objective queries and their properties**

In Table 1, we summarize the properties of existing multi-objective queries, and juxtapose them with our operators. In Section 2, we offer a more comprehensive description of related work.

## 2 RELATED WORK

The two traditional alternatives to determine the most preferable records from a dataset $D$ with $d$ attributes, are based on dominance and on ranking by utility score. A record *dominates* another if it is at least as preferable in all dimensions, and strictly more preferable in at least one dimension. The records that are not dominated by any other comprise the *skyline* [14] while, more generally, those dominated by fewer than $k$ form the $k$-*skyband* [58]. In contrast, in the ranking approach, the score of a record is typically defined as the weighted sum of its attributes for a vector of $d$ user-specific weights. The top-$k$ set includes the $k$ records with the largest scores [38].

For large, indexed datasets, the most common processing algorithms in both cases follow the *branch-and-bound* methodology. BBS [58] visits index nodes and data records in increasing distance from the top corner of the data space (i.e., the corner with the maximum possible attribute values), using a min-heap to organize them by that distance. It maintains as skyline/$k$-skyband the records dominated by none/fewer than $k$ records encountered so far. BBR [66] computes the top-$k$ set by visiting index nodes and data records in decreasing (upper bound of) score, using a max-heap. The first $k$ records popped from the heap are the top-$k$.

**OSS Skylines:** The size of the skyline is uncontrollable and oftentimes very large [30]. That being a major shortcoming, there have been several approaches to limit it.

Chan et al. [15] consider that a record $\boldsymbol{r}_i$ *m-dominates* another $\boldsymbol{r}_j$ for an $m \leq d$ if there is a subspace of $m$ dimensions where $\boldsymbol{r}_i$ dominates $\boldsymbol{r}_j$. A smaller $m$ implies a smaller skyline, thus controlling its size. Koltun and Papadimitriou [45] propose $\epsilon$-dominance, where the attributes of a record $\boldsymbol{r}_i$ are multiplied by $(1+\epsilon)$ to check whether it dominates another record $\boldsymbol{r}_j$. In the same spirit, Xia et al. [72] increment the attributes of $\boldsymbol{r}_i$ by an absolute $\delta$ value on all (appropriately scaled) dimensions. Other studies aim to select the $m$ most representative skyline records. The *dominance count* of a record, i.e., the number of records it dominates, has been used as a measure of its importance [28, 68, 77]. By that intuition, Lin et al. [49] choose the $m$ skyline records that dominate the most other records. Lee and Hwang [46] propose a pivot-based space partitioning for that problem, while Gao et al. [25] enhance it by favoring representatives that dominate the less frequently dominated non-skyline records. The latter's performance is improved by Han et al. [33]. By a different, distance-based intuition, Tao et al. [65]

choose as representatives the $m$ skyline records that minimize the distance from the remaining skyline members.

Sarma et al. [61] pick $m$ records from the skyline, so that the probability that a random user would click on one of them is maximized. Assuming that a record $\boldsymbol{r}_i$ is interesting if its attributes exceed a certain threshold per dimension, and that the distribution of the threshold values is known, they propose approximate and sampling methods to select the $m$ representatives. Magnani et al. [50] consider various measures of diversity and significance, and assume a linear combination of these two factors as the objective function that the $m$ chosen skyline records must maximize.

Another approach considers membership in *subspace* skylines [59, 67]. Chan et al. [16] report the $m$ skyline records that appear in the most subspace skylines. Vlachou and Vazirgiannis [69] measure importance according to dominance in different subspaces, and assume propagation of importance via dominance links. Another formulation considers that some attributes are more important [43, 52]; Lee et al. [47] select representatives according to skyline membership in the induced prioritized subspaces.

Most OSS skylines do not take into account a user's personal preferences. An exception, in abstract terms at least, are Bartolini et al. [10], who consider that record attributes correspond to user-specific ratings. If a user has not provided ratings for records $\boldsymbol{r}_i$ and $\boldsymbol{r}_j$, but at least a fraction of similar users have indicated ratings where $\boldsymbol{r}_i$ dominates $\boldsymbol{r}_j$, the same is assumed for the user at hand too. The required fraction indirectly controls the skyline size. The focus in [10] is to infer dominance when user ratings (i.e., record attributes) are missing. In contrast, in our target applications the records' attributes are given and no information for other users is required. Another distinction between OSS skylines and our work is that they consider $k = 1$, i.e., once dominated, a record is eliminated. Instead, our operators may dig deeper, to larger $k$ values, because they can rely on the personal preferences (roughly) specified by $\boldsymbol{w}$.

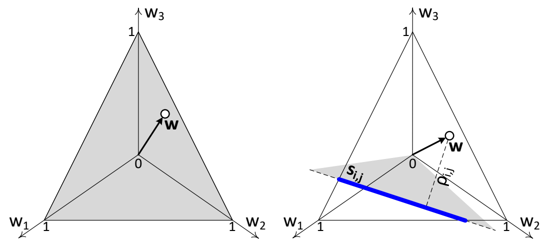
**Regret Minimization:** Work on *regret-minimizing sets* (RMS) produces an $m$-sized subset $S \subset D$ that tries to satisfy as it best can any possible user. In the original formulation [57], the regret ratio for a user is defined as the relative difference between the maximum utility score in $S$ and that in the entire $D$. The objective for $S$ is to minimize the maximum regret ratio for any possible user. There have been many follow-up studies (e.g., [7, 75]), considering also RMS variants, most notably $k$-RMS [18] (where the regret ratio reflects the difference between the top-scorer in $S$ and the top-$k$-th in $D$), minimizing the *average* regret ratio [79], defining regret based on the rank of records [8], etc. A survey is given in [74]. RMS studies are not concerned with personalization. Also, even if fed with our operators' stopping radius, they cannot reproduce our output. For example, to solve classic RMS [57], it suffices to consider only skyline records. In contrast, our output may also include records below the skyline. Moreover, RMS and its common variants are NP-hard for $d > 2$ [74]. Thus, research has focused on approximate solutions. Conversely, we develop exact ORD/ORU algorithms.

Inspired by RMS, but aiming for personalization, *interactive regret minimization* (IRM) involves the user in the search process [56]. Initially oblivious of her preferences, IRM goes through multiple rounds of interaction. In each round, it presents her with a number of records and asks her to choose the best, thus learning her (latent) preference vector increasingly well. When the regret ratio is guaranteed to be small enough (or the actual top-scorer is found), the last record chosen becomes the answer for this user. The original IRM method [56] involves artificial records in its interactions, which is resolved in [73]. The latter is enhanced in [81] by asking the user to sort the presented records (instead of just choosing the best). IRM assumes a different query processing model altogether, requiring active user involvement. Moreover, its objective is to eventually identify the one record with maximum utility, and thus considers only records on the skyline or convex hull.

**Fixed-region Techniques:** The closest related studies to our work are [20] and [54]. Given a convex preference polytope $R$, Ciaccia and Martinenghi [20] define that $\boldsymbol{r}_i$ *R-dominates* $\boldsymbol{r}_j$ if $\boldsymbol{r}_i$ scores higher than $\boldsymbol{r}_j$ for any $\boldsymbol{w} \in R$. They propose an $R$-dominance test which checks one linear condition per extreme vertex of $R$, and compute the *R-skyline* (i.e., the records that are not $R$-dominated by any other) by integrating that test into standard skyline algorithms. They also introduce an operator that reports as *potentially optimal* every $\boldsymbol{r}_i$ that is the top record for at least one $\boldsymbol{w} \in R$. To check a record for potential optimality, they solve a linear programming (LP) problem defined according to the extreme vertices of $R$. Mouratidis and Tang [54] extend potential optimality to $k \geq 1$, i.e., they identify the records that appear in the top-$k$ result for at least one $\boldsymbol{w} \in R$. In a more advanced variant, they explicitly report every possible (order-insensitive) top-$k$ set for any $\boldsymbol{w} \in R$. They first disqualify records $R$-dominated by $k$ or more others. Among the remaining candidates, they determine the top-$k$-th in each partition of $R$ and, accordingly, the (order-insensitive) prefix of the top-$k$ set.

In terms of practicality, the operators in [20] and [54] lack the OSS property, meaning that the user/application cannot determine (or even predict) the size of the output. The techniques themselves cannot be extended to our problem, because they rely on $R$ being fixed and given in advance. For example, their $R$-dominance and LP tests are defined according to the extreme vertices of $R$, and are contingent on these vertices being fixed and known. Furthermore, they require $R$ to be a convex polytope. In contrast, in our case $R$ is not specified, and the preference region (even if it were given in advance) is effectively a hyper-sphere, not a polytope. If we approximate hyper-spheres with hyper-cubes and make repetitive calls for different side-lengths of $R$ in an exploratory manner, the approaches in [20] (for $k = 1$) or [54] (for general $k$) could somehow simulate our operators, but even with that slack, they would require an excessive number of trials/executions to produce an output of exactly $m$ records. In other words, a second compromise is necessary, i.e., allow them to terminate when the output size is "almost" $m$ (e.g., within a 10% deviation). Our framework not only produces output of the exact desired size (strictly OSS), but it also reports *order-sensitive* top-$k$ results anywhere within its stopping radius $\rho$.

**Related Top-$k$ Work:** On the top-$k$ front, there are studies for unspecific or unknown preference vector $\boldsymbol{w}$ that are somewhat related to our work. For example, Soliman et al. [63] compute the most probable top-$k$ result if $\boldsymbol{w}$ is a random, uniformly distributed vector. Uncertain records/attributes have also been considered, leading to probabilistic top-$k$ outputs [6, 21, 76]. On the other hand, Zhang et al. [80] compute the preference region that corresponds to a given top-$k$ result, in a task which, loosely speaking, is inverse from ours.




(a) The preference domain for $d = 3$ (b) Hyper-plane $U_{\boldsymbol{v}}(\boldsymbol{r}_i) = U_{\boldsymbol{v}}(\boldsymbol{r}_j)$

**Figure 1: Preference domain, and minidist $\rho_{i,j}$ example**

# 3 PROBLEM FORMULATION

We consider that the available options are represented as $d$-dimensional records $\boldsymbol{r} = \langle x_1, x_2, ..., x_d \rangle$ in a dataset $D$ indexed by a spatial access method, e.g., an R-tree [11, 62]. We make the convention that the larger the attributes the better, yet our findings adapt easily to cases where some/all attributes are to be minimized. Given a preference vector $\boldsymbol{v}$ of $d$ non-negative weights $w_i$, the utility score of a record $\boldsymbol{r}$ is defined as their inner product, i.e., $U_{\boldsymbol{v}}(\boldsymbol{r}) = \sum_{i=1}^{d} x_i \cdot w_i$. Accordingly, the top-$k$ result comprises the $k$ records with the highest scores. Ordering $D$ by utility is independent from the magnitude of $\boldsymbol{v}$ [36, 48], thus we assume preference vectors where $\sum_{i=1}^{d} w_i = 1$. In other words, the domain of the preference vectors, called *preference domain*, is the unit $(d-1)$-simplex in a space whose $d$ axes correspond to the $w_i$ values, i.e., the simplex $\Delta^{d-1} = \{\boldsymbol{v} \in \mathbb{R}_+^d | \sum_{i=1}^{d} w_i = 1\}$. For $d = 3$, the preference domain is an equilateral triangle, shown in gray in Figure 1(a). Effectively, any valid preference vector is represented as a vertex in that triangle. For $d = 4$, the preference domain is a tetrahedron, and so on.

Let $\boldsymbol{w}$ be a best-effort estimate of the user's preference vector, henceforth called the *seed*, and consider the preference vectors $\boldsymbol{v}$ within distance $\rho$ from $\boldsymbol{w}$, i.e., where $|\boldsymbol{v} - \boldsymbol{w}| \leq \rho$. If a record $\boldsymbol{r}_i$ scores at least as high as another $\boldsymbol{r}_j$ for every such vector $\boldsymbol{v}$, and strictly higher for at least one of them, we say that $\boldsymbol{r}_i$ *$\rho$-dominates* $\boldsymbol{r}_j$. The records that are $\rho$-dominated by fewer than $k$ others form the *$\rho$-skyband*. This general notion includes the *$\rho$-skyline* as a special case for $k = 1$. Note that a larger $\rho$ implies a larger $\rho$-skyband. In the extreme settings, $\rho = 0$ renders the $\rho$-skyband equivalent to a traditional top-$k$ query at $\boldsymbol{w}$, while $\rho = \infty$ makes it equivalent to the standard $k$-skyband[1]. We may now define our first operator, abbreviated as ORD to stress its OSS property, relaxed input, and stronger dominance-oriented flavor.

