Creating Top Ranking Options in the Continuous Option and Preference Space

Bo Tang[†] Kyriakos Mouratidis[‡] Man Lung Yiu[§] Zhenyu Chen[†]
[†] Southern University of Science and Technology, {tangb3@,11510237@mail.}sustc.edu.cn
[‡] Singapore Management University, kyriakos@smu.edu.sg
[§] The Hong Kong Polytechnic University, csmlyiu@comp.polyu.edu.hk

ABSTRACT

Top-k queries are extensively used to retrieve the k most relevant options (e.g., products, services, accommodation alternatives, etc) based on a weighted scoring function that captures user preferences. In this paper, we take the viewpoint of a business owner who plans to introduce a new option to the market, with a certain type of clientele in mind. Given a target region in the consumer spectrum, we determine what attribute values the new option should have, so that it ranks among the top-k for any user in that region. Our methodology can also be used to improve an existing option, at the minimum modification cost, so that it ranks consistently high for an intended type of customers. This is the first work on competitive option placement where no distinct user(s) are targeted, but a general clientele type, i.e., a continuum of possible preferences. Here also lies our main challenge (and contribution), i.e., dealing with the interplay between two continuous spaces: the targeted region in the preference spectrum, and the option domain (where the new option will be placed). At the core of our methodology lies a novel and powerful interlinking between the two spaces. Our algorithms offer exact answers in practical response times, even for the largest of the standard benchmark datasets.

PVLDB Reference Format:
Bo Tang, Kyriakos Mouratidis, Man Lung Yiu and Zhenyu Chen. Creating Top Ranking Options in the Continuous Option and Preference Space. *PVLDB*, 12(10): xxxx-yyyy, 2019.
DOI: https://doi.org/110.14778/3339490.3339500

1. INTRODUCTION

In today's digital world, there is instant availability of numerous alternatives to cover the consumers' daily needs, like purchasing a product, finding a restaurant, booking accommodation, etc. Choosing among available options (e.g., laptops) often involves multiple, possibly conflicting criteria, such as speed, weight, and cost. To support such multi-criteria decisions in a personalized manner and at scale, the top-k query has emerged as the standard tool [23]. The user's preferences are expressed by specifying a weight for each criterion to indicate its significance. These weights implicitly associate each available option with a numeric score, i.e., the weighted sum of its attribute values. The k highest-scoring options are presented to the user. This corresponds to a *linear* top-k query, which is the most common in the literature, and whose linear scoring is shown to capture closely the way real users make decisions [37].

Assuming a setting where users browse options via such top-k queries, we take the standpoint of a business owner, and suggest optimal placement strategies to maximize the competitiveness of her product or service. Consider a computer manufacturer who is planning to introduce a new laptop to the market, targeting a specific type of users (e.g., business users in need of moderately powerful, yet highly portable laptops). In this work, we determine what attribute values the laptop should have, so that it ranks consistently high for any user in the targeted part of the consumer spectrum.

More formally, preferences in the top-k model are expressed by a weight value per criterion (i.e., option attribute), together comprising a weight vector $\boldsymbol{w}$. We refer to the domain of $\boldsymbol{w}$ as the *preference space*. Each option in the market can be viewed as a point $\boldsymbol{p}_i$ in a d-dimensional *option space*, where every axis corresponds to an attribute. The targeted type of users is represented by a region wR in preference space; choosing a target clientele is a business decision supported by demographic information (e.g., income, profession, and population statistics [39, 53]) that is orthogonal to our processing framework. Given wR, we compute the region in option space where an option $\boldsymbol{o}$ should be, so that it is in the top-k set for any weight vector in wR. We call this the *top-ranking region* problem (TopRR) and denote the computed region as oR. Note that oR must be maximal, meaning that it includes all possible option placements that would rank among the top-k for every preference in wR.

TopRR enables data-driven tools for market impact maximization. Consider the creation of a new product (option) $\boldsymbol{o}$. Placing $\boldsymbol{o}$ in oR guarantees its presence in the top-k result for any user of the targeted type. Moreover, if manufacturing cost is also a concern, TopRR enables the cost-optimal placement of $\boldsymbol{o}$. Manufacturing cost is typically monotonic to $\boldsymbol{o}$'s attribute values [43], e.g., proportional to their summed squares. In this scenario, we may compute the cost-optimal $\boldsymbol{o}$ using region oR (which we prove to be a convex polytope) as the constraint to a quadratic programming solver [29]. In case the business owner's profitability/manufacturing cost function is more complex, convex (or other types of mathematical) optimization [38] could be used to derive the cost-optimal $\boldsymbol{o}$ in oR.

In addition to option creation, TopRR also supports the optimal modification (enhancement) of an existing option $\boldsymbol{p}_i$. Consider a computer manufacturer who is revamping an existing laptop model (option) $\boldsymbol{p}_i$ so that it becomes highly appealing to a target customer segment, represented by region wR in preference space. On the requirement that the revamped version $\boldsymbol{p}'_i$ is among the top-k for every weight vector in wR, TopRR produces the region oR in option space where $\boldsymbol{p}'_i$ should lie. Furthermore, if the option modification

Proceedings of the VLDB Endowment, Vol. 12, No. 10
ISSN 2150-8097.
DOI: https://doi.org/110.14778/3339490.3339500

cost is proportional to the Euclidean distance between $\boldsymbol{p}_i$ and $\boldsymbol{p}'_i$ (as commonly assumed [33, 26]), the cost-optimal $\boldsymbol{p}'_i$ can be identified via quadratic programming with oR as the constraint.

Option creation and impact maximization problems have been considered in the context of dominance/skyline operator (e.g., [25, 47, 27]), but such formulations are inapplicable to score-based ranking and are not configurable to specific user types. Option creation and enhancement has also been studied in the context of top-k queries, yet for finite sets of specific users/weight vectors [26, 50, 49]. Our problem is fundamentally different, because no specific individual users (or distinct weight vectors) need to be known in advance. Instead, a generic clientele type is specified, by means of a region wR that includes infinite possible weight vectors. Some recent studies do consider the (continuous) preference space of the top-k query (e.g., [35, 41]), but none of them deals with the creation or improvement of an option, i.e., all options are taken as is, and their attributes are not to be decided or modified. The very need in TopRR to deal concurrently with two continuous domains (i.e., the option space and the preference space) is our major challenge, and contribution. In summary, our contributions are as follows:

- We introduce TopRR, which enables data-driven market impact maximization for options browsed by top-k queries.
- We make crucial observations that lead to a novel and insightful interlinking of the two involved continuous domains, i.e., the option space and the preference space.
- Based on that interlinking, we codify the conditions that define oR (i.e., the output of TopRR), and develop a suite of algorithms to exactly and efficiently compute it.
- We present a case study using real data, and empirically demonstrate the efficiency of our methods, which take seconds even for the largest benchmark datasets, outperforming by up to two orders of magnitude a baseline constructed from previous work.

2. RELATED WORK

The top-k query reports the k options in a dataset that maximize a scoring function. The most prevalent type of scoring function is linear, where the user provides a weight per data attribute, and the score of an option is the weighted sum of its attribute values [11, 22]. The literature focuses primarily on low-dimensional cases (a few attributes), as ranking by score generally becomes meaningless in high dimensions [52, 31]. Common processing approaches employ thresholding (on options organized by sorted lists) [17], the branch-and-bound paradigm (on options indexed by spatial access methods) [42], etc. A comprehensive survey is [23].

2.1 Considering Specific Weight Vectors

The proliferation of the top-k query has led to definitions of market impact, as well as option creation/improvement strategies, centered on it. In this subsection, we review such definitions, which however are customized for finite sets of specific weight vectors.

In [44], Vlachou et al. introduce the *reverse top-k* query. Given a set of options $\mathcal{D}$ and a set of weight vectors $\mathcal{Q}$, they identify those weight vectors in $\mathcal{Q}$ that rank a specific option $\boldsymbol{p}_i$ among the top-k. Follow up work improved performance using branch-and-bound techniques [46] and batch processing [18]. In related formulations, [45] computes the options that belong to the top-k result of most vectors in $\mathcal{Q}$, while [24] selects m options from $\mathcal{D}$ (for an input integer m), such that the largest number of vectors in $\mathcal{Q}$ are covered, i.e., include at least one of the m options in their top-k result. In a similar setting, the *reverse k-ranks* query [55] reports the m query vectors in $\mathcal{Q}$ that rank a specific option $\boldsymbol{p}_i$ the highest, or (in another variant) that associate $\boldsymbol{p}_i$ with the highest scores.

The *why-not top-k* query [21] computes the minimum perturbation (in terms of Euclidean distance) required in the weight vector $\boldsymbol{w}$ of a top-k query, so that a non-result option $\boldsymbol{p}_i$ (or a set of non-result options) is included in the top-k result. Specified a set of weight vectors $\mathcal{Q}$ and an option $\boldsymbol{p}_i$ that does not belong to any of their top-k results, the *why-not reverse top-k* query [26] computes the minimum perturbation required in $\boldsymbol{p}_i$ or in the weight vectors in $\mathcal{Q}$, so that $\boldsymbol{p}_i$ is included in all their results.

Assuming a known set of weight vectors $\mathcal{Q}$, Yang et al. [50] compute the attribute values a new option $\boldsymbol{o}$ should have, so that (i) it is the top-1 option in the dataset for at least m weight vectors, and (ii) the cost to create $\boldsymbol{o}$ is minimized. The cost function is monotonic to the attribute values, implying that the more competitive $\boldsymbol{o}$ is made, the more expensive it is to create. Again for a known set of weight vectors $\mathcal{Q}$, and a cost function, Yang and Cai [49] compute the minimum-cost "improvement vector" (that specifies an increment value for each dimension) for an existing option $\boldsymbol{p}_i$, so that the enhanced $\boldsymbol{p}_i$ is in the top-k result of at least m vectors in $\mathcal{Q}$.

Studies in this subsection are intrinsically bound to a finite number of specific weight vectors. On the contrary, in our problem we need to consider every possible weight vector in a continuous (part of the) preference space. Take for example the why-not reverse top-k query [26], which is related to our side-problem of modifying (enhancing) an existing option $\boldsymbol{p}_i$, so that it belongs to the top-k result of any weight vector in a region wR. To adapt [26], we could apply it for a set $\mathcal{Q}$ of samples (weight vectors) randomly/uniformly chosen within wR. For one, there is no guarantee that the modified $\boldsymbol{p}_i$ would be among the top-k for every possible vector in wR, i.e., the solution would be inexact. Second, it is not trivial (if possible at all) to derive approximation bounds for the quality of the result, e.g., to estimate the ratio of wR that is covered with respect to the number of sample vectors. Our methodology respects the continuous nature of the preference space and produces exact results, i.e., guarantees the complete coverage of wR (at minimum cost, when the option creation/modification cost is a concern).