Definition 1 (ORD). *Given the seed vector* $\mathbf{w}$ *and the required output size m,* ORD *reports the records that are $\rho$-dominated by fewer than k others, for the minimum $\rho$ that produces exactly m records in the output.*

Observe that user and application are both transparent to $\rho$, which relieves them from being concerned with the complex dynamics of the preference domain. The appropriate $\rho$ is determined automatically by our framework, according to the desired output size $m$. Our second operator shares that trait too, but follows more closely the ranking by utility paradigm, thus its abbreviation, ORU.

[1]If $\boldsymbol{r}_i$ scores no lower than $\boldsymbol{r}_j$ for every vector in the preference domain, and higher for at least one vector therein, then (and only then) $\boldsymbol{r}_i$ dominates $\boldsymbol{r}_j$ [19].

Definition 2 (ORU). *Given the seed vector* $\mathbf{w}$ *and the required output size m,* ORU *reports the records that belong to the top-k result for at least one preference vector within distance $\rho$ from* $\mathbf{w}$*, for the minimum $\rho$ that produces exactly m records in the output.*

While beyond the requirements of Definition 2, a byproduct of our ORU algorithm is the reporting of the specific (order-sensitive) top-$k$ result for any vector within radius $\rho$ from $\boldsymbol{w}$. This enables additional applications, like determining the most stable [80] or the most representative [63] top-$k$ results in the vicinity of $\boldsymbol{w}$, according to the volume of the preference regions that produce them.

Our ORD/ORU techniques require no precomputation other than a general-purpose spatial index on $D$. This implies that updates in $D$ affect only (and are readily supported by) the index. Also, it enables the integration of common predicates into our framework. For example, should the user impose arbitrary range predicates (e.g., price between \$150 and \$200, size between 400ft$^2$ and 600ft$^2$, etc), we may execute a multi-dimensional range query on $D$, followed by ORD/ORU in the selected part of the index/dataset.

Multi-objective querying generally loses its meaning in high dimensions. For instance, for more than a handful of dimensions almost every record tends to belong to the skyline [16, 30], while utility-wise the scores of all records tend to converge [55, 78]. We, hence, focus on low-dimensional settings. A final remark is that although we position our work within preference-based record shortlisting for a human user, our techniques apply to general multi-objective scenarios where the suitability of available options is defined by a linear function over the options' attributes.

# 4 OSS DOMINANCE-BASED OPERATOR

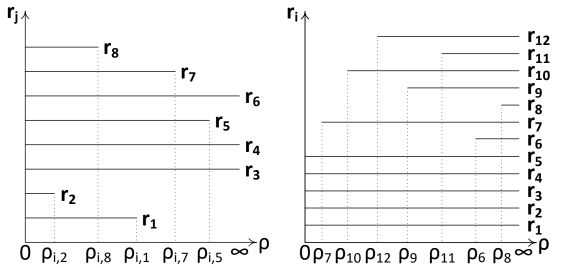
The output of ORD is a $\rho$-skyband, and in particular the one for the smallest $\rho$ that includes $m$ records. We make several observations that lead to an efficient ORD processing methodology.

## 4.1 Observations and Main Idea

**Properties of Candidate Records:** Without loss of generality, assume that no two records coincide or score the same for the seed vector $\boldsymbol{w}$. Unless a record belongs to the traditional $k$-skyband, it cannot belong to the top-$k$ result for any preference vector [14]. Hence, for any $\rho$, the $\rho$-skyband is a subset of the $k$-skyband. Therefore, the latter includes all the candidates we may need for ORD. Consider a record $\boldsymbol{r}_i$ among them. The remaining candidates fall into three categories regarding their potential to $\rho$-dominate $\boldsymbol{r}_i$:

- Records that score lower than $\boldsymbol{r}_i$ for the seed vector $\boldsymbol{w}$ cannot $\rho$-dominate it for any radius $\rho$ (because any $\rho$ includes the seed itself). By the way, a corollary of this is that the top-$k$ records of $\boldsymbol{w}$ belong to the $\rho$-skyband for every $\rho$.
- Records that dominate $\boldsymbol{r}_i$ in the traditional sense, score higher for any preference vector, thus they $\rho$-dominate $\boldsymbol{r}_i$ for any $\rho$.
- The remaining records (i.e., those that do not dominate $\boldsymbol{r}_i$ but score higher than it for $\boldsymbol{w}$) $\rho$-dominate $\boldsymbol{r}_i$ for a non-empty range of $\rho$ values, as we explain next.

Consider a record $\boldsymbol{r}_j$ that falls in the third category. As such, it does not dominate $\boldsymbol{r}_i$. Also, since $U_{\boldsymbol{w}}(\boldsymbol{r}_j) > U_{\boldsymbol{w}}(\boldsymbol{r}_i)$, record $\boldsymbol{r}_j$ is not dominated by $\boldsymbol{r}_i$ either. Every pair of records that do not dominate each other define a hyper-plane with equation $U_{\boldsymbol{v}}(\boldsymbol{r}_i) = U_{\boldsymbol{v}}(\boldsymbol{r}_j)$ that




(a) Intervals where $\boldsymbol{r}_i$ is $\rho$-dominated (b) $\rho$-skyband membership intervals

**Figure 2: Determining the** ORD **output**

cuts through the preference domain, i.e., it divides $\Delta^{d-1}$ into two non-empty parts. Since $\boldsymbol{r}_j$ scores higher than $\boldsymbol{r}_i$ for the seed $\boldsymbol{w}$, it holds that $U_{\boldsymbol{v}}(\boldsymbol{r}_i) < U_{\boldsymbol{v}}(\boldsymbol{r}_j)$ in the entire part that includes $\boldsymbol{w}$, while $U_{\boldsymbol{v}}(\boldsymbol{r}_i) > U_{\boldsymbol{v}}(\boldsymbol{r}_j)$ in the other part. Assuming $d = 3$, Figure 1(b) illustrates in gray a hyper-plane with equation $U_{\boldsymbol{v}}(\boldsymbol{r}_i) = U_{\boldsymbol{v}}(\boldsymbol{r}_j)$.

Let $\boldsymbol{s}_{i,j}$ be the intersection of hyper-plane $U_{\boldsymbol{v}}(\boldsymbol{r}_i) = U_{\boldsymbol{v}}(\boldsymbol{r}_j)$ with the preference domain $\Delta^{d-1}$. Geometrically, $\boldsymbol{s}_{i,j}$ is a $(d-2)$-simplex, e.g., for $d = 3$ it is a line segment, shown bold in Figure 1(b). Let also $\rho_{i,j}$ be the minimum distance between $\boldsymbol{w}$ and any $\boldsymbol{v} \in \boldsymbol{s}_{i,j}$. For any preference vector in $\Delta^{d-1}$ that is within distance $\rho_{i,j}$ from $\boldsymbol{w}$, record $\boldsymbol{r}_j$ scores higher than $\boldsymbol{r}_i$, i.e., $\boldsymbol{r}_j$ $\rho$-dominates $\boldsymbol{r}_i$ for every $\rho \leq \rho_{i,j}$. In contrast, it does not $\rho$-dominate $\boldsymbol{r}_i$ for $\rho > \rho_{i,j}$. In implementation terms, we can compute the mindist $\rho_{i,j}$ using a quadratic programming solver [31, 53] with (squared) distance as the minimization objective, subject to the linear constraints that define $\boldsymbol{s}_{i,j}$, i.e., $U_{\boldsymbol{v}}(\boldsymbol{r}_i) = U_{\boldsymbol{v}}(\boldsymbol{r}_j)$ and $\sum_{i=1}^{d} w_i = 1$.

**Inflection Radius:** By considering all records $\boldsymbol{r}_j$ in the third category against $\boldsymbol{r}_i$, they are each mapped into an interval of $\rho$ values where they $\rho$-dominate it. Figure 2(a) offers an example. Assume that $k = 5$ and that the 5-skyband includes 8 records that score higher than $\boldsymbol{r}_i$ for $\boldsymbol{w}$. Out of these, $\boldsymbol{r}_i$ is dominated in the traditional sense by 3 (i.e., $\boldsymbol{r}_3$, $\boldsymbol{r}_4$, $\boldsymbol{r}_6$), thus their infinite intervals. The remaining 5 do not dominate $\boldsymbol{r}_i$, hence, they fall in the third category; they each have a finite mindist $\rho_{i,j}$, mapped to intervals as illustrated. By sweeping the intervals from left to right, we can easily identify the $\rho$ value past which $\boldsymbol{r}_i$ is dominated by fewer than $k$ others, i.e., it becomes part of the $\rho$-skyband. We call that value the *inflection radius* of $\boldsymbol{r}_i$ and denote it as $\rho_i$. In our example, $\rho_i = \rho_{i,7}$.

**A Preliminary Approach:** Based on the above, a first-cut ORD solution is to compute the entire $k$-skyband, and for each record in it, to derive the inflection radius. Then, to output the $m$ of them with the smallest inflection radii. To exemplify, assume that the 5-skyband includes 12 records; in Figure 2(b), we map each of them to an interval of $\rho$ values where it belongs to the $\rho$-skyband, according to its inflection radius. These intervals have different meaning from Figure 2(a). In Figure 2(a), all intervals refer to $\boldsymbol{r}_i$, helping to compute its own inflection radius $\rho_i$. In contrast, Figure 2(b) is a global representation of the $\rho$-skyband for different $\rho$ values. Specifically, if we sweep the chart with a vertical line, the intervals that intersect the line at any position, indicate the $\rho$-skyband members for the $\rho$ value that corresponds to that position. In our example, assuming that $m = 8$, the ORD output is set $\{\boldsymbol{r}_1, \boldsymbol{r}_2, \boldsymbol{r}_3, \boldsymbol{r}_4, \boldsymbol{r}_5, \boldsymbol{r}_7, \boldsymbol{r}_{10}, \boldsymbol{r}_{12}\}$, which corresponds to the $\rho$-skyband for $\rho = \rho_{12}$.

An interesting insight is that the ORD output may vary from standard ranking by utility (top-$k$) all the way to traditional dominance-based querying ($k$-skyband), depending on $m$. On the one hand, by definition, the top-$k$ records are the only members of the $\rho$-skyband for $\rho = 0$, which corresponds to the smallest possible $m$ (i.e., $m = k$). On the other hand, every $k$-skyband member will appear to the $\rho$-skyband for a sufficiently large $\rho$ or, equivalently, for a sufficiently large $m$. To visualize these extremes, at the leftmost position in Figure 2(b) the sweeping line intersects only the intervals of the top-$k$ records ($\boldsymbol{r}_1$ to $\boldsymbol{r}_5$), while at the rightmost[2] the entire $k$-skyband. That said, although this generality is welcome, the practical strength of ORD is for $m$ values in between these extremes.

## 4.2 Efficient ORD Processing

The ORD processing idea described so far is a foundation that offers an abstract-level understanding of our method. However, an efficient solution must address several performance issues. Primarily, we would want to avoid computing the entire $k$-skyband in the beginning of the process. Indeed, the $k$-skyband may include numerous records, many times more than the $m$ required [12, 30]. Ideally, we want to limit the number of considered candidates to as tight a superset of the ORD output as possible. The algorithm we present next serves that objective.

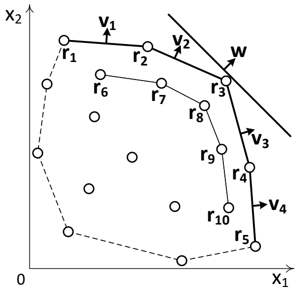
We first invoke a progressive $k$-skyband retrieval that fetches its members one by one, and place them into a candidate set. Importantly, unlike standard $k$-skyband computation, we enforce that its members are fetched in decreasing score order for $\boldsymbol{w}$ (we will explain how shortly). This retrieval order is essential, because when a new candidate $\boldsymbol{r}_i$ is fetched, we can definitively compute its inflection radius $\rho_i$ already, without having to derive the entire $k$-skyband. The rationale is that only $k$-skyband records with higher score may $\rho$-dominate $\boldsymbol{r}_i$, and these are guaranteed to be fetched before it.

We keep fetching new $k$-skyband members in that fashion, until the candidate set reaches size $(m + 1)$. At that stage, we evict the candidate with the largest inflection radius. Also, an important algorithmic shift takes place. Let $\bar{\rho}$ be the maximum inflection radius of the remaining $m$ candidates. The $\bar{\rho}$-skyband is guaranteed to include at least the $m$ existing candidates, thus $\bar{\rho}$ upper bounds the eventual stopping radius of the algorithm. Therefore, from this point onwards, we switch to fetching $\bar{\rho}$-skyband members (instead of $k$-skyband members), still in decreasing order of score for $w$. The switch can be performed transparently, as we elaborate later.

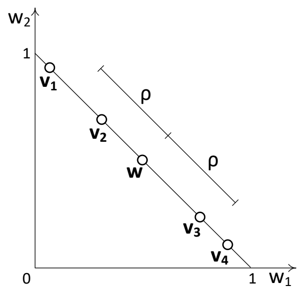
To exemplify, consider Figure 2(b). Assume that $k = 5$, $m = 11$, and that we have fetched the $m + 1 = 12$ depicted records, in the order indicated by their subscripts, i.e., $\boldsymbol{r}_i$ was fetched $i$-th. To bring the candidates down to $m = 11$, we discard $\boldsymbol{r}_8$, as it has the largest inflection radius. Practically, that sets $\bar{\rho}$ to $\rho_8$.

As new candidates are fetched, and evictions are made to keep their total number to $m$, the current $\bar{\rho}$ keeps shrinking. Meanwhile, as $\bar{\rho}$ shrinks, records tend to $\rho$-dominate more others. This implies that the $\bar{\rho}$-skyband retrieval becomes increasingly more selective, thus filtering out more aggressively regular $k$-skyband members that cannot participate in the ORD result. When the $\bar{\rho}$-skyband module cannot fetch any more records, the candidate set is finalized

[2]The largest meaningful $\rho$ (although visualized as $\infty$) is the distance between $\boldsymbol{w}$ and its furthest point in $\Delta^{d-1}$, since that $\rho$ already covers the entire preference domain.




(a) Convex hull, facets, and norms




(b) Top-regions for layer $L_1$

**Figure 3: Fundamental notions and principles**

as the ORD result. The latter corresponds to the $\rho$-skyband for $\rho$ equal to the maximum inflection radius across its members.

The ORD algorithm relies on a progressive $k$-skyband module, with the extra requirement to fetch records in decreasing score according to $\boldsymbol{w}$. We use an adaptation of BBS [58] where we visit index nodes and records in decreasing (upper bound of) score for $\boldsymbol{w}$, using a max-heap. Once the $(m+1)$-th record is fetched, we shift to $\bar{\rho}$-skyband computation using the exact same heap as per normal, but replace the regular dominance tests of BBS with $\bar{\rho}$-dominance, for the current $\bar{\rho}$ value. The visiting order by score and the use of $\bar{\rho}$-dominance tests instead of regular dominance are permissible modifications to vanilla BBS, because its correctness is guaranteed as long as no record $\boldsymbol{r}_j$ fetched after another $\boldsymbol{r}_i$ may dominate $\boldsymbol{r}_i$ [58]. That property is upheld by our visiting order (by score), both initially for regular dominance, and after the shift to $\bar{\rho}$-dominance. Indeed, $U_{\boldsymbol{w}}(\boldsymbol{r}_i) > U_{\boldsymbol{w}}(\boldsymbol{r}_j)$ ensures that $\boldsymbol{r}_j$ cannot dominate nor $\rho$-dominate $\boldsymbol{r}_i$ for any $\rho$. An implementation note on the adapted BBS regards its $\rho$-dominance building block. That block tests whether an already-fetched $\bar{\rho}$-skyband record $\boldsymbol{r}_i$ $\bar{\rho}$-dominates a not-yet-fetched record $\boldsymbol{r}_j$ (or an unvisited index node whose top corner is $\boldsymbol{r}_j$). The test is performed as explained in Section 4.1, by comparing the mindist $\rho_{i,j}$ with $\bar{\rho}$.

## 5 OSS UTILITY-BASED OPERATOR

Our second operator, ORU, adheres more closely to ranking by utility; it reports records that belong to the top-$k$ for at least one preference vector within radius $\rho$ from the seed $\boldsymbol{w}$, for the minimum $\rho$ that produces exactly $m$ records. Despite the seeming similarity to ORD's definition, the top-$k$ ranking involved in ORU renders it innately different, and its solution considerably more complex.

We present important preliminaries (in Section 5.1), a crucial theorem and algorithmic basis to process ORU (in Section 5.2), and eventually a complete implementation (in Section 5.3).

### 5.1 Fundamentals

The abstractions and techniques used for ORU have the notion of the *convex hull* at their core [13]. The convex hull of $D$ is the smallest convex polytope that encloses all its records. It comprises facets, each defined by $d$ extreme vertices (records) in general position. The outer polygon in Figure 3(a) is the convex hull of an example dataset. Facet $\boldsymbol{r}_1\boldsymbol{r}_2$ is defined by extreme vertices $\boldsymbol{r}_1$ and $\boldsymbol{r}_2$, etc.

A vector is *normal* to a hyper-plane when its direction is perpendicular to the hyper-plane. The *norm* of a facet on the hull is the normal vector to that facet whose sum of coordinates is 1, and is directed towards the exterior of the hull. In our example, the norm of $\boldsymbol{r}_1\boldsymbol{r}_2$ is vector $\boldsymbol{v}_1$. Effectively, the norm of a facet can be seen as a point in the preference domain $\Delta^{d-1}$.

The top record for a preference vector $\boldsymbol{v}$ is the one met first by a hyper-plane normal to $\boldsymbol{v}$ that sweeps the data space from the top corner to the origin [22, 27]. Hence, the top record in $D$ is guaranteed to lie on its convex hull [17, 51]. Since in our case the weights are non-negative, the top record is among the extreme vertices of facets with non-negative norms. We call *upper hull* the part that corresponds to these facets. In Figure 3(a), the upper hull is bold (and the rest of the convex hull is dashed). For the shown $\boldsymbol{w}$, the top record is $\boldsymbol{r}_3$, as it is met first by the sweeping line normal to $\boldsymbol{w}$.

To explain the fundamentals of our methodology, ***assume*** that we have already computed the first $k$ upper hull layers: the first layer, $L_1$, includes the upper hull of $D$; the second, $L_2$, includes the upper hull of $D-L_1$; and, generally, layer $L_i$ the upper hull of $D$ after subtracting layers $L_1$ to $L_{i-1}$. In Figure 3(a), layer $L_1$ includes records $\boldsymbol{r}_1$ to $\boldsymbol{r}_5$, and $L_2$ records $\boldsymbol{r}_6$ to $\boldsymbol{r}_{10}$. Note that in reality our complete ORU algorithm *does not require such precomputation*, but instead it builds *on the fly* (i.e., at query time) *only parts* of the necessary layers, thus applying to arbitrary $k$, avoiding precomputation costs (time and space), and extending transparently to dynamic datasets, i.e., to cases where record insertions/deletions may occur in $D$ and invalidate precomputed information. Also, ***assume*** that we already know the necessary radius $\rho$ for ORU to produce $m$ records. Of course, this too is an assumption we will drop later (in Section 5.3).

**Adjacent Set $\mathcal{A}(\boldsymbol{r})$:** Consider a record $\boldsymbol{r}$ in layer $L_i$. We denote by $\mathcal{F}(\boldsymbol{r})$ the set of $L_i$ facets with $\boldsymbol{r}$ as one of their extreme vertices, and by $\mathcal{A}(\boldsymbol{r})$ the records adjacent to $\boldsymbol{r}$, i.e., the $L_i$ records (other than $\boldsymbol{r}$) that define facets in $\mathcal{F}(\boldsymbol{r})$. In Figure 3(a), for example, $\mathcal{F}(\boldsymbol{r}_3) = \{\boldsymbol{r}_2\boldsymbol{r}_3, \boldsymbol{r}_3\boldsymbol{r}_4\}$ and $\mathcal{A}(\boldsymbol{r}_3) = \{\boldsymbol{r}_2, \boldsymbol{r}_4\}$. The following Lemmas 1, 2, and 3 are crucial. Note that they refer to records within the same layer $L_i$.

Lemma 1. *Given a preference vector* $\mathbf{v}$ *whose top record in* $L_i$ *is* $\mathbf{r}$, *if we start shifting* $\mathbf{v}$ *towards any direction in the preference domain, the first record in* $L_i$ *to outscore* $\mathbf{r}$ *is always in* $\mathcal{A}(\mathbf{r})$, *i.e., among the records adjacent to* $\mathbf{r}$. *Furthermore, each of the records in* $\mathcal{A}(\mathbf{r})$ *is the first outscoring record for some shifting direction of* $\mathbf{v}$.

Proof. Let $H_{\boldsymbol{r}}$ be the hyper-plane in the data space that is normal to $\boldsymbol{v}$ and passes through $\boldsymbol{r}$. Record $\boldsymbol{r}$ is the top for $\boldsymbol{v}$, as long as there is no record *above* $H_{\boldsymbol{r}}$ (i.e., in the half-space that includes the top corner of the data space). Assume that $\boldsymbol{v}$ gradually shifts towards a specific direction, with $H_{\boldsymbol{r}}$ always passing through $\boldsymbol{r}$. As the orientation of $H_{\boldsymbol{r}}$ shifts together with $\boldsymbol{v}$, the first record in $L_i$ that is met by $H_{\boldsymbol{r}}$ is the record $\boldsymbol{r}_i$ that will outscore $\boldsymbol{r}$ if we shift $\boldsymbol{v}$ infinitesimally any further (in the same direction). Suppose that $\boldsymbol{r}_i$ is not in $\mathcal{A}(\boldsymbol{r})$, i.e., it shares no common $L_i$ facet with $\boldsymbol{r}$. At the time that $H_{\boldsymbol{r}}$ touches $\boldsymbol{r}_i$, according to the hypothesis, no other record in $L_i$ should lie above $H_{\boldsymbol{r}}$. This, however, is a contradiction, because the convexity of $L_i$ implies that any hyper-plane that passes through two non-adjacent records in $L_i$ ($\boldsymbol{r}$ and $\boldsymbol{r}_i$, in this case) cuts through the interior of $L_i$, i.e., there is at least one other extreme vertex (record) in $L_i$ that lies above $H_{\boldsymbol{r}}$. We conclude that the first record to outscore $\boldsymbol{r}$, for any direction of shifting $\boldsymbol{v}$, must be in $\mathcal{A}(\boldsymbol{r})$.

It remains to show that for each record $\boldsymbol{r}_i$ in $\mathcal{A}(\boldsymbol{r})$, there is a direction of shifting $\boldsymbol{v}$ that makes $\boldsymbol{r}_i$ the first outscoring record.

Let $f$ be a facet in $\mathcal{F}(\boldsymbol{r})$ where $\boldsymbol{r}_i$ is a defining vertex. Consider the shifting of $\boldsymbol{v}$ towards the norm of $f$, equivalently, the shifting of $H_{\boldsymbol{r}}$ until it falls on $f$. Since $f$ is a facet of the convex hull, it leaves all $L_i$ records towards its interior. Thus, there is no $L_i$ record above $H_{\boldsymbol{r}}$ at all times until now. After $H_{\boldsymbol{r}}$ has fallen on $f$, if $\boldsymbol{v}$ shifts infinitesimally towards $\boldsymbol{r}_i$, $\boldsymbol{r}_i$ will become the first to outscore $\boldsymbol{r}$. □

By Lemma 1, if $\boldsymbol{w}$ shifts clockwise/anticlockwise in Figure 3(a), $\boldsymbol{r}_4$ and $\boldsymbol{r}_2$, respectively, will be the first $L_1$ records to outscore $\boldsymbol{r}_3$.