2.2 Processing in the Continuous Pref. Space

The first work to consider the continuous preference space assumes 2-dimensional options, and maintains a subset of a (dynamic) dataset that is guaranteed to contain the answer to any top-k query [15]. That is achieved by observations in a dual space (essentially, the preference space), where options are mapped to lines and queries to vertical rays. This approach is conceptually extended to higher dimensions in the context of continuous top-k queries [51].

In [40], Soliman et al. identify the most probable score-based ordering of a dataset when the weight vector is equally likely to be anywhere in the preference space. This task entails considering different partitions in preference space, and the respective orderings, although it resorts to sampling for more than 3 dimensions. On regions and respective orderings, given a top-k result, Zhang et al. [54] compute the region in preference space where any weight vector produces the said result. That region's volume could serve as a sensitivity measure for the top-k result.

In [31], the market impact of an option $\boldsymbol{p}_i$ is assessed by computing the best rank it could achieve for any weight vector. That entails a progressive consideration of competing options and a gradual pruning of unpromising partitions in the preference space. Tang et al. [41] address the monochromatic version of the reverse top-k query (originally defined and solved for 2-dimensional datasets in [44]), i.e., they compute all regions in preference space where a weight vector would rank a specific option $\boldsymbol{p}_i$ among the top-k.

Assuming that the weight vector $\boldsymbol{w}$ may lie anywhere in a region wR in preference space, [14] identifies all possible top-1 options.

In a similar setting, [30] reports all possible top-k sets for any $\boldsymbol{w}$ in wR, together with the specific partition in wR that produces each of these top-k sets. Starting with an option $\boldsymbol{p}_i$ as anchor, it computes the rank of $\boldsymbol{p}_i$ (and identifies all options that rank higher than it) in different parts of wR. For each part where the rank of $\boldsymbol{p}_i$ is not exactly k, a different anchor is chosen, and the process is recursively repeated (for the specific part and the new anchor). The recursion in a part stops when the anchor has rank exactly k everywhere in it, and thus the (order-insensitive) top-k set for that part is known.

In [35], specified the probability density function of the weight vector, the objective is to select m options from $\mathcal{D}$, such that the top-1 answer has the highest probability to be among the chosen options. That problem is related to regret minimizing sets [32, 13, 7, 48], e.g., selecting a subset of m options from $\mathcal{D}$, so that the top-1 option in the subset does not score much lower than the top-k-th option in the entire $\mathcal{D}$ for any possible weight vector.

The methods described in this subsection work for a continuous (part of the) preference space. However, they do not consider the creation of a new option or the improvement of an existing one. That is, the options are taken as is, and no deciding on their values is possible. This renders them inapplicable to our problem, although in Section 3 we build a (inefficient, as we show in the experiments) baseline that relies on [30] as a building block.

2.3 Option Creation and Enhancement

Creating competitive options and improving existing ones has been considered in the context of the *skyline*, an one-size-fits-all operator that (unlike top-k) does not take personalized preferences into account. An option dominates another if it has no smaller value in any dimension, and has a larger value in at least one dimension. The skyline of $\mathcal{D}$ is the set of options that are not dominated by any other [10]; more generally, the *k-skyband* includes options that are dominated by fewer than k others. The standard approach for skyline (and k-skyband) computation applies the branch-and-bound paradigm to options indexed by a spatial access method [34].

Dominance has been used as a guide to create or enhance options. E.g., [25] and [19] aim to create a competitive option, on the assumption that it will be more marketable if it dominates many existing options but is dominated by few, subject to several constraints (e.g., creation budget). Wan et al. [47] choose options to create (from a combinatorial set of candidates), so that they all belong to the skyline of a greater set of alternatives. Peng et al. [36] consider a similar problem, where the creation cost and the potential earnings of each candidate are taken into account. Assuming that some options are upgradable (at a cost), [27] identifies m of them that can be enhanced at the lowest total cost, so that they all appear in the skyline. Skyline-based methods are inapplicable to our problem, as they do not deal with score-based ranking. Application-wise, due to the non-personalizable nature of dominance, they are not suitable when the target is a specific type of clientele.

Miah et al. [28] propose techniques to publicize only a subset of an option's attributes in order to increase the number of queries (from a known set) that would include it in their result. Asudeh et al. [6] determine which additional services/amenities an option should offer, so that the cost of the additions does not exceed a budget, and a gain function (defined in relation to other available options) is maximized. In a marketplace where options receive (positive and negative) tags from users, Das et al. [16] use past correlations between option attributes and tags, in order to create new options that are most likely to attract a set of desirable tags. Problem definitions in these studies are fundamentally different from TopRR and their methods are inapplicable to it. Moreover, they focus on Boolean option attributes, or ordinal at best.

3. PRELIMINARIES & FUNDAMENTALS

3.1 Problem Definition

Our problem involves a dataset $\mathcal{D}$ with d attributes, where each option $\boldsymbol{p}_i \in \mathcal{D}$ is represented as $\boldsymbol{p}_i = (p_i[1], p_i[2], \cdots, p_i[d])$. The attributes (dimensions) define a continuous *option space*, denoted by $\mathcal{O}$, in which each option can be regarded as a d-dimensional point. Without loss of generality, we assume that larger values are preferable for every attribute. The top-k query takes as input a weight vector $\boldsymbol{w} = (w[1], w[2], \cdots, w[d])$ and the dataset $\mathcal{D}$, and retrieves the k options with the highest scores. The score of an option $\boldsymbol{p}_i \in \mathcal{D}$ according to $\boldsymbol{w}$ is defined as $S_{\boldsymbol{w}}(\boldsymbol{p}_i) = \boldsymbol{w} \cdot \boldsymbol{p}_i = \sum_{j=1}^{d} w[j] \cdot p_i[j]$. We assume that each weight vector $\boldsymbol{w}$ is normalized such that $\sum_{j=1}^{d} w[j] = 1$. Note that this does not affect the ranking of options, since the ranking does not depend on the magnitude of $\boldsymbol{w}$, but only on its direction [23]. Consequently, we may drop the last weight in $\boldsymbol{w}$, because it can be derived as $w[d] = 1 - \sum_{j=1}^{d-1} w[j]$. We refer to the $(d-1)$-dimensional domain of the weight vector as the *preference space* $\mathcal{W}$.

Specified value k and a preference region wR $\subset \mathcal{W}$, an option is deemed as a *top-ranking option* if it is among the top-k in $\mathcal{D}$ for every weight vector $\boldsymbol{w} \in$ wR. TopRR is defined as follows.

DEFINITION 1 (TOP-RANKING REGION PROBLEM). *Given a dataset $\mathcal{D}$, a positive integer k, and a preference region* wR $\subset \mathcal{W}$, *compute the maximal region* oR *in option space where a new option* **o** *should lie, so that it is a top-ranking option.*

Choosing a target clientele/region wR is a business decision [39, 53] that is orthogonal to our work. Our methodology requires that wR is a convex polytope in $\mathcal{W}$ (for easy presentation, we illustrate wR as an axis-aligned hyper-rectangle). However, our framework indirectly addresses non-convex polytopes too; given that any non-convex polytope can be partitioned into convex ones [8], the latter could be processed independently, and the intersection of their TopRR solutions reported as overall oR. Similarly, closed regions with curved sides can be approximated infinitely well by polytopes.

Maximality in Definition 1 asserts that oR includes all possible placements that would render $\boldsymbol{o}$ a top-ranking option. As we show later, oR is a convex polytope that includes the top-corner of the option space (i.e., the corner with maximum attribute values).

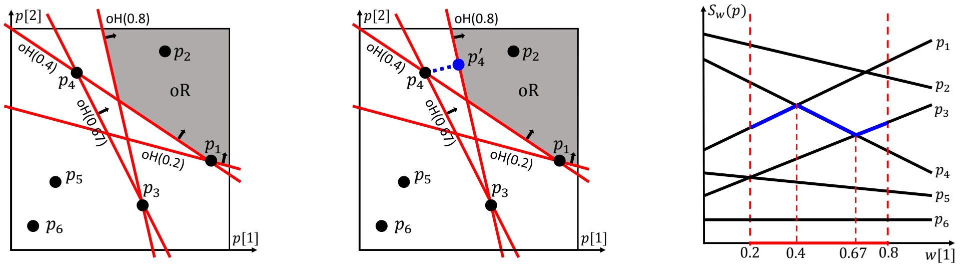
For example, consider the 2-dimensional dataset $\mathcal{D}$ in Figure 1(a), where the options are laptops with attributes speed and battery life. Since $d = 2$, the preference space $\mathcal{W}$ is the 1-dimensional domain $[0, 1]$. Suppose that the target preference region is wR $= [0.2, 0.8]$ and $k = 3$. Figure 1(b) represents the option space, and shows in gray the TopRR output, i.e., region oR.

If a new laptop $\boldsymbol{o}$ were to enter the market with the objective to be consistently among the top-3 for user types represented by wR, it should fall inside region oR. Similarly, if an existing model, like $\boldsymbol{p}_4$, were to be revamped with the same top-3 requirement, it should enter oR. If the redesign cost is proportional to the Euclidean distance between the current and the new version of $\boldsymbol{p}_4$, the cost-optimal top-ranking placement to go for is $\boldsymbol{p}'_4$ in Figure 1(c).

TopRR can additionally (albeit indirectly) address the problem of maximizing market impact (i.e., ensuring a top-k guarantee for the smallest possible k) subject to a specified redesign budget B. From the problem definition it follows that the TopRR result region in Figure 1 for any $k < 3$ is a subset of the gray area. In other words, the optimal redesign cost is certain to increase monotonically as we lower k. Thus, we may compute oR (and the respective cost-optimal placement $\boldsymbol{p}'_4$) for progressively smaller values of k,

Laptop	Speed	Battery
$\boldsymbol{p}_1$	0.9	0.4
$\boldsymbol{p}_2$	0.7	0.9
$\boldsymbol{p}_3$	0.6	0.2
$\boldsymbol{p}_4$	0.3	0.8
$\boldsymbol{p}_5$	0.2	0.3
$\boldsymbol{p}_6$	0.1	0.1

(a) Dataset



(b) oR region in option space (c) Enhancing option $\boldsymbol{p}_4$ (d) Scores in pref. space

Figure 1: TopRR example ($k = 3$)

and report the placement $\boldsymbol{p}'_4$ for the smallest k that requires a redesign cost within the available budget B.