**Top-region $C(\boldsymbol{r})$:** Building on Lemma 1, our next proposition reveals an important property within $L_i$, and helps define the *top-region* of a record $\boldsymbol{r} \in L_i$, i.e., the region $C(\boldsymbol{r})$ in the preference domain where every vector has $\boldsymbol{r}$ as its top record in $L_i$.

Lemma 2. *Let* $\mathbf{r}$ *be a record in* $L_i$. $\mathbf{r}$ *is the top-scorer across all* $L_i$ *records for those preference vectors* $\mathbf{v}$ *that fall in the convex polytope* $C(\mathbf{r})$ *defined by (i.e., whose extreme vertices correspond to) the norms of the facets in* $\mathcal{F}(\mathbf{r})$.

Proof. From Lemma 1, we infer that $C(\boldsymbol{r})$ is determined by records in $\mathcal{A}(\boldsymbol{r})$, since they are the first to outscore $\boldsymbol{r}$ once $\boldsymbol{v}$ leaves $C(\boldsymbol{r})$. In particular, each adjacent record $\boldsymbol{r}_i$ corresponds to a half-space $U_{\boldsymbol{v}}(\boldsymbol{r}) \geq U_{\boldsymbol{v}}(\boldsymbol{r}_i)$ in the preference domain (simply expressing that $\boldsymbol{r}$ should score no lower than $\boldsymbol{r}_i$ anywhere in $C(\boldsymbol{r})$). $C(\boldsymbol{r})$ is the intersection of all these half-spaces, which (by definition [13]) is a convex polytope. Each facet of $C(\boldsymbol{r})$ is attributed to one of the intersected half-spaces, say, $U_{\boldsymbol{v}}(\boldsymbol{r}) \geq U_{\boldsymbol{v}}(\boldsymbol{r}_i)$ and, in effect, to one of the adjacent records, i.e., $\boldsymbol{r}_i$ in this case. In general position, every extreme vertex of $C(\boldsymbol{r})$, say $\boldsymbol{v}_j$, corresponds to the intersection of $(d-1)$ of its facets, i.e., to $(d-1)$ equalities of the form $U_{\boldsymbol{v}}(\boldsymbol{r}) = U_{\boldsymbol{v}}(\boldsymbol{r}_i)$, where each $\boldsymbol{r}_i$ is adjacent to $\boldsymbol{r}$. Let $S$ be the record set composed of $\boldsymbol{r}$ and these specific $(d-1)$ adjacent records. As all records in $S$ have the same score according to $\boldsymbol{v}_j$, by Lemma 1, such a tie is only feasible if any pair of records in $S$ are adjacent to each other on $L_i$. Since, in general position, each facet on $L_i$ is defined by $d$ records, the records in $S$ define a facet $f$ in $\mathcal{F}(\boldsymbol{r})$, with $\boldsymbol{v}_j$ as its norm. In other words, there is a direct one-to-one mapping between the facets in $\mathcal{F}(\boldsymbol{r})$ and the extreme vertices of $C(\boldsymbol{r})$. □

By Lemma 2, $C(\boldsymbol{r})$ can be seen as a dual representation of $\mathcal{F}(\boldsymbol{r})$, where the former refers to the preference domain and the latter to the data space. Consider $\boldsymbol{r}_3$ in Figure 3(a). Facet set $\mathcal{F}(\boldsymbol{r}_3) = \{\boldsymbol{r}_2\boldsymbol{r}_3, \boldsymbol{r}_3\boldsymbol{r}_4\}$ translates to the top-region defined by their norms $\boldsymbol{v}_2$ and $\boldsymbol{v}_3$, i.e., $C(\boldsymbol{r}_3)$ is segment $\boldsymbol{v}_2\boldsymbol{v}_3$ in the preference domain. Note that for $d = 2$, the preference domain $\Delta^{d-1}$ is a line segment.

**Order Continuity:** Lemmas 1 and 2, in tandem, suggest a continuity in the score order among $L_i$ records for every $\boldsymbol{v}$. Specifically, the different top-regions for any given layer $L$ define a partitioning of the preference domain, with adjacent records $\boldsymbol{r}_i, \boldsymbol{r}_j$ in $L$ having neighboring top-regions $C(\boldsymbol{r}_i), C(\boldsymbol{r}_j)$ in $\Delta^{d-1}$. Considering layer $L_1$ in our running example, Figure 3(b) demonstrates the partitioning of the preference domain. The top-region of $\boldsymbol{r}_1$ is the segment from point $(0, 1)$ to $\boldsymbol{v}_1$; of $\boldsymbol{r}_2$ from $\boldsymbol{v}_1$ to $\boldsymbol{v}_2$; of $\boldsymbol{r}_3$ from $\boldsymbol{v}_2$ to $\boldsymbol{v}_3$, etc. Lemma 3 establishes a property for every vector in a top-region.

Lemma 3. *For any preference vector* $\mathbf{v} \in C(\mathbf{r})$*, the top-2-nd record in* $L_i$ *is always in* $\mathcal{A}(\mathbf{r})$*, i.e., among the records adjacent to* $\mathbf{r}$.

Proof. Let $H_{\boldsymbol{v}}$ be the hyper-plane (in data space) that is normal to $\boldsymbol{v}$. Sweeping the data space with $H_{\boldsymbol{v}}$, the first encountered record in $L_i$ is, by definition, $\boldsymbol{r}$. As sweeping continues further, the convexity of $L_i$ ensures that $H_{\boldsymbol{v}}$ cuts only through the facets in $\mathcal{F}(\boldsymbol{r})$. Since $L_i$ is hollow (i.e., has no records in its interior), the $L_i$ record to be encountered next (i.e., the top-2-nd in $L_i$) must be an extreme vertex of a facet in $\mathcal{F}(\boldsymbol{r})$, i.e., a record in $\mathcal{A}(\boldsymbol{r})$. □

In our example, Lemma 3 implies that the top-2-nd record in layer $L_1$ for any $\boldsymbol{v} \in C(\boldsymbol{r}_3)$ is either $\boldsymbol{r}_2$ or $\boldsymbol{r}_4$. Note that all three lemmas consider a layer in isolation. For instance, although $\boldsymbol{w} \in C(\boldsymbol{r}_3)$, its top-2-nd record in the entire $D$ is none of $\boldsymbol{r}_2$ or $\boldsymbol{r}_4$, but $\boldsymbol{r}_8$ from $L_2$.

## 5.2 An Algorithmic Basis for ORU

In this section, we prove an important theorem and provide an algorithmic basis to process ORU. Recall that we assumed we already know the minimum radius $\rho$ required to produce $m$ records, and that the first $k$ upper hull layers are precomputed. We do not drop these assumptions yet. Here we focus on determining the top-$k$ result for any possible preference vector within radius $\rho$ from the seed $\boldsymbol{w}$, in order to form the ORU output.

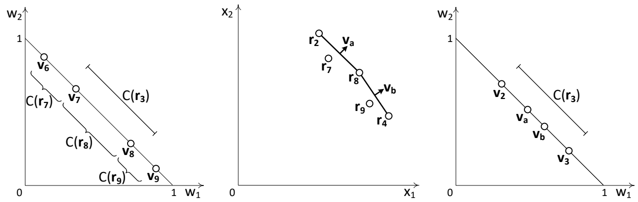
First, we find all records in layer $L_1$ whose top-region has mindist to $\boldsymbol{w}$ no greater than $\rho$. Let $C$ be one of these regions. We already know the top record in it, say, $\boldsymbol{r}$. Considering $C$ in isolation, our next task is to determine the top-2-nd record anywhere in it (i.e., for any possible preference vector $\boldsymbol{v} \in C$), and to partition $C$ accordingly. By Lemma 3, if we only considered $L_1$, the top-2-nd record for any $\boldsymbol{v} \in C$ would be among those adjacent to $\boldsymbol{r}$. On the other hand, in the remaining dataset (i.e., if we ignored the records in $L_1$), the top-2-nd record would be in $L_2$ and more specifically, by Lemma 2, among the $L_2$ records $\boldsymbol{r}_i$ whose top-region $C(\boldsymbol{r}_i)$ overlaps $C$. Theorem 1 generalizes this key observation.

Theorem 1. *Assume that anywhere in a preference region* $C$ *the (order-sensitive) top-*$i$ *result is the same and it is known. Also, let* $L_t$ *be the deepest layer that any of the top-*$i$ *records belongs to. The top-*$(i+1)$*-th record anywhere in* $C$ *must be in the union of:*

- *Set (i): The adjacent records to any member of the known top-*$i$ *result in its respective layer, and*
- *Set (ii): The records in the* $(t+1)$*-th layer (i.e.,* $L_{t+1}$*) whose top-region overlaps* $C$*.*

Proof. Let $S$ be the union of all records in the first $t$ layers. Due to Lemma 3, when it is applied to each of the top-$i$ records in their respective layer, if we only considered $S$, the top-$(i+1)$-th record would be in Set (i). On the other hand, if we only considered the rest of the dataset, i.e., $D - S$, the next highest-scoring record would be in $L_{t+1}$ and specifically, by Lemma 2, in Set (ii). Hence, in the overall product set $D$ (i.e., in the union of $S$ and $D - S$), the top-$(i+1)$-th record anywhere in $C$ must be in the union of Sets (i) and (ii). □

Returning to our processing description for region $C$, the top record (the order-sensitive top-$i$ result, in the general case) is already known and fixed anywhere in it, thus we can readily determine Set (i). We can also extract from $L_2$ (from $L_{t+1}$, in the general case) the part of the upper hull that corresponds to records in Set (ii); let us denote that part as $L_{prt}$. We update the upper hull $L_{prt}$ to also cover Set (i) records, and denote its updated version as $L_{upd}$. Next, we apply Lemma 2 to $L_{upd}$ to identify the top-2-nd records (the top-$(i+1)$-th, in the general case) for any $\boldsymbol{v} \in C$, and we partition $C$ accordingly. We continue this process recursively in each produced partition until the full, order-sensitive top-$k$ result is known anywhere in $C$. Repeating that process for all $L_1$ top-regions with mindist up to $\rho$, we derive all the required top-$k$ results.



(a) Top-regions for layer $L_2$ (b) $L_{upd}$ upper hull (c) Top-2 results in $C(\boldsymbol{r}_3)$

**Figure 4: Applying Theorem 1**

In our example, assume that $\rho$ is as shown in Figure 3(b). The $L_1$ top-regions with mindist from $\boldsymbol{w}$ up to $\rho$ correspond to $\boldsymbol{r}_2$, $\boldsymbol{r}_3$, and $\boldsymbol{r}_4$. Focusing on $C(\boldsymbol{r}_3)$ (i.e., segment $\boldsymbol{v}_2\boldsymbol{v}_3$), we know already that the top record is $\boldsymbol{r}_3$, and seek to find the top-2-nd. Set (i) includes $\boldsymbol{r}_2$ and $\boldsymbol{r}_4$. To determine Set (ii), we refer to $L_2$. Figure 4(a) illustrates the $L_2$ top-regions. Among them, those that overlap $C(\boldsymbol{r}_3)$ are $C(\boldsymbol{r}_7)$, $C(\boldsymbol{r}_8)$, and $C(\boldsymbol{r}_9)$. Thus, we form $L_{prt}$ as the part of $L_2$ that corresponds to $\boldsymbol{r}_7, \boldsymbol{r}_8, \boldsymbol{r}_9$. Updating $L_{prt}$ to also cover Set (i) (i.e., $\boldsymbol{r}_2, \boldsymbol{r}_4$), results in the upper hull $L_{upd}$ in Figure 4(b). $L_{upd}$ suggests that the top-2-nd record is one of $\boldsymbol{r}_2, \boldsymbol{r}_8, \boldsymbol{r}_4$. Furthermore, by Lemma 2, their $L_{upd}$ top-regions (determined by facet norms $\boldsymbol{v}_a$ and $\boldsymbol{v}_b$) help partition $C(\boldsymbol{r}_3)$ according to which exactly among them is the top-2-nd. Figure 4(c) presents the induced partitioning. The top-2 result is $\{\boldsymbol{r}_3, \boldsymbol{r}_2\}$ for preference vectors in $\boldsymbol{v}_2\boldsymbol{v}_a$; $\{\boldsymbol{r}_3, \boldsymbol{r}_8\}$ in $\boldsymbol{v}_a\boldsymbol{v}_b$; and $\{\boldsymbol{r}_3, \boldsymbol{r}_4\}$ in $\boldsymbol{v}_b\boldsymbol{v}_3$. The process repeats recursively in order to determine the top-3-rd record in each of the three partitions, and so on.