Although we focus on option attributes with continuous domains, our work also applies when some attributes take values from a finite domain. An option is still top-ranking if and only if it falls in region oR, exactly as we define/compute it. The finiteness of an attribute's domain is an extra condition, applied after oR computation. Similarly, possible manufacturing constraints and attribute interdependencies (e.g., $p[1] + p[2] \leq 1.5$) could subsequently be imposed on (i.e., intersected with) oR.

3.2 Main Challenge

The main challenge in TopRR is the continuity of both the option space and the preference space. To elaborate, assume that we are to create a top-ranking option $\boldsymbol{o}$ for a specific weight vector $\boldsymbol{w} \in$ wR. Let $TopK(\boldsymbol{w})$ denote the k-th highest score in $\mathcal{D}$ according to $\boldsymbol{w}$ currently. Any top-ranking option should have a score no smaller than $TopK(\boldsymbol{w})$. The part of the option space that satisfies that condition is a halfspace, called *impact halfspace for* **w**.

Definition 2 (Impact Halfspace for $\boldsymbol{w}$). *Given a weight vector* **w** *(and problem inputs* k *and* $\mathcal{D}$*), the impact halfspace for* **w** *is defined as:*

$$\mathsf{oH}(\mathbf{w}) = \{\mathbf{o} \in \mathcal{O} : S_{\mathbf{w}}(\mathbf{o}) \geq TopK(\mathbf{w})\}$$

It follows that if we wish to create a top-ranking option $\boldsymbol{o}$ for a set of weight vectors $\{\boldsymbol{w}_1, \boldsymbol{w}_2, \cdots, \boldsymbol{w}_m\}$, option $\boldsymbol{o}$ should fall in the intersection of their impact halfspaces, i.e., in $\mathsf{oH}(\boldsymbol{w}_1) \cap \mathsf{oH}(\boldsymbol{w}_2) \cap \cdots \cap \mathsf{oH}(\boldsymbol{w}_m)$. Applying that to TopRR definition, we get:

$$\mathsf{oR} = \bigcap_{\boldsymbol{w} \in \mathsf{wR}} \mathsf{oH}(\boldsymbol{w})$$

which, although theoretically valid, by itself is of little practical use, because the preference region wR includes infinite possible weight vectors. The main challenge is to define oR exactly, yet in a form of finite terms.

3.3 Crucial Theorem

Our way to define oR in manageable terms passes through the definition of *rank-k invariant preference region* (kIPR), and two lemmata that lead to the core theorem in our methodology.

Definition 3 (Rank-k Invariant Preference Region). *A region* wR_i *in preference space is a rank-k invariant preference region (*kIPR*) if the top-k result for every weight vector* $\mathbf{w} \in \mathsf{wR}_i$ *(i) comprises the same k options, and (ii) the top-k-th option is always the same.*

The definition disregards any score reordering among the top $(k-1)$ options for different weight vectors, but requires that the k-th be the same. To illustrate, in Figure 1(d) the x-axis represents the preference space in our running example, and the plotted lines indicate the scores of options. Region $[0.67, 0.8]$ on the x-axis is a kIPR, because every weight vector $\boldsymbol{w} \in [0.67, 0.8]$ yields (i) the same top-3 set[1] $\{\boldsymbol{p}_1, \boldsymbol{p}_2, \boldsymbol{p}_3\}$, and (ii) the same top-3-rd option $\boldsymbol{p}_3$. In contrast, regions $[0.2, 0.8]$ and $[0.2, 0.67]$ are not kIPR.

To perceive the shape of a (maximal) kIPR region, observe that conditions (i) and (ii) in Definition 3 are each equivalent to a number of pairwise inequalities of the form $S_{\boldsymbol{w}}(\boldsymbol{p}_i) \geq S_{\boldsymbol{w}}(\boldsymbol{p}_j)$. Specifically, condition (i) requires for each option $\boldsymbol{p}_i$ in the top-k set, that $S_{\boldsymbol{w}}(\boldsymbol{p}_i) \geq S_{\boldsymbol{w}}(\boldsymbol{p}_j)$ for every non-result option $\boldsymbol{p}_j$. Condition (ii) additionally requires for each of the top $(k-1)$ options $\boldsymbol{p}_i$, that $S_{\boldsymbol{w}}(\boldsymbol{p}_i) \geq S_{\boldsymbol{w}}(\boldsymbol{p}_k)$, where $\boldsymbol{p}_k$ is the top-k-th option. Since each of the pairwise inequalities corresponds to a halfspace in preference space, a maximal kIPR is the intersection of the respective halfspaces, i.e., it is a convex polytope [8].

We first attempt to solve TopRR for a kIPR region wR_i, before extending to general input regions. Key to that attempt is Lemma 1.

Lemma 1. *Let* **p** *and* **p**$'$ *be two options, and* wR_i *be a convex polytope in preference space. Let also* $\mathcal{V}$ *be the vertex set that defines* wR_i*, i.e., the set of the polytope's extreme vertices. If it holds that* $S_{\mathbf{v}}(\mathbf{p}) \geq S_{\mathbf{v}}(\mathbf{p}')$ *for every vertex* $\mathbf{v} \in \mathcal{V}$*, then* $S_{\mathbf{w}}(\mathbf{p}) \geq S_{\mathbf{w}}(\mathbf{p}')$ *also holds for every weight vector* $\mathbf{w} \in \mathsf{wR}_i$*.*

Proof. Consider a weight vector $\boldsymbol{w} \in \mathsf{wR}_i$. Since wR_i is a convex polytope, $\boldsymbol{w}$ can be expressed as:

$$\boldsymbol{w} = \sum_{\boldsymbol{v} \in \mathcal{V}} \lambda_{\boldsymbol{v}} \boldsymbol{v} \qquad (1)$$

for some positive $\lambda_{\boldsymbol{v}}$ coefficients, where $\sum_{\boldsymbol{v} \in \mathcal{V}} \lambda_{\boldsymbol{v}} = 1$.

Since $S_{\boldsymbol{v}}(\boldsymbol{p}) \geq S_{\boldsymbol{v}}(\boldsymbol{p}')$ for each vertex $\boldsymbol{v} \in \mathcal{V}$, it follows that $\lambda_{\boldsymbol{v}} S_{\boldsymbol{v}}(\boldsymbol{p}) \geq \lambda_{\boldsymbol{v}} S_{\boldsymbol{v}}(\boldsymbol{p}')$ too. By summing across all $\boldsymbol{v} \in \mathcal{V}$, we get:

$$\begin{aligned}
& \sum_{\boldsymbol{v} \in \mathcal{V}} \lambda_{\boldsymbol{v}} S_{\boldsymbol{v}}(\boldsymbol{p}) \geq \sum_{\boldsymbol{v} \in \mathcal{V}} \lambda_{\boldsymbol{v}} S_{\boldsymbol{v}}(\boldsymbol{p}') \\
\Rightarrow\ & \sum_{\boldsymbol{v} \in \mathcal{V}} \lambda_{\boldsymbol{v}} (\boldsymbol{v} \cdot \boldsymbol{p}) \geq \sum_{\boldsymbol{v} \in \mathcal{V}} \lambda_{\boldsymbol{v}} (\boldsymbol{v} \cdot \boldsymbol{p}') \\
\Rightarrow\ & \boldsymbol{p} \cdot \sum_{\boldsymbol{v} \in \mathcal{V}} \lambda_{\boldsymbol{v}} \boldsymbol{v} \geq \boldsymbol{p}' \cdot \sum_{\boldsymbol{v} \in \mathcal{V}} \lambda_{\boldsymbol{v}} \boldsymbol{v} \qquad \triangleright \text{ applying Equation (1)} \\
\Rightarrow\ & \boldsymbol{p} \cdot \boldsymbol{w} \geq \boldsymbol{p}' \cdot \boldsymbol{w}
\end{aligned}$$

Thus, $\forall\, \boldsymbol{w} \in \mathsf{wR}_i$, it holds that $S_{\boldsymbol{w}}(\boldsymbol{p}) \geq S_{\boldsymbol{w}}(\boldsymbol{p}')$. $\square$

The following lemma defines (in finite terms) the TopRR solution for a kIPR convex polytope.

[1] We follow the convention that "top-k set" refers to the (order-insensitive) collection of the top k options, while "top-k result" to their score-sorted list.

LEMMA 2. *The* TopRR *solution for a* kIPR *convex polytope* wR_i *is option region*

$$\mathsf{oR}_i = \bigcap_{\mathbf{v} \in \mathcal{V}} \mathsf{oH}(\mathbf{v})$$

where $\mathcal{V}$ *is the vertex set that defines* wR_i*, and impact halfspaces* $\mathsf{oH}(\mathbf{v})$ *are as defined in Definition 2.*

PROOF. Since wR_i is a kIPR, the top-k-th option $\boldsymbol{p}_k$ is the same everywhere in it. If a new option $\boldsymbol{o}$ scores higher than $\boldsymbol{p}_k$ according to all weight vectors (vertices) $\boldsymbol{v} \in \mathcal{V}$, Lemma 1 guarantees that $\boldsymbol{o}$ also scores higher than $\boldsymbol{p}_k$ for every $\boldsymbol{w} \in \mathsf{wR}_i$, i.e., it is a top-ranking option for wR_i.

The option region oR_i is also maximal, i.e., any option $\boldsymbol{o}$ outside of it cannot be a top-ranking option. To see this, if $\boldsymbol{o}$ falls outside $\mathsf{oR}_i = \bigcap_{\boldsymbol{v} \in \mathcal{V}} \mathsf{oH}(\boldsymbol{v})$, there is a vertex $\boldsymbol{v} \in \mathcal{V}$ for which $\boldsymbol{o}$ is outside $\mathsf{oH}(\boldsymbol{v})$, i.e., $S_{\boldsymbol{v}}(\boldsymbol{o}) < S_{\boldsymbol{v}}(\boldsymbol{p}_k)$, thus $\boldsymbol{o}$ is not among the top-k for $\boldsymbol{v}$, and consequently not a top-ranking option in wR_i. □

With a manageable definition of the TopRR solution for a kIPR region, we frame the core theorem for general target wR regions (convex polytopes) in preference space. Assuming that a partitioning of wR into kIPRs is given, Theorem 1 provides an exact definition (in finite terms) of the TopRR solution oR.