Observe that we may not need to reach as deep as the $k$-th layer, since members of Set (i) could prevent those of Set (ii) to enter the result, thus giving more "width" to the ORU search (in the data space) than "depth". For instance, in Figure 4(b), it could be the case that none of the $L_2$ records belongs to $L_{upd}$, i.e., that the top-2-nd record comes from $L_1$ (i.e., $\boldsymbol{r}_2$ or $\boldsymbol{r}_4$) for any $\boldsymbol{v} \in C(\boldsymbol{r}_3)$.

## 5.3 Dropping Assumptions; Complete ORU

In this section, we describe our complete ORU algorithm. So far, we have made two impractical assumptions, i.e., that we have already computed the first $k$ upper hull layers, and that we know in advance the necessary radius $\rho$ to output $m$ records. Here we drop these assumptions; the first so that our algorithm is precomputation-free, and the second because it defies our problem formulation.

Without any precomputed layers or known $\rho$, our first step is to produce an overestimate of $\rho$, denoted as $\bar{\rho}$, which ensures an output size of at least $m$. That overestimate can be the radius required so that ORU's output for $k = 1$ includes $m$ records. This radius, for any $k$, is guaranteed to produce at least as many records.

A straightforward approach to derive $\bar{\rho}$ (based on $k = 1$) would be to compute the upper hull of the entire dataset $D$, get the $m$ top-regions with the smallest mindist to $\boldsymbol{w}$, and use the largest mindist among them as $\bar{\rho}$. That, however, may be too costly. Ideally, we would want to localize the upper hull computation to just the vicinity of $\boldsymbol{w}$. To achieve this, we exploit the fact that the $\rho$-skyline is a superset of ORU's output for the same $\rho$ and $k = 1$, as follows directly from the definition of $\rho$-dominance.

We use an *incremental* $\rho$-skyline algorithm, which supports "get next" calls to extend a $\rho$-skyline for the immediately larger $\rho$ around $\boldsymbol{w}$ that admits exactly one new record to it. Details on that technique (for its general, $\rho$-skyband version) are presented in Section 5.3.2. We initialize that algorithm and prompt it until the $\rho$-skyline includes $m$ records. Next, we compute their upper hull $L_{tmp}$. In general, not all $\rho$-skyline records will make it to $L_{tmp}$. If that is the case, we keep prompting the $\rho$-skyline algorithm, and updating the upper hull $L_{tmp}$ to cover the additional records, until $L_{tmp}$ includes $m$ extreme vertices (records). We use as $\bar{\rho}$ the final radius reported by the $\rho$-skyline algorithm. On top of this, note that the final $L_{tmp}$ is guaranteed to include all the part of layer $L_1$ that ORU processing could possibly need for an output of size $m$. In other words, the final $L_{tmp}$ can serve already as layer $L_1$ in subsequent processing.

Using the obtained $\bar{\rho}$, we compute the $\rho$-skyband (for the actual $k$ specified in the input). That can be done with a standard $k$-skyband algorithm by simply replacing regular dominance tests with $\bar{\rho}$-dominance ones (described in the last paragraph of Section 4.2). The derived $\bar{\rho}$-skyband is a guaranteed superset of the ORU output, thus we place its members into a *candidate set* $M$.

Even with $\bar{\rho}$ available, we have only an overestimate of the actual radius required to output $m$ records. Thus, computing upper hull layers directly on $M$ would be computationally wasteful. To circumvent this, and to also ensure an output of exactly $m$ records, we employ an advanced, adaptive technique which: (i) gradually expands $\rho$ around $\boldsymbol{w}$, moving from $\rho = 0$ towards $\bar{\rho}$, (ii) computes new upper hull layers on $M$ as and when needed only, and (iii) progressively outputs confirmed candidates (i.e., records guaranteed to be in the ORU result) while the search is ongoing. The latter property enables the tightening of $\bar{\rho}$ on the fly, and hence the shrinking of the $\bar{\rho}$-skyband and the elimination of candidates from $M$, so that the layer (i.e., upper hull) computations execute on increasingly fewer records. This improved implementation is presented next.

### 5.3.1 *Gradually expanding $\rho$.*

What we already have is the initial overestimate $\bar{\rho}$, (the necessary part of) layer $L_1$, and the candidate set $M$. Subsequent ORU search expands concurrently (i) in terms of radius, from 0 towards $\bar{\rho}$, and (ii) in terms of layer depth, computing on demand (the necessary parts of) deeper $L_i$ layers.

We treat the structure of all upper hull layers as an implicit tree, and apply the *best-first* approach to gradually explore that tree in increasing distance from the seed $\boldsymbol{w}$. In particular, we maintain a min-heap $Q$ that organizes known top-$i$ results (for $i \le k$) and their respective preference regions $C$, with mindist to $\boldsymbol{w}$ as their key. Let $\boldsymbol{r}$ be the top record according to $\boldsymbol{w}$. We start with the top-region of $\boldsymbol{r}$ (i.e., $C(\boldsymbol{r})$), whose mindist is by definition 0. First, we partition $C(\boldsymbol{r})$ according to the possible top-2-nd records, which requires computing $L_2$ (i.e., the upper hull of record set $M - L_1$) and applying Theorem 1, as demonstrated in Section 5.2. Each produced partition is pushed into $Q$ with key equal to its mindist to $\boldsymbol{w}$, and associated with its (now known) top-2 result. Second, for each $L_1$ record $\boldsymbol{r}_i$ that is adjacent to $\boldsymbol{r}$, we push its top-region $C(\boldsymbol{r}_i)$ into $Q$ (with $\{\boldsymbol{r}_i\}$ as its top-1 result). Then, we iteratively pop the heap. For each region $C$ popped from $Q$, we distinguish two cases:

**Case 1:** If $C$ corresponds to a top-$i$ result (with $i < k$), we partition it according to the different top-$(i+1)$-th records in $C$ (using Theorem 1, as in Section 5.2), and push the produced partitions into the heap (associated with their, now known, top-$(i+1)$ results). Applying Theorem 1 might require computing a new upper hull layer on the candidate set; details and an optimization are discussed later

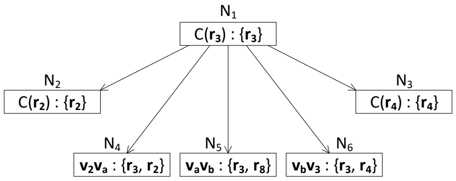



**Figure 5: Implicit tree**

in this section. Importantly, if $C$ corresponds to a top-1 result $\{\boldsymbol{r}_i\}$, we additionally push into the heap the top-regions of its adjacent records (omitting any that were pushed into $Q$ previously, to avoid duplication). The reason is that, unlike best-first search in an actual tree, at the "root level" of our implicit structure, we initially did not push into $Q$ the top-regions of all $L_1$ records, but only those neighboring $C(\boldsymbol{r})$. That was in order to save mindist calculations and unnecessary push operations, since many $L_1$ top-regions may lie too far from $\boldsymbol{w}$ to affect ORU processing. Instead, we use the continuity implied by Lemmas 1 and 2 to gradually push top-regions from $L_1$ into $Q$, only when one of their neighbors is popped.

**Case 2:** If $C$ corresponds to a top-$k$ result, it is considered *finalized*. That is, the top-$k$ result and its respective region $C$ are appended to the ORU output. Observe that, as we explained in Section 3, our algorithm goes beyond Definition 2, to output not only records, but also specific order-sensitive top-$k$ results, together with the preference regions that produce them.

The process terminates when the output includes $m$ distinct records. The mindist of the last finalized region is the minimum radius $\rho$ that appears in Definition 2. In a nutshell, our ORU methodology explores (i.e., partitions or finalizes, for $i < k$ and for $i = k$, respectively) regions $C$ in increasing distance from $\boldsymbol{w}$, utilizing the implicit tree structure to dismiss those too distant to affect the result.

Figure 5 shows the implicit tree for our running example. Each node $N_j$ represents a preference region (i.e., a segment, for $d = 2$) and its respective top-$i$ result. The root corresponds to the $L_1$ top-region that includes $\boldsymbol{w}$, i.e., $C(\boldsymbol{r}_3)$. First, we partition $C(\boldsymbol{r}_3)$ into 3 regions, namely, $\boldsymbol{v}_2\boldsymbol{v}_a$, $\boldsymbol{v}_a\boldsymbol{v}_b$, $\boldsymbol{v}_b\boldsymbol{v}_3$, as demonstrated in Figure 4. Associated with their top-2 results, they conceptually form nodes $N_4, N_5, N_6$, and are pushed into $Q$. We also push the top-regions of $L_1$ records adjacent to $\boldsymbol{r}_3$, i.e., $C(\boldsymbol{r}_2), C(\boldsymbol{r}_4)$, associated with their top-1 results (nodes $N_2, N_3$). Then, iterative popping commences.

The first popped node is $N_5$ (with mindist 0, since it actually includes $\boldsymbol{w}$), whose top-2 result is $\{\boldsymbol{r}_3, \boldsymbol{r}_8\}$. If $k = 2$, it is finalized and its top-$k$ records are output directly (Case 2). Popping continues with $N_4$ and $N_6$, which are finalized too; their top-$k$ results contribute two new records to the output (i.e., $\boldsymbol{r}_2$ and $\boldsymbol{r}_4$). If $m = 4$, the process terminates. Otherwise, the next popped node is $N_2$, for which we know the top-1 result (i.e., $N_2$ falls under Case 1). We will need to partition it by Theorem 1, and push into $Q$ its resulting "children" (not illustrated). Importantly, since $N_2$ belongs to $L_1$, we will also need to push its neighboring top-regions that were not encountered before, i.e., $C(\boldsymbol{r}_1)$ (not illustrated). The implicit tree is constructed gradually, with new nodes formed for each Case 1 pop.

An important point on Case 1 is that partitioning $C$ according to its different top-$(i+1)$-th records might require computing a new upper hull layer. That is, if layer $L_{t+1}$ (referring to the value of $t$ in Theorem 1 for $C$) was not previously computed, we need to compute it now. A naïve approach is to simply remove the first $t$ layers from the candidate set $M$ and compute the upper hull on the remaining candidates. An improvement however is possible, by shrinking $M$. Recall that our initial $\bar{\rho}$ estimation assumed we needed $m$ records in the $\bar{\rho}$-skyline. Letting $\tau$ be the number of distinct records which (i) belong to the top-$k$ results already finalized, and (ii) are not members of the $\bar{\rho}$-skyline, we can roll back the incremental $\rho$-skyline computation so that it includes only $m - \tau$ records. This backtracking effectively reduces $\bar{\rho}$, and in turn enables the shrinking of $M$ to only keep $\rho$-skyband records (for the actual $k$ input) for the reduced $\bar{\rho}$. The shrinking can be done trivially if we record the inflection radius for each record in the $\rho$-skyline during the original $\bar{\rho}$ estimation, and for each record in the initial candidate set (i.e., in the $\rho$-skyband for the original $\bar{\rho}$ estimate).

A concluding remark is that our ORU methodology is certain not to overshoot the target output size, i.e., it reports exactly $m$ records. To see this, as we explore the preference regions in increasing order of distance to $\boldsymbol{w}$, any newly finalized region $C$ is guaranteed to share a facet with a previously finalized region. The hyper-plane that defines this facet corresponds to exactly one order swap between two records; either an order swap between members of the top-$k$ result, or a replacement of the top-$k$-th record by a non-result one. This means that for each newly finalized region (with the exception of the very first), there is a maximum of one new record added to the ORU result. Thus, we may stop when we have output exactly $m$.

5.3.2 *Incremental $\rho$-skyband.* The OSS property and $m$ being dictated by the user/application is a central point of our motivation, and thus a hard requirement for our operators. However, the ORU algorithm requires as a building block an incremental $\rho$-skyband module (IRD). While ORU requires this for the $\rho$-skyline only (i.e., $k = 1$), here we address the arbitrary $k$ version, for generality.

The IRD challenge is that, unlike ORD, no $\rho$-dominance can ever be used to narrow down the search, because every $k$-skyband member may be output after a sufficient number of "get next" calls. Thus, the key question is how to serve these calls without computing the entire $k$-skyband. The main idea is to progressively fetch $k$-skyband members, but only output them when their inflection radius is no larger than a gradually growing threshold $\underline{\rho}$, introduced later.

IRD invokes a regular $k$-skyband algorithm to progressively fetch its members, in decreasing score order for $\boldsymbol{w}$. We use BBS [58] for that building block, but amend its default record/index node visiting order to order by score, as we did in Section 4.2. Let set $T$ hold the $k$-skyband records fetched by BBS so far. As ensured by the score-based fetching order, we can compute the exact inflection radius for each record in $T$ at the time it was fetched.

An invariant of branch-and-bound algorithms, like BBS, is that at any point during execution, their heap contents (records and index nodes) represent the not-yet-considered part of the dataset. Let $S$ be the set of all records and nodes currently in the heap. For simplicity, we extend notation $\boldsymbol{r}_i$ to nodes too, since BBS anyway represents them by the top corner of their minimum bounding box. For each $\boldsymbol{r}_i \in S$ we can compute an inflection radius $\underline{\rho_i}$ based on the current set $T$. However, that $\underline{\rho_i}$ is just a lower bound of the actual inflection radius, because BBS may have not yet fetched in $T$ all the $k$-skyband records with score larger than $U_{\boldsymbol{w}}(\boldsymbol{r}_i)$, i.e., $T$ may currently not include all records that $\rho$-dominate $\boldsymbol{r}_i$.

Since $S$ serves as a representation of all unexplored records, the minimum $\underline{\rho_i}$ among the members of $S$ serves as an overall lower bound $\underline{\rho}$ for any non-fetched record. Therefore, every record in $T$ with inflection radius no greater than $\underline{\rho}$ has a confirmed order in the output of IRD. In other words, these records are guaranteed to comprise the $\rho$-skyband for radius $\underline{\rho}$.

Consider Figure 2(b), where $k = 5$. When IRD is first invoked, we execute BBS to progressively fetch the first $k$ records, i.e., $\boldsymbol{r}_1$ to $\boldsymbol{r}_5$, which by definition are the top-$k$. They are placed into set $T$ and also output directly by IRD. When prompted with a "get next" call, IRD resumes BBS to progressively fetch new $k$-skyband members into $T$. Whenever BBS fetches a new record, we update $\underline{\rho}$ according to the current contents of the BBS heap (i.e., of set $S$). Let $\boldsymbol{r}_i$ be the not-yet-output record in $T$ with the smallest inflection radius. If $\rho_i \leq \underline{\rho}$, IRD outputs $\boldsymbol{r}_i$ and pauses. Otherwise, we keep fetching new records by BBS and updating $\underline{\rho}$ after each retrieval, until $\underline{\rho}$ becomes at least as large as the inflection radius of one record in $T$. That record is output by IRD as the next $\rho$-skyband member. Subsequent "get next" calls are served by resuming this process.

Returning to our example, consider that IRD receives a "get next" call (after its initialization, which reports the top-5 records all at once). It resumes BBS, but as it fetches $\boldsymbol{r}_6$, $\boldsymbol{r}_7$, $\boldsymbol{r}_8$ into $T$, assume that $\underline{\rho}$ does not become as large as any of the inflection radii in $T$ after any of these retrievals. However, when $\boldsymbol{r}_9$ is fetched too, suppose that the updated $\underline{\rho}$ becomes greater than (or equal to) $\rho_7$. IRD outputs $\boldsymbol{r}_7$ and pauses (until it receives another "get next" call).

## 6 EXPERIMENTS

In this section, we present qualitative and performance experiments. We use both real and synthetic datasets, i.e., *HOTEL*, *HOUSE*, *NBA*, and *ANTI*, *COR*, *IND*, respectively. *HOTEL* contains 418,843 hotel records with $d = 4$ attributes [1]. *HOUSE* includes 315,265 records of $d = 6$ types of household expenses [2]. *NBA* holds $d = 8$ statistics for 21,960 NBA players [3]. The synthetic datasets represent typical distributions in multi-objective decisions [14]. Table 2 lists the problem parameters with their tested and default values (in bold). In each experiment, we vary one parameter and fix the others to their defaults. Every measurement is the average for 50 random seeds $\boldsymbol{w}$. The datasets are indexed by R-trees and kept in memory. That said, our methods are applicable to disk-based storage too. All algorithms were implemented in C++ and run on a machine with Intel i7-7700 CPU at 3.60Ghz and 32Gb RAM. We used QuadProg++ [26] as the quadratic programming solver, and Qhull [9] for computational geometric primitives. Our implementation is available at [4].

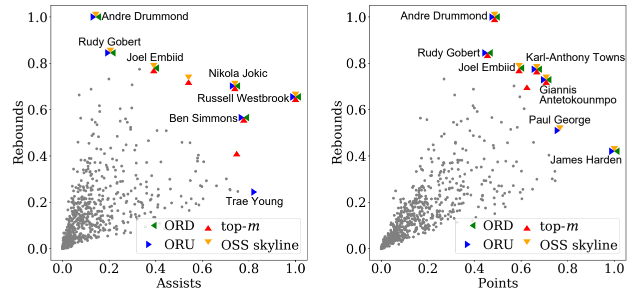
| Parameter | Tested and default values |
|---|---|
| Dataset cardinality $\lvert D \rvert$ | 100K, **400K**, 1.6M, 6.4M, 25.6M |
| Dimensionality $d$ | 2, 3, **4**, 5, 6, 7 |
| Parameter $k$ | 1, **5**, 10, 15, 20 |
| Output size $m$ | 10, 30, **50**, 70, 90 |

**Table 2: Parameters, tested values, and defaults**

We start with qualitative results to draw a distinction between our operators and representative previous ones from Table 1.

### 6.1 Qualitative Study

First, we perform a case study to visualize how the results of ORD and ORU differ from (i) an OSS skyline, and (ii) a top-$m$ query for



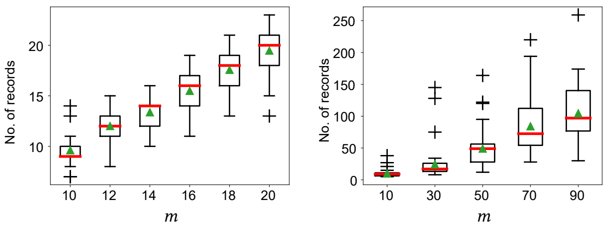

(a) Assists-Rebounds, $\boldsymbol{w} = (0.49, 0.51)$ (b) Points-Rebounds, $\boldsymbol{w} = (0.43, 0.57)$

**Figure 6: Case study on NBA 2018-19 statistics ($k = 2, m = 6$)**

the same vector $\boldsymbol{w}$. The former reports the $m$ skyline members that dominate the most non-skyline records, following the most cited full-dimensionality OSS skyline definition [49]. We use the NBA 2018-19 season statistics for the total of its 708 players on Assists and Rebounds (in Figure 6(a)), and Points and Rebounds (in Figure 6(b)), normalized in the $[0, 1]$ range. We set $k = 2$, $m = 6$, and use $(0.49, 0.51)$ and $(0.43, 0.57)$ as the seed $\boldsymbol{w}$, respectively. The results of the methods are illustrated as differently oriented/colored triangles.

A first observation is that our operators report distinct results from past approaches (and from each other). For example, in Figure 6(a) only ORU reports *Trae Young* (rising stars challenge player), while half of its output records are not in the top-$m$ result, and one third of them are not in the OSS skyline. The comparison of our operators with top-$m$ reveals an even more interesting fact. In Figure 6(a), top-$m$ misses *Andre Drummond* (the rebound leader in 2018-19 season), whom it would report if we only slightly revised $\boldsymbol{w}$ from $(0.49, 0.51)$ to $(0.48, 0.52)$. Similarly, in Figure 6(b), top-$m$ misses *James Harden* (the season's scoring leader), whom it would report if we revised $\boldsymbol{w}$ from $(0.43, 0.57)$ to $(0.44, 0.56)$. The inclusion of these players in the result of both our operators (in the respective Figure 6(a) and 6(b) settings) confirms that they successfully employ some "width" in their search, by reporting records that are particularly strong for alternative, very similar preferences to the seed $\boldsymbol{w}$. Investigations in our full-scale experimental settings, using the Jaccard coefficient as the similarity measure (charts omitted in the interest of space), demonstrate that (i) OSS skyline is more dissimilar to our operators than top-$m$, as it is not guided by $\boldsymbol{w}$, and (ii) top-$m$ becomes increasingly more dissimilar to ORD/ORU as $m$ grows. As an indication for the omitted charts, for *IND* data and the default parameters in Table 2, the Jaccard similarity of OSS skyline to ORD is 0.25, and to ORU it is 0.24. In the same setting, the Jaccard similarity of top-$m$ to ORD is 0.44, and to ORU it is 0.32.

Next, we use real TripAdvisor data and reviews (*TA*) [5]. *TA* includes ratings for 1,850 hotels on $d = 7$ aspects (value, room, location, etc), forming dataset $D$. It also includes actual user reviews for these hotels, each comprising a comment and an overall score. The reviews offer a practical example of how preference vectors could be extracted for real users. Specifically, we employ [70], an established preference mining method that estimates a user's weight vector based on her reviews. Applying [70], we get vectors for 137,563 users. The dataset and vectors from *TA* are used in the next experiment on fixed-region methods [20, 54], but at the same time they offer an end-to-end application example for our techniques.



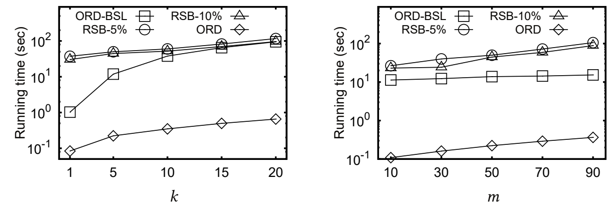

(a) Output size variability in *TA* (b) Output size variability in *IND*

**Figure 7: Output size analysis of fixed-region techniques**

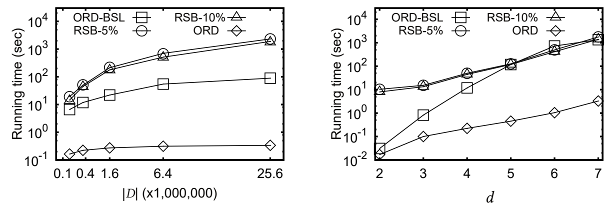
The fixed-region methods require a preference polytope $R$ to be specified as part of their input. We demonstrate that it is not feasible to estimate the size of $R$ required to produce, even approximately, $m$ records. Worse yet, even the same polytope $R$, when positioned at different parts of the preference domain, can produce vastly different output sizes. Specifically, we use $k = 5$ and vary $m$ from 10 to 20 (since the dataset in *TA* is highly correlated and its 5-skyband includes only 61 hotels). We first execute ORU for 50 randomly selected *TA* users (preference vectors) and record the average stopping radius, denoted as $\rho^*$. We then produce a hypercube $R$ with volume equal to the hyper-sphere with radius $\rho^*$. Next, we count the number of distinct records (*TA* hotels) output by the fixed-region top-$k$ operator in [54] when $R$ is positioned around each of the 50 user vectors (we use [54] since [20] works only for $k = 1$). Figure 7(a) presents in box-plot the observed variation in output size. To confirm, in Figure 7(b), we repeat this process for our full-scale setting, using random preference vectors, *IND* data, the default parameters, and standard range for $m$ from Table 2. The box-plots indicate that even with knowledge of $\rho^*$, fixed-region techniques can produce dramatically different output sizes, which may hugely under- or over-shoot the target $m$. In contrast, our operators relieve the user/application from the need to meddle with the preference domain's complex dynamics, and abide strictly by the requested output size $m$. Note that the counterpart of this experiment, using ORD to compute $\rho^*$ and producing the fixed-region $R$-skyband, demonstrates an even greater variability than Figure 7 for both *TA* and *IND* (charts omitted in the interest of space). For example, for target $m = 50$ in *IND* data, the fixed-region $R$-skyband outputs from 12 all the way to 269 records.