THEOREM 1. *Given the partitioning of* wR *into* kIPR*s, let* $\mathcal{V}_{all}$ *be the set of all vertices that define those* kIPR*s. The* TopRR *solution for* wR *is option region*

$$\mathsf{oR} = \bigcap_{\mathbf{v} \in \mathcal{V}_{all}} \mathsf{oH}(\mathbf{v})$$

PROOF. For any of the kIPR partitions wR_i, the vertex set $\mathcal{V}$ that defines wR_i is a subset of $\mathcal{V}_{all}$. Hence, any new option $\boldsymbol{o}$ that falls in $\bigcap_{\boldsymbol{v} \in \mathcal{V}_{all}} \mathsf{oH}(\boldsymbol{v})$ must also fall in $\bigcap_{\boldsymbol{v} \in \mathcal{V}} \mathsf{oH}(\boldsymbol{v})$, and therefore (by Lemma 2) it is a top-ranking option for wR_i. As the union of all kIPR partitions is the entire wR, option $\boldsymbol{o}$ is a top-ranking option for any $\boldsymbol{w} \in \mathsf{wR}$.

On the maximality of oR, if an option $\boldsymbol{o}$ is outside $\bigcap_{\boldsymbol{v} \in \mathcal{V}_{all}} \mathsf{oH}(\boldsymbol{v})$, there is a $\boldsymbol{v} \in \mathcal{V}_{all}$ such that $\boldsymbol{o}$ is outside $\mathsf{oH}(\boldsymbol{v})$. This means that $\boldsymbol{o}$ scores lower than the top-k-th option for weight vector $\boldsymbol{v}$, and since $\boldsymbol{v} \in \mathsf{wR}$, option $\boldsymbol{o}$ is not top-ranking in wR. □

Returning to our running example, in Figure 1(d) the kIPRs are $[0.2, 0.4]$, $[0.4, 0.67]$, and $[0.67, 0.8]$. Set $\mathcal{V}_{all}$ includes vertices (i.e., values in the 1-dimensional preference space) 0.2, 0.4, 0.67, and 0.8. The respective impact halfspaces in option space are shown in Figure 1(b), and their intersection (gray area) is the TopRR solution oR.

Based on Theorem 1, our algorithmic design is centered on efficiently partitioning wR into kIPRs, and forming $\mathcal{V}_{all}$ from their defining vertices. In the next subsection, we adapt a method from existing work to perform that partitioning.

3.4 Baseline Solution

A straightforward first step (which applies across the board) is to disregard options that cannot be among the top-k for any $\boldsymbol{w}$ in wR, since it is evident from our problem definition that they cannot affect the TopRR solution. The literature includes several alternatives to do this (e.g., k-skyband [34], k-onion layers [11], etc); we review and choose the most suitable among them in Section 6.3.

As elaborated in Section 2.2, given a dataset $\mathcal{D}$ and a preference region $\mathsf{wR} \subset \mathcal{W}$, the *uncertain top-k* problem (UTK) in [30] computes all possible top-k sets (order-insensitive) for any $\boldsymbol{w}$ in wR, together with a partitioning of wR into regions (convex polytopes), each associated with one of the reported top-k sets. While UTK outputs top-k sets and regions in the preference space (as opposed to TopRR reporting a region in option space), the specific algorithm in [30] can serve as a building block in a baseline TopRR solution. In particular, recall that its termination condition is that the anchor (i.e., a specific option) of each output region has rank exactly k in the region. In other words, every region in the produced partitioning of wR is a kIPR (albeit most likely far from being a maximal kIPR). Although the number of output regions (kIPRs) may be exceeding, we can apply Theorem 1 on the union of their defining vertices to produce the TopRR result region oR in option space. We call the described baseline *partition-and-convert* (PAC).

3.5 Notation

Before we present our algorithms, we summarize important notation in Table 1.

Table 1: Notation

Notation	Description
$\mathcal{D}$	Option dataset
$\boldsymbol{p}_i$	An (existing) option in $\mathcal{D}$
$\mathcal{W}$	Continuous preference space
$\boldsymbol{w}$	A weight vector in $\mathcal{W}$
wR	Region in preference space
$\mathcal{O}$	Continuous option space
$\boldsymbol{o}$	New option to be placed in $\mathcal{O}$
oR	Region in option space (output of TopRR)
$S_{\boldsymbol{w}}(\boldsymbol{p}_i)$	The score of $\boldsymbol{p}_i$ according to $\boldsymbol{w}$ (equal to $\boldsymbol{p}_i \cdot \boldsymbol{w}$)
$TopK(\boldsymbol{w})$	k-th highest score in $\mathcal{D}$ according to $\boldsymbol{w}$
$\mathsf{oH}(\boldsymbol{w})$	Impact halfspace for $\boldsymbol{w}$ (Definition 2)
$\mathsf{wHP}(\boldsymbol{p}_i, \boldsymbol{p}_j)$	Hyperplane for $S_{\boldsymbol{w}}(\boldsymbol{p}_i) = S_{\boldsymbol{w}}(\boldsymbol{p}_j)$ in preference space
$\mathsf{wH}(\boldsymbol{p}_i, \boldsymbol{p}_j)$	Halfspace for $S_{\boldsymbol{w}}(\boldsymbol{p}_i) \geq S_{\boldsymbol{w}}(\boldsymbol{p}_j)$ in preference space

4. TEST-AND-SPLIT APPROACH

The performance of PAC suffers due to the high computation cost of the UTK component, and the numerous kIPRs it produces. In this section, we design an approach directly for our problem, called *test-and-split* (TAS). As explained in Section 3.3, our design is centered around the efficient partitioning of wR into kIPRs.

Initially, we test whether the input preference region wR is a kIPR. If not, we split it into sub-regions. We recursively repeat that test-and-split procedure on the sub-regions, until each of them is verified as a kIPR. Hence, the fundamental modules in our algorithm are (i) testing whether a region is kIPR, and (ii) splitting a non-kIPR region. These modules are described in Sections 4.1 and 4.2, respectively, before brought together in the complete TAS in Section 4.3.

4.1 kIPR Testing

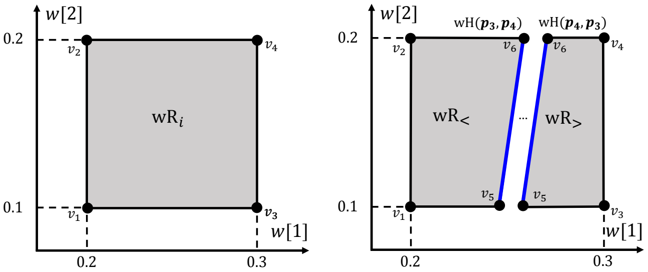
The testing module receives a preference region (convex polytope) wR_i as input and determines whether it is kIPR. To deal with the challenge that wR_i contains infinite possible weight vectors, Lemma 3 establishes an exact, yet practical condition to perform the test.

LEMMA 3. *Let* $\mathcal{V}$ *be the vertex set that defines a convex polytope* wR_i *in preference space.* wR_i *is* kIPR *if and only if all vertices in* $\mathcal{V}$ *have the same top-k set and the same top-k-th option.*

PROOF. Consider first the "if" direction of the lemma, and let $\boldsymbol{p}_k$ be the top-k-th option, and L_k be the top-k set (order-insensitive) at all vertices in $\mathcal{V}$. The "if" direction can be proven by showing that the following two statements hold for any $\boldsymbol{w} \in \mathsf{wR}_i$:
(1) For each $p_j \in \mathsf{L}_k - \{\boldsymbol{p}_k\}$, we have $S_{\boldsymbol{w}}(\boldsymbol{p}_j) \geq S_{\boldsymbol{w}}(\boldsymbol{p}_k)$.
(2) For each $p_j \in \mathcal{D} - \mathsf{L}_k$, we have $S_{\boldsymbol{w}}(\boldsymbol{p}_k) \geq S_{\boldsymbol{w}}(\boldsymbol{p}_j)$.

Table 2: Dataset

Laptop	Speed	Battery	Portability
$\boldsymbol{p}_1$	0.32	0.72	0.96
$\boldsymbol{p}_2$	0.85	0.91	0.65
$\boldsymbol{p}_3$	0.25	0.94	0.88
$\boldsymbol{p}_4$	0.81	0.65	0.72
$\boldsymbol{p}_5$	0.92	0.98	0.99



(a) wR_i (b) $\mathsf{wR}_<$ and $\mathsf{wR}_>$

Figure 2: Preference region splitting

For (1), each option $p_j \in \mathsf{L}_k - \{\boldsymbol{p}_k\}$ satisfies $S_{\boldsymbol{v}}(\boldsymbol{p}_j) \geq S_{\boldsymbol{v}}(\boldsymbol{p}_k)$ at each vertex $\boldsymbol{v} \in \mathcal{V}$. Using $\boldsymbol{p}_j, \boldsymbol{p}_k$ in Lemma 1 instead of $\boldsymbol{p}, \boldsymbol{p}'$ proves the statement true.

For (2), each option $p_j \in \mathcal{D} - \mathsf{L}_k$ satisfies $S_{\boldsymbol{v}}(\boldsymbol{p}_k) \geq S_{\boldsymbol{v}}(\boldsymbol{p}_j)$ at each vertex $\boldsymbol{v} \in \mathcal{V}$. Using $\boldsymbol{p}_k, \boldsymbol{p}_j$ in Lemma 1 instead of $\boldsymbol{p}, \boldsymbol{p}'$ proves that statement true as well.

Since both (1) and (2) hold, every $\boldsymbol{w} \in \mathsf{wR}_i$ yields L_k as the top-k set, and $\boldsymbol{p}_k$ as the top-k-th option. Thus, wR_i is kIPR.

The "only if" direction is trivially true according to the definition of kIPR, since the vertices in $\mathcal{V}$ are also in wR_i. □

Lemma 3 enables a simple procedure to verify whether wR_i is kIPR by simply computing the top-k result at each vertex in $\mathcal{V}$. Clearly, testing may terminate early (with negative result) when it encounters the first pair of vertices in $\mathcal{V}$ with different top-k set or different top-k-th option.