## 6.2 ORD **Performance**

In this section, we evaluate our ORD algorithm. For comparison, we adapted a fixed-region $R$-skyband technique (RSB), as described at the end of Section 2. In the adaptation, the initial hyper-cube $R$ is sized so that its volume ratio to the preference domain equals the ratio of $m$ to the expected $k$-skyband cardinality (i.e., $\frac{k \ln^{d-1} n}{d!}$, according to [30]). Based on the size of the $R$-skyband computed for the initial $R$, its volume is re-estimated proportionally to the desired $m$. The trials ($R$-skyband computation and $R$ re-estimation) are repeated until the output is within 5% or 10% from $m$, resulting in two RSB versions. For the implementation of RSB, we use the index-based $R$-skyband module of [54], as it is considerably faster than the no-index approach of [20]. We also include baseline ORD-BSL that computes the entire $k$-skyband according to Section 4.1, without the enhancements in Section 4.2.

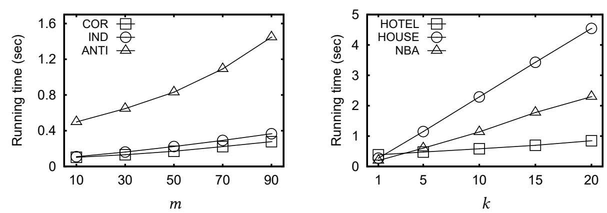



(a) Effect of $k$ (b) Effect of $m$




(c) Effect of $|D|$ (d) Effect of $d$

**Figure 8:** ORD **versus adaptations and baseline**




(a) Different data distributions, varying $m$ (b) Real datasets, varying $k$

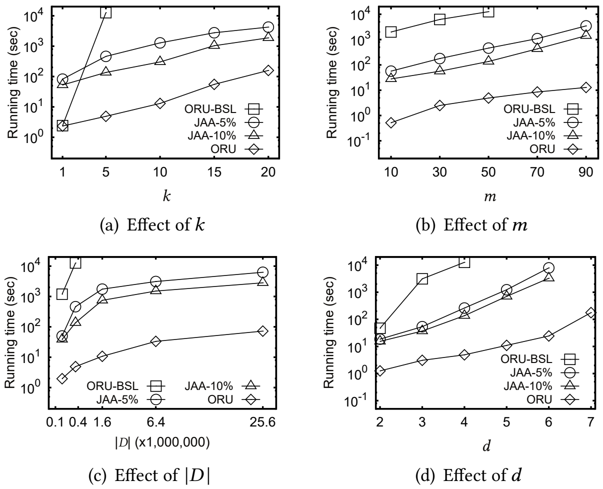
**Figure 9:** ORD **on different distributions and real datasets**

In Figure 8, we present (in logarithmic scale) the running time for all competitors versus each problem parameter, using *IND* data. ORD is 2 to 4 orders of magnitude faster than both versions of RSB, indicating the impracticality of fixed-region approaches to mimic its output, despite the ample slack given. The main reason is the numerous trials required for RSB to "converge" to an acceptable deviation from $m$. The runner-up is ORD-BSL, which is 1 to 2 orders of magnitude slower than the fully-enhanced ORD. The results also demonstrate ORD's ability to scale. Its running time grows almost linearly to $k$ and $m$, and sub-linearly to $|D|$. It increases more sharply with $d$, due to the growing complexity of its geometric building blocks. Still, even for $d = 7$ ORD takes only 3.2s.

Having established the superiority of our ORD algorithm, in Figure 9(a) we plot its running time for different synthetic distributions. Processing in *ANTI* is the slowest. The reason is that dominance (and, by deduction, $\rho$-dominance too) is less frequent among *ANTI* records, thus many candidates need to be considered before ORD can terminate. *COR* exhibits the inverse effect. In Figure 9(b), we use real data and vary $k$. *HOTEL* and *HOUSE* have comparable size, yet ORD is faster on *HOTEL*, due to its smaller dimensionality. *NBA* has the smallest cardinality, but the largest dimensionality, which explains why its line is in between the other two.

## 6.3 ORU **Performance**

Turning to ORU, we use for comparison an adaptation of the most efficient fixed-region top-$k$ algorithm in [54], termed JAA. We employ a similar $R$ estimation and trial approach as for RSB, allowing




(a) Effect of $k$

(b) Effect of $m$

(c) Effect of $|D|$

(d) Effect of $d$

**Figure 10:** ORU **versus adaptations and baseline**

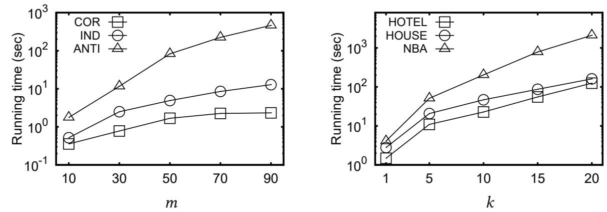
a deviation of up to 5% or 10% from the desired output size $m$. We also include ORU-BSL, a baseline that utilizes the initial overestimate $\bar{\rho}$ in Section 5.3, computes upper hull layers on the entire candidate set $M$, partitions the $L_1$ top-regions by Theorem 1, and reports the $m$-sized union of top-$k$ records for the closest produced regions to $\boldsymbol{w}$.

In Figure 10, we plot the running time of all competitors versus each problem parameter, in logarithmic scale. ORU-BSL, although identical to ORU for $k = 1$, generally performs very poorly, failing to terminate within reasonable time in most large settings. This demonstrates the vital role of our gradual expansion in terms of both $\rho$ and layer depth, presented in Section 5.3.1. Indeed, the full-fledged ORU is 2 to 4 orders of magnitude faster than ORU-BSL. ORU is also 12 to 134 times faster than JAA-10%, and even more compared to JAA-5%, confirming the general unsuitability of fixed-region approaches to our problems. Regarding ORU itself, it scales well with all parameters. Its running time increases faster than ORD because, despite the similarities in their definition, the nature of ORU is significantly more complex. To intuitively see this, determining whether a record $\boldsymbol{r}_i$ belongs to the $\rho$-skyband of a dataset, we need a $\rho$-dominance test per competitor at worst. In contrast, to determine whether a record $\boldsymbol{r}_i$ could appear in the top-$k$ result for any vector $\boldsymbol{v}$ within radius $\rho$ from $\boldsymbol{w}$, we need to consider multiple combinations of competitors that may outscore $\boldsymbol{r}_i$, and different $\boldsymbol{v}$ possibilities within distance $\rho$ from $\boldsymbol{w}$.

Focusing on our ORU algorithm, in Figure 11(a), we try it on different synthetic data distributions. As expected, the problem is the toughest in *ANTI*, and the easiest in *COR*, for the reason explained in Figure 9(a). In Figure 11(b), we execute ORU on real datasets. Unlike Figure 9(b), processing in *NBA* is slower than *HOUSE*. The reason is that *NBA*'s higher dimensionality affects ORU's performance more than ORD's. ORU has a stronger geometric nature and a greater reliance on computational geometric primitives (such as upper hull computation), whose complexity grows with $d$.

## 6.4 Discussion

In our default setting, for 400K *IND* records ORU requires 4.9s, while for 25.6M records it takes 72s. On the other hand, ORD runs




(a) Different data distributions, varying $m$

(b) Real datasets, varying $k$

**Figure 11:** ORU **on different distributions and real datasets**

in less than 1s in both cases (0.22s and 0.34s, respectively). That is, ORD is ready even for applications that require sub-second responses, while ORU is not quite there. To employ ORU in such applications, a viable approach is parallelization. At the core of its execution, ORU partitions numerous regions $C$ under Case 1 in Section 5.3.1. With little synchronization effort, multiple regions could be (popped from ORU's min-heap and) partitioned in parallel, since determining the different top-$(i+1)$-th records in each of these regions is independent of the others. An orthogonal approach is to materialize
*Onion* technique [17], or to cache and reuse the partial upper hulls from past ORU queries, together with the respective top-regions. Note that this will benefit performance even if ORU needs to consider (and thus compute on the fly parts of) upper hulls deeper than the available ones. A more general direction would be to let go of order-sensitivity in the reported top-$k$ results (since it is anyway not required by Definition 2) or to switch to ORD processing if a region under Case 1 is deemed too small. For the former direction, algorithmic redesign will be necessary; for the latter, formal guarantees should be given to bound the induced deviation from the exact ORU answer. The last two directions could be promising for future work.

# 7 CONCLUSION

This paper draws motivation from the known weaknesses of standard skyline and top-$k$ queries. Based on these shortcomings, we identify three hard requirements for practical decision support in multi-objective settings: personalization; controllable output size; and flexibility in preference specification. We argue that no previous study has effectively satisfied all three requirements, and propose two new operators (ORD and ORU) to bridge that gap. Our qualitative analysis indicates that they offer a novel type of support, distinct from past practices. Also, our experiments demonstrate that our algorithms deliver practical and scalable performance. Future work could explore the directions listed in Section 6.4 and/or consider ORD/ORU in highly skewed or sparse datasets, where multi-objective querying may be meaningful in higher dimensions too.


# ACKNOWLEDGMENTS

Kyriakos Mouratidis was supported by the Singapore Management University Lee Kong Chian Fellowship. Bo Tang and Keming Li were supported by the National Science Foundation of China (NSFC No. 61802163), the Education Department of Guangdong (Grant No. 2020KZDZX1184), and the Guangdong Provincial Key Laboratory (Grant No. 2020B121201001).