To exemplify, assume that $k = 3$, and consider the dataset in Table 2 and the preference region wR_i in Figure 2(a). In Table 3 we show the top-3 set and the top-3-rd option at each defining vertex of wR_i. The test returns false because $\boldsymbol{v}_1$ and $\boldsymbol{v}_3$ have different top-3-rd options.

Table 3: Top-3 set at each defining vertex of wR_i

Weight vector	Top-3 options	Top-3-rd option
$\boldsymbol{v}_1$	$\boldsymbol{p}_5, \boldsymbol{p}_1, \boldsymbol{p}_3$	$\boldsymbol{p}_3$
$\boldsymbol{v}_2$	$\boldsymbol{p}_5, \boldsymbol{p}_1, \boldsymbol{p}_3$	$\boldsymbol{p}_3$
$\boldsymbol{v}_3$	$\boldsymbol{p}_5, \boldsymbol{p}_1, \boldsymbol{p}_4$	$\boldsymbol{p}_4$
$\boldsymbol{v}_4$	$\boldsymbol{p}_5, \boldsymbol{p}_2, \boldsymbol{p}_4$	$\boldsymbol{p}_4$

4.2 Splitting Non-kIPR Regions

If a preference region (convex polytope) wR_i is not kIPR, it should be split recursively until all sub-regions are kIPR. Two issues arise in splitting wR_i:

(Q1) How to determine the splitting hyperplane?
(Q2) How to perform the split?

4.2.1 *Determining the Splitting Hyperplane*

The basic requirement for this module is a guarantee that the splitting hyperplane cuts through wR_i. That is essential, because invoking the splitting module (in Section 4.2.2) for a hyperplane that does not cut through wR_i would waste computations, without partitioning wR_i at all. Our strategy is as follows.

We first identify two among the defining vertices of wR_i that violate the kIPR requirements, say vertices $\boldsymbol{v}_a$ and $\boldsymbol{v}_b$. This can be done during the execution of the kIPR testing procedure for wR_i. Next, we identify two options $\boldsymbol{p}_{z1}$ and $\boldsymbol{p}_{z2}$ (from the top-k sets at $\boldsymbol{v}_a$ and $\boldsymbol{v}_b$, respectively) that lead to the violation. These options will indicate the splitting hyperplane for wR_i. We distinguish two cases in choosing a suitable pair $\boldsymbol{p}_{z1}, \boldsymbol{p}_{z2}$:

- Case 1: $\boldsymbol{v}_a, \boldsymbol{v}_b$ have different top-k sets. In this case, we choose as $\boldsymbol{p}_{z1}$ an option that appears in the top-k set at $\boldsymbol{v}_a$ but not in the top-k set at $\boldsymbol{v}_b$. Similarly, we choose as $\boldsymbol{p}_{z2}$ an option that appears in the top-k set at $\boldsymbol{v}_b$ but not in the top-k set at $\boldsymbol{v}_a$. If there are multiple pairs $(\boldsymbol{p}_{z1}, \boldsymbol{p}_{z2})$ that satisfy this requirement, one of them is chosen at random.
- Case 2: $\boldsymbol{v}_a, \boldsymbol{v}_b$ have the same top-k set but different top-k-th options. In this case, we choose as $\boldsymbol{p}_{z1}$ and $\boldsymbol{p}_{z2}$ the respective top-k-th options at $\boldsymbol{v}_a$ and $\boldsymbol{v}_b$.

After identifying pair $\boldsymbol{p}_{z1}, \boldsymbol{p}_{z2}$, we use the representation of equality $S_{\boldsymbol{w}}(\boldsymbol{p}_{z1}) = S_{\boldsymbol{w}}(\boldsymbol{p}_{z2})$ in preference space as the splitting hyperplane for wR_i, i.e.,

$$\mathsf{wHP}(\boldsymbol{p}_{z1}, \boldsymbol{p}_{z2}) = \{\boldsymbol{w} \in \mathcal{W} : S_{\boldsymbol{w}}(\boldsymbol{p}_{z1}) = S_{\boldsymbol{w}}(\boldsymbol{p}_{z2})\}$$

Continuing the example in Figure 2(a), vertices $\boldsymbol{v}_1$ and $\boldsymbol{v}_3$ violate the requirement of kIPR, because they yield different top-k-th options. Falling under Case 2, we choose options $\boldsymbol{p}_3$ and $\boldsymbol{p}_4$ as the violating pair from the respective top-k sets, and determine $\mathsf{wHP}(\boldsymbol{p}_3, \boldsymbol{p}_4)$ as the splitting hyperplane for wR_i. Lemma 4 proves that the described strategy fulfills our fundamental requirement.

LEMMA 4. *The chosen splitting hyperplane* $\mathsf{wHP}(\mathbf{p}_{z1}, \mathbf{p}_{z2})$ *is guaranteed to cut through* wR_i*, i.e., to divide it into non-empty parts.*

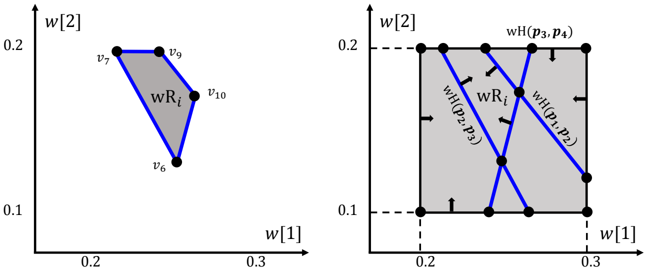
PROOF. In both Case 1 and Case 2 of our strategy for choosing pair $\boldsymbol{p}_{z1}, \boldsymbol{p}_{z2}$, the score order between them is sure to change inside wR_i, i.e., there is a (non-empty) part of wR_i where $S_{\boldsymbol{w}}(\boldsymbol{p}_{z1}) > S_{\boldsymbol{w}}(\boldsymbol{p}_{z2})$, and another (also non-empty) where $S_{\boldsymbol{w}}(\boldsymbol{p}_{z1}) < S_{\boldsymbol{w}}(\boldsymbol{p}_{z2})$. □

4.2.2 *Performing the Split*

Hyperplane $\mathsf{wHP}(\boldsymbol{p}_{z1}, \boldsymbol{p}_{z2})$ enables us to split wR_i into two sub-regions, namely, $\mathsf{wR}_<$ and $\mathsf{wR}_>$. Figure 2(b) demonstrates these sub-regions in our example. To a great extent, the efficiency of this operation depends on the representation model for convex polytope wR_i. We first describe the two standard preference space indexing/representation models in existing literature (employed by methods in Section 2.2), and then present our custom approach that is optimized for particular operations required by our algorithms.

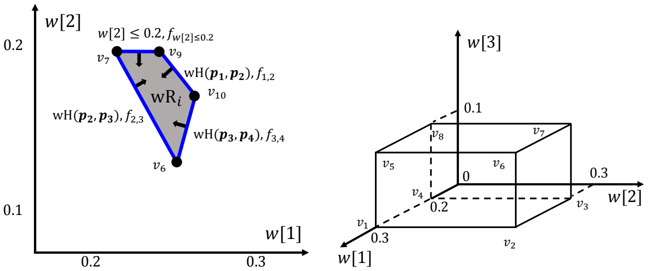
A common model is vertex-based representation [51, 31], which stores the defining vertices of wR_i. For example, in Figure 3(a), region wR_i is represented by vertices $\boldsymbol{v}_6, \boldsymbol{v}_7, \boldsymbol{v}_9, \boldsymbol{v}_{10}$. This model can readily support kIPR verification, as the latter operates on vertices. It also supports splitting a region, but requires two steps: (i) computing the exact geometry of wR_i by a convex hull algorithm [8], in order to derive the edges of wR_i (generally, the facets of the polytope), and (ii) computing the intersection points between the edges of wR_i and the splitting line (generally, the intersections between polytope facets and the splitting hyperplane). Step (i) is the most expensive, taking $O(n^{\lfloor d/2 \rfloor})$ time [12].

Another standard model is halfspace-based representation [41, 30], which stores the halfspaces that bound wR_i. Let $\mathsf{wH}(\boldsymbol{p}_i, \boldsymbol{p}_j)$ denote the halfspace corresponding to $S_{\boldsymbol{w}}(\boldsymbol{p}_i) \geq S_{\boldsymbol{w}}(\boldsymbol{p}_j)$ in preference space, and consider Figure 3(b). Convex polytope wR_i is


w[2]
0.2
0.1
v_7
v_9
v_{10}
v_6
wR$_i$
w[1]
0.2
0.3
w[2]
wH($\boldsymbol{p}_3,\boldsymbol{p}_4$)
wH($\boldsymbol{p}_2,\boldsymbol{p}_3$)
wR$_i$
wH($\boldsymbol{p}_1,\boldsymbol{p}_2$)
0.2
0.1
w[1]
0.2
0.3

(a) Vertex-based (b) Halfspace-based

Figure 3: Standard convex polytope representations


w[2]
$w[2] \leq 0.2, f_{w[2]\leq 0.2}$
0.2
v_7
v_9
wH($\boldsymbol{p}_1,\boldsymbol{p}_2$), $f_{1,2}$
wR$_i$
v_{10}
wH($\boldsymbol{p}_2,\boldsymbol{p}_3$), $f_{2,3}$
wH($\boldsymbol{p}_3,\boldsymbol{p}_4$), $f_{3,4}$
v_6
0.1
w[1]
0.2
0.3
w[3]
0.1
v_8
v_7
v_5
v_6
0
0.3
v_4
0.2
v_3
w[2]
v_1
0.3
v_2
w[1]

(a) Facet-based (2-D) (b) Facet-based (3-D)

Figure 4: Facet-based representation

expressed implicitly as the intersection of 7 halfspaces: 3 by option comparison (i.e., wH($\boldsymbol{p}_1$,$\boldsymbol{p}_2$), wH($\boldsymbol{p}_2$,$\boldsymbol{p}_3$), wH($\boldsymbol{p}_3$,$\boldsymbol{p}_4$)), and 4 by wR$_i$'s boundaries (i.e., $w[1] \geq 0.2, w[1] \leq 0.3, w[2] \geq 0.1, w[2] \leq 0.2$). Since this representation does not store vertices, it cannot readily support kIPR verification.

To avoid the above limitations, we consider a hybrid model to represent wR$_i$, the *facet-based representation*. A facet $f_{i,j}$ is expressed as a hyperplane wHP($\boldsymbol{p}_i$,$\boldsymbol{p}_j$), augmented with those defining vertices of wR$_i$ that fall on the facet. Figure 4(a) shows wR$_i$ in the 2-dimensional preference region. It is represented by 4 facets: $f_{1,2}, f_{2,3}, f_{3,4}, f_{w[2]\leq 0.2}$. Facet $f_{3,4}$, for instance, corresponds to hyperplane (line, in 2 dimensions) wHP($\boldsymbol{p}_3$,$\boldsymbol{p}_4$) and is augmented with vertices $\boldsymbol{v}_6$ and $\boldsymbol{v}_{10}$. Figure 4(b) illustrates a 3-dimensional preference region wR$_i$. It is represented by 6 facets: $f_{w[1]\geq 0.2}, f_{w[1]\leq 0.3}, f_{w[2]\geq 0}, f_{w[2]\leq 0.3}, f_{w[3]\geq 0}, f_{w[3]\leq 0.1}$. For example, facet $f_{w[1]\leq 0.3}$ is augmented with 4 vertices $\{\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{v}_5, \boldsymbol{v}_6\}$.

When considering a preference region wR$_i$, the facet representation offers fast kIPR testing (by applying Lemma 3 directly). In addition, since the facet representation stores bounding facets only, it excludes irrelevant hyperplanes (unlike halfspace-based representation, where a fraction of the halfspaces kept for wR$_i$ are not truly bounding it, as detailed in [41]).

Having elaborated on our representation model for preference regions (convex polytopes), we next describe the splitting operation on a region wR$_i$ according to the (already decided) splitting hyperplane wHP($\boldsymbol{p}_{z1}$,$\boldsymbol{p}_{z2}$). Let $\mathcal{V}$ be the set of vertices that define wR$_i$. We first classify them into the following sets:

$$\mathcal{V}_< = \{\boldsymbol{v} \in \mathcal{V} : S_{\boldsymbol{v}}(\boldsymbol{p}_{z1}) < S_{\boldsymbol{v}}(\boldsymbol{p}_{z2})\} \quad (2)$$

$$\mathcal{V}_> = \{\boldsymbol{v} \in \mathcal{V} : S_{\boldsymbol{v}}(\boldsymbol{p}_{z1}) > S_{\boldsymbol{v}}(\boldsymbol{p}_{z2})\} \quad (3)$$

Then, we assign each facet f of wR$_i$ to sub-regions as follows:

1. If all vertices of f are in $\mathcal{V}_<$, then f is assigned to wR$_<$.
2. If all vertices of f are in $\mathcal{V}_>$, then f is assigned to wR$_>$.
3. Otherwise, f contains vertices in both $\mathcal{V}_<$ and $\mathcal{V}_>$. In this case, it is necessary to intersect f with wHP($\boldsymbol{p}_{z1}$,$\boldsymbol{p}_{z2}$), splitting f into two facets, $f_<$ and $f_>$, and assigning them to wR$_<$ and wR$_>$, respectively.

After this process, the data structure of wR$_i$ can be released to free resources, since recursive testing-and-splitting will be applied to its sub-regions (convex polytopes) wR$_<$ and wR$_>$.

In Figure 2(a), for example, wR$_i$ is represented by 4 facets:

$$f_{w[1]\geq 0.2} : \{\boldsymbol{v}_1, \boldsymbol{v}_2\}$$
$$f_{w[1]\leq 0.3} : \{\boldsymbol{v}_3, \boldsymbol{v}_4\}$$
$$f_{w[2]\geq 0.1} : \{\boldsymbol{v}_1, \boldsymbol{v}_3\}$$
$$f_{w[2]\leq 0.2} : \{\boldsymbol{v}_2, \boldsymbol{v}_4\}$$

Continuing from Section 4.2.1, the splitting hyperplane was decided to be wHP($\boldsymbol{p}_3$,$\boldsymbol{p}_4$). We illustrate the splitting steps in Table 4. Steps 1 and 2 correspond to the simple cases. In step 3, facet $f_{w[2]\geq 0.1} : \{\boldsymbol{v}_1, \boldsymbol{v}_3\}$ is split into two parts and a new vertex $\boldsymbol{v}_5$ is produced. Step 4 is similar to step 3. Finally, the facets for the splitting hyperplane wHP($\boldsymbol{p}_3$,$\boldsymbol{p}_4$) are assigned to wR$_<$ and wR$_>$.

Table 4: Splitting steps for wR$_i$ in Figure 2(a)

Step	Facet of wR$_i$	Case	Action
1	$f_{w[1]\geq 0.2} : \{\boldsymbol{v}_1, \boldsymbol{v}_2\}$	(1)	assign it to wR$_<$
2	$f_{w[1]\leq 0.3} : \{\boldsymbol{v}_3, \boldsymbol{v}_4\}$	(2)	assign it to wR$_>$
3	$f_{w[2]\geq 0.1} : \{\boldsymbol{v}_1, \boldsymbol{v}_3\}$	(3)	assign $f_{w[2]\geq 0.1} : \{\boldsymbol{v}_1, \boldsymbol{v}_5\}$ to wR$_<$ assign $f_{w[2]\geq 0.1} : \{\boldsymbol{v}_5, \boldsymbol{v}_3\}$ to wR$_>$
4	$f_{w[2]\leq 0.2} : \{\boldsymbol{v}_2, \boldsymbol{v}_4\}$	(3)	assign $f_{w[2]\leq 0.2} : \{\boldsymbol{v}_2, \boldsymbol{v}_6\}$ to wR$_<$ assign $f_{w[2]\leq 0.2} : \{\boldsymbol{v}_6, \boldsymbol{v}_4\}$ to wR$_>$
5			assign $f_{3,4} : \{\boldsymbol{v}_5, \boldsymbol{v}_6\}$ to wR$_<$ assign $f_{3,4} : \{\boldsymbol{v}_5, \boldsymbol{v}_6\}$ to wR$_>$

4.3 Algorithm

Algorithm 1 summarizes the recursive test-and-split approach (TAS), which is initially invoked for preference region wR (input to TopRR), and partitions it into kIPRs. It maintains as global variable a set $\mathcal{V}_{all}$ to keep the union of vertices that define confirmed kIPRs. When the process terminates, we apply Theorem 1 on $\mathcal{V}_{all}$ to produce option region oR (TopRR output). Note that line 2 and lines 6-7 invoke the test and the split modules described in Sections 4.1 and 4.2, respectively.

Algorithm 1 Test-and-Split ($\mathcal{D}$, k, wR$_i$)

```
   Global var. $\mathcal{V}_{all}$: union of defining vertices of confirmed $k$IPRs
1: $\mathcal{V} \leftarrow$ the set of vertices that define wR$_i$
2: $flag \leftarrow$ apply $k$IPR testing                    ▷ test by Section 4.1
3: if $flag$ is true then
4:     $\mathcal{V}_{all} \leftarrow \mathcal{V}_{all} \cup \mathcal{V}$
5: else                                                       ▷ spit by Section 4.2
6:     identify options $\boldsymbol{p}_{z1}$,$\boldsymbol{p}_{z2}$ that lead to violation
7:     split wR$_i$ into wR$_<$ and wR$_>$ by hyperplane wHP($\boldsymbol{p}_{z1}$,$\boldsymbol{p}_{z2}$)
8:     Test-and-Split ($\mathcal{D}$, $k$, wR$_<$)
9:     Test-and-Split ($\mathcal{D}$, $k$, wR$_>$)
```

There are rudimentary dominance-based approaches to quickly eliminate options that could never be in the top-k for any $\boldsymbol{w}$ in wR. These methods are discussed in Section 6.3, and are applied before TopRR processing. Let n' be the number of remaining options (generally, $n' \ll n$, with several analyses available to estimate it, e.g., [20, 56]). In the worst case, during TopRR processing, all TAS pruning may fail, yielding a worst case complexity equal to that of the k-*level* of n' hyperplanes [5], i.e., $O(n'^{\lfloor d/2 \rfloor} k^{\lceil d/2 \rceil})$. The k-level is a standard notion in computational geometry [8].

5. OPTIMIZED TAS APPROACH

In this section, we propose *optimized test-and-split* (TAS*). Specifically, we enhance the TAS approach by (i) safely pruning additional options, (ii) optimizing region testing, and (iii) making more effective splitting hyperplane selections.

5.1 Disregarding Consistent Top-λ Options

In Section 3.4 we mentioned that there are standard techniques to prune uncompetitive options that cannot be among the top-k anywhere in wR, and therefore cannot affect TopRR solution. Likewise, here we show that we can also disregard options that are so competitive that they score higher than the top-k-th option everywhere in wR. Lemma 5 paves the way for said pruning and proves that it is safe, i.e., it does not threaten algorithm correctness.

LEMMA 5. *Let $\mathcal{V}$ be the vertex set that defines a convex polytope* wR *in preference space. If the top-λ set for a value $\lambda < k$ at every vertex $\mathbf{v} \in \mathcal{V}$ is the same, and letting Φ be that top-λ set, then* TopRR$(\mathcal{D}, k, \mathsf{wR})$ *is equivalent to* TopRR$(\mathcal{D} - \Phi, k - \lambda, \mathsf{wR})$*, i.e., they output the same option region* oR.

PROOF. Let $\boldsymbol{p}$ be an option in Φ, and $\boldsymbol{p}'$ be an option in $\mathcal{D} - \Phi$. Since $S_{\boldsymbol{v}}(\boldsymbol{p}) \geq S_{\boldsymbol{v}}(\boldsymbol{p}')$ for every vertex $\boldsymbol{v} \in \mathcal{V}$, Lemma 1 guarantees that $S_{\boldsymbol{w}}(\boldsymbol{p}) \geq S_{\boldsymbol{w}}(\boldsymbol{p}')$ everywhere in wR. That it, Φ is also the top-λ set for every $\boldsymbol{w} \in$ wR.

Since $\lambda < k$, no option in Φ could be the top-k-th anywhere in wR. As elaborated in Section 3.2, the TopRR result (option region oR) depends only on $TopK(\boldsymbol{w})$ (i.e., the k-th highest score) for vectors $\boldsymbol{w} \in$ wR, and therefore no option in Φ can affect it. In other words, the output of TopRR$(\mathcal{D}, k, \mathsf{wR})$ is the same as TopRR$(\mathcal{D} - \Phi, k - \lambda, \mathsf{wR})$. □

In effect, Lemma 5 suggests that set Φ can be disregarded, subject to parameter k being reduced by λ. To illustrate, consider the dataset in Table 2, preference region wR in Figure 2(a), and $k = 3$. The top-3 results of the vertices defining wR are shown in Table 3. They all share a common top-1 set $\Phi = \{\boldsymbol{p}_5\}$. By Lemma 5, we can safely ignore $\Phi = \{\boldsymbol{p}_5\}$ from $\mathcal{D}$, and decrement k by 1.