## REFERENCES


[1] Hotel dataset. https://www.hotels-base.com.
[2] House dataset. https://www.ipums.org.
[3] NBA dataset. https://www.basketball-reference.com.
[4] ORD and ORU implementation. https://github.com/ghlkm/ordu.
[5] TripAdvisor dataset. https://www.cs.virginia.edu/~hw5x/dataset.html.
[6] L. Antova, T. Jansen, C. Koch, and D. Olteanu. Fast and simple relational processing of uncertain data. In *ICDE*, pages 983–992, 2008.
[7] A. Asudeh, A. Nazi, N. Zhang, and G. Das. Efficient computation of regret-ratio minimizing set: A compact maxima representative. In *SIGMOD Conference*, pages 821–834, 2017.
[8] A. Asudeh, A. Nazi, N. Zhang, G. Das, and H. V. Jagadish. RRR: rank-regret representative. In *SIGMOD Conference*, pages 263–280, 2019.
[9] C. B. Barber and H. Huhdanpaa. qhull. http://www.qhull.org.
[10] I. Bartolini, Z. Zhang, and D. Papadias. Collaborative filtering with personalized skylines. *IEEE Trans. Knowl. Data Eng.*, 23(2):190–203, 2011.
[11] N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger. The R*-tree: An efficient and robust access method for points and rectangles. In *SIGMOD Conference*, pages 322–331, 1990.
[12] J. L. Bentley, H. T. Kung, M. Schkolnick, and C. D. Thompson. On the average number of maxima in a set of vectors and applications. *J. ACM*, 25(4):536–543, 1978.
[13] M. d. Berg, O. Cheong, M. v. Kreveld, and M. Overmars. *Computational geometry: algorithms and applications*. Springer-Verlag TELOS, 2008.
[14] S. Börzsönyi, D. Kossmann, and K. Stocker. The skyline operator. In *ICDE*, pages 421–430, 2001.
[15] C. Y. Chan, H. V. Jagadish, K. Tan, A. K. H. Tung, and Z. Zhang. Finding k-dominant skylines in high dimensional space. In *SIGMOD Conference*, pages 503–514, 2006.
[16] C. Y. Chan, H. V. Jagadish, K. Tan, A. K. H. Tung, and Z. Zhang. On high dimensional skylines. In *EDBT*, pages 478–495, 2006.
[17] Y.-C. Chang, L. Bergman, V. Castelli, C.-S. Li, M.-L. Lo, and J. R. Smith. The onion technique: Indexing for linear optimization queries. In *SIGMOD Conference*, pages 391–402, 2000.
[18] S. Chester, A. Thomo, S. Venkatesh, and S. Whitesides. Computing k-regret minimizing sets. *PVLDB*, 7(5):389–400, 2014.
[19] J. Chomicki, P. Ciaccia, and N. Meneghetti. Skyline queries, front and back. *SIGMOD Rec.*, 42(3):6–18, 2013.
[20] P. Ciaccia and D. Martinenghi. Reconciling skyline and ranking queries. *PVLDB*, 10(11):1454–1465, 2017.
[21] G. Cormode, F. Li, and K. Yi. Semantics of ranking queries for probabilistic data and expected ranks. In *ICDE*, pages 305–316, 2009.
[22] G. Das, D. Gunopulos, N. Koudas, and N. Sarkas. Ad-hoc top-k query answering for data streams. In *VLDB*, pages 183–194, 2007.
[23] J. S. Dyer and R. K. Sarin. Measurable multiattribute value functions. *Oper. Res.*, 27(4):810–822, 1979.
[24] J. Fürnkranz and E. Hüllermeier, editors. *Preference Learning*. Springer, 2010.
[25] Y. Gao, Q. Liu, L. Chen, G. Chen, and Q. Li. Efficient algorithms for finding the most desirable skyline objects. *Knowl. Based Syst.*, 89:250–264, 2015.
[26] L. D. Gaspero. Quadprogpp. https://github.com/liuq/QuadProgpp.
[27] S. Ge, L. H. U, N. Mamoulis, and D. W. Cheung. Efficient all top-$k$ computation - A unified solution for all top-$k$, reverse top-$k$ and top-$m$ influential queries. *IEEE Trans. Knowl. Data Eng.*, 25(5):1015–1027, 2013.
[28] S. Ge, L. H. U, N. Mamoulis, and D. W. Cheung. Dominance relationship analysis with budget constraints. *Knowl. Inf. Syst.*, 42(2):409–440, 2015.
[29] A. M. Geoffrion, J. S. Dyer, and A. Feinberg. An interactive approach for multi-criterion optimization, with an application to the operation of an academic department. *Management Science*, 19(4-part-1):357–368, 1972.
[30] P. Godfrey, R. Shipley, and J. Gryz. Algorithms and analyses for maximal vector computation. *VLDB J.*, 16(1):5–28, 2007.
[31] D. Goldfarb and A. U. Idnani. A numerically stable dual method for solving strictly convex quadratic programs. *Math. Program.*, 27(1):1–33, 1983.
[32] J. Gross. Redefining Hick's law. SMASHING Magazine, 2012. https://www.smashingmagazine.com/2012/02/redefining-hicks-law/.
[33] X. Han, B. Wang, J. Li, and H. Gao. Ranking the big sky: efficient top-k skyline computation on massive data. *Knowl. Inf. Syst.*, 60(1):415–446, 2019.
[34] W. E. Hick. On the rate of gain of information. *Quarterly Journal of Experimental Psychology*, 4(1):11–26, 1952.
[35] C. Holst. Infinite scrolling, pagination or "load more" buttons? Usability findings in eCommerce. SMASHING Magazine, 2016. https://www.smashingmagazine.com/2016/03/pagination-infinite-scrolling-load-more-buttons/.
[36] V. Hristidis, N. Koudas, and Y. Papakonstantinou. PREFER: A system for the efficient execution of multi-parametric ranked queries. In *SIGMOD Conference*, pages 259–270, 2001.
[37] R. Hyman. Stimulus information as a determinant of reaction time. *Journal of Experimental Psychology*, 45(3):188–196, 1953.
[38] I. F. Ilyas, G. Beskales, and M. A. Soliman. A survey of top-k query processing techniques in relational database systems. *ACM Comp. Surveys*, 40(4):11:1–11:58, 2008.
[39] D. R. Insua and S. French. A framework for sensitivity analysis in discrete multi-objective decision-making. *Eur. J. Oper. Res.*, 54(2):176–190, 1991.
[40] K. G. Jamieson and R. D. Nowak. Active ranking using pairwise comparisons. In *NIPS*, pages 2240–2248, 2011.
[41] T. Joachims. Optimizing search engines using clickthrough data. In *KDD*, pages 133–142, 2002.
[42] R. L. Keeney and H. Raiffa. *Decisions with Multiple Objectives: Preferences and Value Trade-Offs*. Wiley, 1976.
[43] W. Kießling. Foundations of preferences in database systems. In *VLDB*, pages 311–322, 2002.
[44] M. Köksalan, J. Wallenius, and S. Zionts. *Multiple Criteria Decision Making: From Early History to the 21st Century*. World Scientific Publishing Co. Pte. Ltd., 2011.
[45] V. Koltun and C. H. Papadimitriou. Approximately dominating representatives. *Theor. Comput. Sci.*, 371(3):148–154, 2007.
[46] J. Lee and S. Hwang. Scalable skyline computation using a balanced pivot selection technique. *Inf. Syst.*, 39:1–21, 2014.
[47] J. Lee, G. You, and S. Hwang. Personalized top-k skyline queries in high-dimensional space. *Inf. Syst.*, 34(1):45–61, 2009.
[48] H. Li, T. N. Chan, M. L. Yiu, and N. Mamoulis. FEXIPRO: fast and exact inner product retrieval in recommender systems. In *SIGMOD Conference*, pages 835–850, 2017.
[49] X. Lin, Y. Yuan, Q. Zhang, and Y. Zhang. Selecting stars: The k most representative skyline operator. In *ICDE*, pages 86–95, 2007.
[50] M. Magnani, I. Assent, and M. L. Mortensen. Taking the big picture: representative skylines based on significance and diversity. *VLDB J.*, 23(5):795–815, 2014.
[51] J. Matousek and O. Schwarzkopf. Linear optimization queries. In *ACM Symposium on Computational Geometry*, pages 16–25, 1992.
[52] D. Mindolin and J. Chomicki. Discovering relative importance of skyline attributes. *PVLDB*, 2(1):610–621, 2009.
[53] R. D. C. Monteiro and I. Adler. Interior path following primal-dual algorithms. part II: convex quadratic programming. *Math. Program.*, 44(1-3):43–66, 1989.
[54] K. Mouratidis and B. Tang. Exact processing of uncertain top-k queries in multi-criteria settings. *PVLDB*, 11(8):866–879, 2018.
[55] K. Mouratidis, J. Zhang, and H. Pang. Maximum rank query. *PVLDB*, 8(12):1554–1565, 2015.
[56] D. Nanongkai, A. Lall, A. D. Sarma, and K. Makino. Interactive regret minimization. In *SIGMOD Conference*, pages 109–120, 2012.
[57] D. Nanongkai, A. D. Sarma, A. Lall, R. J. Lipton, and J. J. Xu. Regret-minimizing representative databases. *PVLDB*, 3(1):1114–1124, 2010.
[58] D. Papadias, Y. Tao, G. Fu, and B. Seeger. Progressive skyline computation in database systems. *ACM Trans. Database Syst.*, 30(1):41–82, 2005.
[59] J. Pei, Y. Yuan, X. Lin, W. Jin, M. Ester, Q. Liu, W. Wang, Y. Tao, J. X. Yu, and Q. Zhang. Towards multidimensional subspace skyline analysis. *ACM Trans. Database Syst.*, 31(4):1335–1381, 2006.
[60] L. Qian, J. Gao, and H. V. Jagadish. Learning user preferences by adaptive pairwise comparison. *PVLDB*, 8(11):1322–1333, 2015.
[61] A. D. Sarma, A. Lall, D. Nanongkai, R. J. Lipton, and J. J. Xu. Representative skylines using threshold-based preference distributions. In *ICDE*, pages 387–398, 2011.
[62] T. K. Sellis, N. Roussopoulos, and C. Faloutsos. The R+-tree: A dynamic index for multi-dimensional objects. In *VLDB*, pages 507–518, 1987.
[63] M. A. Soliman, I. F. Ilyas, D. Martinenghi, and M. Tagliasacchi. Ranking with uncertain scoring functions: semantics and sensitivity measures. In *SIGMOD Conference*, pages 805–816, 2011.
[64] R. E. Steuer. *Multiple Criteria Optimization: Theory, Computation and Application*. Wiley, 1986.
[65] Y. Tao, L. Ding, X. Lin, and J. Pei. Distance-based representative skyline. In *ICDE*, pages 892–903, 2009.
[66] Y. Tao, V. Hristidis, D. Papadias, and Y. Papakonstantinou. Branch-and-bound processing of ranked queries. *Inf. Syst.*, 32(3):424–445, 2007.
[67] Y. Tao, X. Xiao, and J. Pei. Efficient skyline and top-k retrieval in subspaces. *IEEE Trans. Knowl. Data Eng.*, 19(8):1072–1088, 2007.
[68] E. Tiakas, A. N. Papadopoulos, and Y. Manolopoulos. Progressive processing of subspace dominating queries. *VLDB J.*, 20(6):921–948, 2011.
[69] A. Vlachou and M. Vazirgiannis. Ranking the sky: Discovering the importance of skyline points through subspace dominance relationships. *Data Knowl. Eng.*, 69(9):943–964, 2010.
[70] H. Wang, Y. Lu, and C. Zhai. Latent aspect rating analysis on review text data: a rating regression approach. In *KDD*, pages 783–792, 2010.
[71] H. Wang, Y. Lu, and C. Zhai. Latent aspect rating analysis without aspect keyword supervision. In *KDD*, pages 618–626, 2011.
[72] T. Xia, D. Zhang, and Y. Tao. On skylining with flexible dominance relation. In *ICDE*, pages 1397–1399, 2008.
[73] M. Xie, R. C. Wong, and A. Lall. Strongly truthful interactive regret minimization. In *SIGMOD Conference*, pages 281–298, 2019.

[74] M. Xie, R. C. Wong, and A. Lall. An experimental survey of regret minimization query and variants: bridging the best worlds between top-k query and skyline query. *VLDB J.*, 29(1):147–175, 2020.
[75] M. Xie, R. C. Wong, J. Li, C. Long, and A. Lall. Efficient k-regret query algorithm with restriction-free bound for any dimensionality. In *SIGMOD Conference*, pages 959–974, 2018.
[76] K. Yi, F. Li, G. Kollios, and D. Srivastava. Efficient processing of top-k queries in uncertain databases with x-relations. *IEEE Trans. Knowl. Data Eng.*, 20(12):1669–1682, 2008.
[77] M. L. Yiu and N. Mamoulis. Multi-dimensional top-$k$ dominating queries. *VLDB J.*, 18(3):695–718, 2009.
[78] A. Yu, P. K. Agarwal, and J. Yang. Top-k preferences in high dimensions. *IEEE Trans. Knowl. Data Eng.*, 28(2):311–325, 2016.
[79] S. Zeighami and R. C. Wong. Finding average regret ratio minimizing set in database. In *ICDE*, pages 1722–1725, 2019.
[80] J. Zhang, K. Mouratidis, and H. Pang. Global immutable region computation. In *SIGMOD Conference*, pages 1151–1162, 2014.
[81] J. Zheng and C. Chen. Sorting-based interactive regret minimization. In *APWeb-WAIM*, pages 473–490, 2020.