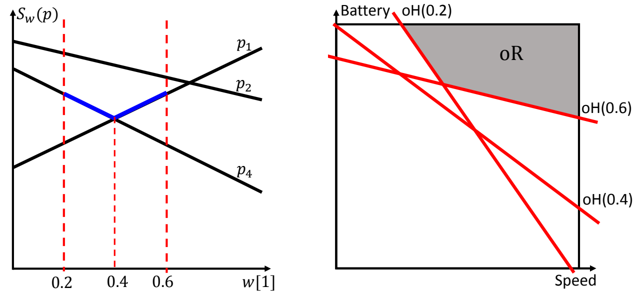
In our experiments, Lemma 5 reduces the number of considered options up to 2.8 times (see Section 6.5).

5.2 Optimized Region Testing

The TAS approach relies on kIPR testing (in Section 4.1). If a preference region wR$_i$ fails the test, it requires splitting and invokes further recursive calls of the algorithm. In this section, we propose an optimized testing technique that might allow wR$_i$ to go unsplit, despite being a non-kIPR region!

To begin the discussion, assume that the TopRR input comprises the dataset in Figure 1(a), $k = 2$, and wR $= [0.2, 0.6]$. Figure 5(a) plots the scores of top-2 options for $\boldsymbol{w} \in$ wR; other options are omitted for clarity. Region wR is not a kIPR, because sub-regions $[0.2, 0.4]$ and $[0.4, 0.6]$ yield different top-2-nd options. Therefore, the TAS approach would split wR (i.e., split it at 0.4).

Interestingly, in this example, the result option region oR (shown in Figure 5(b)) depends on the boundary preferences (i.e., 0.2, 0.6) but not on the splitting position (i.e., 0.4). Specifically, the result option region oR is equivalent to oH$(0.2) \cap$ oH(0.6), which does not involve oH(0.4). This motivates us to develop an optimized testing technique to avoid such inconsequential splitting (and thus save computations). Before generalizing, an important stepping stone is Lemma 6 that applies to the special case of $k = 1$.



(a) Scores in 1-D pref. space (b) Option space ($d = 2$)

Figure 5: Optimized region testing example

LEMMA 6. *The* TopRR *solution for $k = 1$ and preference region (convex polytope)* wR$_i$ *is option region*

$$\mathsf{oR} = \bigcap_{\mathbf{v} \in \mathcal{V}} \mathsf{oH}(\mathbf{v})$$

where $\mathcal{V}$ is the vertex set that defines wR$_i$*, and impact halfspaces* oH$(\mathbf{v})$ *are as defined in Definition 2.*

PROOF. In this proof, we abbreviate the notation $TopK(\boldsymbol{w})$ by $Top(\boldsymbol{w})$, since $k = 1$.

Let $\boldsymbol{o}$ be any option in region $\bigcap_{\boldsymbol{v} \in \mathcal{V}}$ oH$(\boldsymbol{v})$. It follows that, for each vertex $\boldsymbol{v} \in \mathcal{V}$, $S_{\boldsymbol{v}}(\boldsymbol{o}) \geq Top(\boldsymbol{v})$, i.e., $S_{\boldsymbol{v}}(\boldsymbol{o}) \geq S_{\boldsymbol{v}}(\boldsymbol{p})$ for every $\boldsymbol{p} \in \mathcal{D}$. This implies (by Lemma 1) that $S_{\boldsymbol{w}}(\boldsymbol{o}) \geq S_{\boldsymbol{w}}(\boldsymbol{p})$ for every $\boldsymbol{p} \in \mathcal{D}$ and $\boldsymbol{w} \in$ wR$_i$, i.e., $\boldsymbol{o}$ is a top-ranking option.

The option region defined by the lemma is also maximal. To see this, if $\boldsymbol{o}$ falls outside oR $= \bigcap_{\boldsymbol{v} \in \mathcal{V}}$ oH$(\boldsymbol{v})$, there must be a vertex $\boldsymbol{v} \in \mathcal{V}$ for which $\boldsymbol{o}$ is outside oH$(\boldsymbol{v})$, which would mean that $\boldsymbol{o}$ scores below $Top(\boldsymbol{v})$ and is thus not a top-ranking option. □

We utilize the above lemma (for general values of k) to prove that, if all vertices that define a preference region (convex polytope) wR$_i$ share the same top $(k-1)$ options, then the impact halfspaces oH$(\boldsymbol{v})$ at those vertices alone are enough to define the option region oR$_i$ (i.e., the TopRR solution for wR$_i$). In that case, further splitting/partitioning wR$_i$ into kIPRs is unnecessary, and its defining vertices can be directly included into set $\mathcal{V}_{all}$ of Theorem 1.

LEMMA 7. *Let $\mathcal{V}$ be the vertex set that defines a convex polytope* wR$_i$ *in preference space. If every vertex $\mathbf{v} \in \mathcal{V}$ yields the same top-$(k-1)$ set, then the* TopRR *solution for* wR$_i$ *is option region*

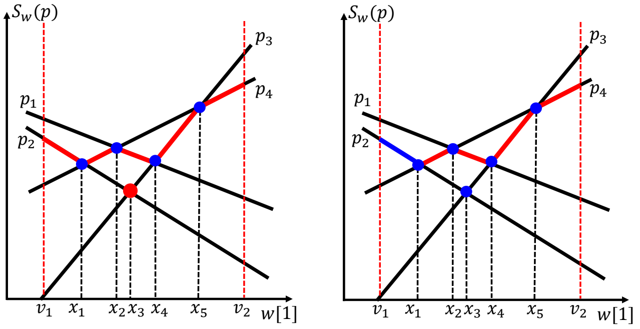
$$\mathsf{oR}_i = \bigcap_{\mathbf{v} \in \mathcal{V}} \mathsf{oH}(\mathbf{v})$$

PROOF. Since every vertex $\boldsymbol{v} \in \mathcal{V}$ yields the same top-$(k-1)$ set, we may apply Lemma 5 (for preference region wR$_i$) to reduce k down to 1. This renders Lemma 6 applicable to wR$_i$, thus completing the proof. □

Our evaluation in Section 6.5 reveals that Lemma 7 leads to 1.9 to 4.2 times fewer vertices in $\mathcal{V}_{all}$. An important remark is that the lemma also eliminates Case 2 in Section 4.2.1, since splitting is no longer necessary in that scenario. The next subsection improves splitting in the remaining (i.e., Case 1) violations.

5.3 Effective Splitting Hyperplane Selection

The TAS approach may choose a splitting hyperplane that is unnecessary for computing kIPRs. To illustrate, consider the scores of options in the 1-dimensional preference space in Figure 6(a), and assume that $k = 2$ and wR $= [\boldsymbol{v}_1, \boldsymbol{v}_2]$. The red curve indicates the score of the top-2-nd option, i.e., $TopK(\boldsymbol{w})$, for any weight



(a) wHP($\boldsymbol{p}_2$,$\boldsymbol{p}_3$), split at x_3

(b) wHP($\boldsymbol{p}_2$,$\boldsymbol{p}_4$), split at x_1

Figure 6: Scores in 1-dimensional preference space

$\boldsymbol{w} \in$ wR. The partitioning of wR into maximal kIPRs is indicated by the blue splitting points, i.e., regions $[\boldsymbol{v}_1, x_1]$, $[x_1, x_2]$, $[x_2, x_4]$, $[x_4, x_5]$, and $[x_5, \boldsymbol{v}_2]$.

TAS first obtains the top-2 sets of $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$, which are $\{\boldsymbol{p}_1,\boldsymbol{p}_2\}$ and $\{\boldsymbol{p}_3,\boldsymbol{p}_4\}$, respectively. Since the top-2 sets are different, wR is not kIPR. The pairs of options that violate the kIPR requirement are: $(\boldsymbol{p}_1,\boldsymbol{p}_3)$, $(\boldsymbol{p}_1,\boldsymbol{p}_4)$, $(\boldsymbol{p}_2,\boldsymbol{p}_3)$, $(\boldsymbol{p}_2,\boldsymbol{p}_4)$. TAS chooses one of these pairs randomly, say $(\boldsymbol{p}_2,\boldsymbol{p}_3)$, to form the splitting hyperplane wHP$(\boldsymbol{p}_2,\boldsymbol{p}_3)$. That hyperplane splits wR into $[\boldsymbol{v}_1, x_3]$ and $[x_3, \boldsymbol{v}_2]$. None of those sub-regions is a kIPR. Moreover, the maximal kIPR $[x_2, x_4]$ will eventually be output in the form of two non-maximal kIPRs (namely, $[x_2, x_3]$ and $[x_3, x_4]$). Such splitting does not threaten algorithm correctness (since Theorem 1 does not require maximality of the kIPRs) but it wastes computations and unnecessarily increases the number of vertices in $\mathcal{V}_{all}$. Hence, we want to avoid it as much as possible.

To reduce the chances of splitting poorly a non-kIPR region wR$_i$, we introduce the concept of *k-switch hyperplane*, which applies to Case 1 in Section 4.2.1, i.e., when two defining vertices $\boldsymbol{v}_a, \boldsymbol{v}_b$ of wR$_i$ have different top-k sets. Among the option pairs $\boldsymbol{p}_{z1},\boldsymbol{p}_{z2}$ responsible for the violation, instead of a random choice, we select the pair where $\boldsymbol{p}_{z1}$ is the top-k-th at $\boldsymbol{v}_a$, and $\boldsymbol{p}_{z2}$ scores lower, but as close as possible to $\boldsymbol{p}_{z1}$ at $\boldsymbol{v}_a$. The intuition is that this way $\boldsymbol{p}_{z2}$ is very likely to be the top-$(k+1)$-th option at $\boldsymbol{v}_a$, and thus the produced splitting hyperplane wHP$(\boldsymbol{p}_{z1},\boldsymbol{p}_{z2})$ to leave an entire maximal kIPR on its side facing $\boldsymbol{v}_a$. Formally:

DEFINITION 4 (k-SWITCH HYPERPLANE). *Given a non-kIPR preference region* wR$_i$*, let* $\mathbf{v}_a, \mathbf{v}_b$ *be a pair of its defining vertices that violate the kIPR requirement. We call* wHP$(\mathbf{p}_{z1}, \mathbf{p}_{z2})$ *a k-switch hyperplane if (i)* $\mathbf{p}_{z1}$ *is the top-k-th option of* $\mathbf{v}_a$*, and (ii)* $\mathbf{p}_{z2} = argmin_{\mathbf{p} \in \mathsf{L}_C} |S_{\mathbf{v}_a}(\mathbf{p}_{z1}) - S_{\mathbf{v}_a}(\mathbf{p})|$ *where set* L_C *is defined as:*

$$\mathsf{L}_C = \{\mathbf{p}_z \in \mathsf{L}_{\mathbf{v}_b} : S_{\mathbf{v}_a}(\mathbf{p}_z) < S_{\mathbf{v}_a}(\mathbf{p}_{z1}) \wedge S_{\mathbf{v}_b}(\mathbf{p}_z) > S_{\mathbf{v}_b}(\mathbf{p}_{z1})\}$$

and $\mathsf{L}_{\mathbf{v}_b}$ *is the top-k set at* $\mathbf{v}_b$*.*

In the example of Figure 6 (with preference region wR $= [\boldsymbol{v}_1, \boldsymbol{v}_2]$ and $k = 2$), vertices $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$ fall under Case 1, i.e., have different top-k sets. Figure 6(b) illustrates how to compute a k-switch hyperplane. We set $\boldsymbol{p}_{z1} = \boldsymbol{p}_2$ (i.e., the top-2-nd option at $\boldsymbol{v}_1$). We form set $\mathsf{L}_C = \{\boldsymbol{p}_3,\boldsymbol{p}_4\}$ by options that (i) belong to the top-k set at $\boldsymbol{v}_2$, and (ii) score lower than $\boldsymbol{p}_2$ at $\boldsymbol{v}_1$, but higher at $\boldsymbol{v}_2$. Among the options in L_C, we pick $\boldsymbol{p}_{z2} = \boldsymbol{p}_4$, because its score at vertex $\boldsymbol{v}_1$ is the closest to $\boldsymbol{p}_2$. The chosen pair $(\boldsymbol{p}_2,\boldsymbol{p}_4)$ forms a k-switch hyperplane wHP$(\boldsymbol{p}_2,\boldsymbol{p}_4)$, which splits wR into sub-regions $[\boldsymbol{v}_1, x_1]$ and $[x_1, \boldsymbol{v}_2]$. Observe that the former is a maximal kIPR. A remark is that it is possible for L_C to be empty, in which case we reverse the role of $\boldsymbol{v}_a$ and $\boldsymbol{v}_b$ and repeat the process.

Our experiments confirm that the k-switch optimization reduces drastically the number of poor/unnecessary splits, offering alone an almost 10-fold reduction in the overall number of vertices in $\mathcal{V}_{all}$ (see Section 6.5).

5.4 Algorithm

In the optimized test-and-split approach (TAS*), we invoke the recursive Algorithm 2 for $\mathcal{D}'$, value k', and input preference region wR; note that $\mathcal{D}'$ and $k' = k - \lambda$ refer to the updated dataset and the possibly decremented value of k after pruning by Lemma 5 in Section 5.1. Algorithm 2 performs region testing in a cascading manner, applying first our original kIPR testing (from Section 4.1) and, in case of failure, the optimized testing (from Section 5.2). If the region passes any of these tests, its defining vertices are inserted into set $\mathcal{V}_{all}$ (global variable). Otherwise, the algorithm selects a splitting hyperplane in line 9 (enhanced by the optimization in Section 5.3), splits the region into two sub-regions, and executes recursively on them. Note that pruning top options by Lemma 5 is applicable at every recursion (line 2), since options that were not pruned for wR$_i$, could be pruned for a sub-region. When Algorithm 2 terminates, we apply Theorem 1 on $\mathcal{V}_{all}$ to produce option region oR (TopRR output).

Algorithm 2 Optimized Test-and-Split ($\mathcal{D}'$, k', wR$_i$)

```
    Global var. $\mathcal{V}_{all}$: union of defining vertices of valid regions
 1: $\mathcal{V}$ ← the set of vertices that define wR$_i$
 2: $\mathcal{D}', k'$ ← apply top option pruning by Lemma 5
 3: $flag$ ← apply $k$IPR testing                          ▷ test by Section 4.1
 4: if $flag$ is false then
 5:     $flag$ ← apply optimized testing                   ▷ test by Section 5.2
 6: if $flag$ is true then
 7:     $\mathcal{V}_{all}$ ← $\mathcal{V}_{all} \cup \mathcal{V}$
 8: else                                                   ▷ spit must be performed
 9:     identify wHP$(\boldsymbol{p}_{z1},\boldsymbol{p}_{z2})$        ▷ enhanced by Section 5.3
10:     split wR$_i$ into wR$_<$ and wR$_>$ by hyperplane wHP$(\boldsymbol{p}_{z1},\boldsymbol{p}_{z2})$
11:     Optimized-Test-and-Split ($\mathcal{D}'$, $k'$, wR$_<$)
12:     Optimized-Test-and-Split ($\mathcal{D}'$, $k'$, wR$_>$)
```

The worst case analysis for TAS at the end of Section 4.3 applies to TAS* too. In practice, however, value n' is expected to be even smaller, and k reduced to k', both due to Lemma 5. Also, Lemma 7 makes the worst case reduction to k-level even more unlikely.

6. EXPERIMENTAL EVALUATION

We describe the experimental setting in Section 6.1. In Section 6.2, we conduct a case study on a real laptop dataset to demonstrate the applicability of the TopRR problem. In Section 6.3, we review standard techniques to use as a fast filtering step in TopRR processing. In Section 6.4, we evaluate all TopRR methods on synthetic and real datasets. Lastly, in Section 6.5, we investigate the effectiveness of each optimization technique described in Section 5.

6.1 Experimental Setting

We conducted all experiments on a machine with Intel Xeon E5-2620, 2.1 GHertz CPU and 64 GBytes memory, running on CentOS. We implemented all methods in C++: (i) *partition-and-covert* (PAC), i.e., the baseline in Section 3.4, (ii) *test-and-split* (TAS), as described in Section 4, and (iii) *optimized test-and-split* (TAS*), which enhances TAS by the techniques in Section 5. The methods call on the qhull library (from qhull.org) for halfspace intersection.

Table 5: Tested parameters

Parameter	Tested values
Dataset size n	0.1M, 0.2M, **0.4M**, 0.8M, 1.6M
Dimensionality d	2, **4**, 6, 8, 10, 12
Value k	1, 5, **10**, 20, 40
Pref. region's side-length σ	0.1%, 0.5%, **1.0%**, 5%, 10%
Data distribution	COR, **IND**, ANTI

The real datasets are (i) HOTEL, with 418,843 options and 4 attributes (stars, price, number of rooms, and number of facilities) [1], (ii) HOUSE, with 315,265 options and 6 attributes (gas, electricity, water, heating, insurance, and tax) [2], and (iii) NBA, with 21,960 options and 8 attributes (e.g., rebounds, points, assists, etc) [4]. The synthetic datasets are standard benchmarks [10] by a data generator, which takes 3 input parameters, namely (1) the number of options n, (2) the number of attributes d, and (3) the type of data distribution, i.e., *Independent* (IND), *Correlated* (COR), and *Anticorrelated* (ANTI).

Table 5 summarizes the experiment parameters and their tested values. Each input region wR is an axis-aligned hyper-cube, randomly generated in a preference space with unit axes. Parameter σ controls the size of wR, and denotes the side-length of wR as a fraction of the respective axis of the preference space. In each experiment, we vary one parameter in its tested range and fix the remaining ones to their default values (typed in boldface in Table 5). The processing times reported represent the average wall-clock running time of 50 TopRR queries.

6.2 Case Study

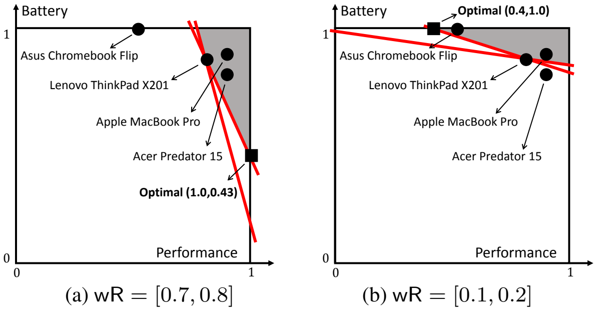
We start by demonstrating the usefulness of TopRR with a case study on real data. Inspired by the first example we used in the paper (Figure 1), we crawled a major tech-product review portal, CNET [3], and formed a dataset of ratings for 149 laptops. For ease of demonstration, we kept only performance and battery life ratings (i.e., $d = 2$), and normalized the dataset to a unit option space. For 2-dimensional options, the preference space is 1-dimensional. That is, weight $\boldsymbol{w} = (1)$ implies preference based only on performance, while $\boldsymbol{w} = (0)$ preference based only on battery life.

Suppose that a laptop manufacturer is targeting designers, i.e., a clientele that traditionally has a strong preference for performance over battery life. Modeling its target user type by preference region wR $= [0.7, 0.8]$, the manufacturer aims for the new laptop to rank consistently among the top-3 options for the said user type. Figure 7(a) illustrates the actual TopRR output (i.e., region oR) as the gray area. That area covers all possible options that will always be among the top-3 for any user preference within wR, and indicates to the manufacturer what characteristics the new laptop should have.

As typically production cost increases with both involved criteria, oR can help determine the ideal laptop placement to attain a guaranteed top-3 ranking at the smallest manufacturing cost. For example, if production cost is proportional to the summed squares of the option attributes, it could be modeled as

$$cost(product) = performance^2 + battery^2$$

The cost-optimal placement can be found by convex optimization [38] within the gray area and, in our scenario, it is shown as the black square in Figure 7(a). If we apply the cost formula to existing products in the gray area, the new laptop is cheaper to produce by 18.6%–27.1%. That allows for a competitive pricing with a wide profit margin for the manufacturer.



(a) wR $= [0.7, 0.8]$ (b) wR $= [0.1, 0.2]$

Figure 7: Introducing a new laptop to the market, $k = 3$

We conduct a similar study, assuming this time that the manufacturer is targeting a business crowd with strong demand for long battery life and little need for performance. Modeling that clientele by preference region wR $= [0.1, 0.2]$, the optimal laptop placement is shown as the black square in Figure 7(b). In addition to a guaranteed top-3 ranking, that laptop is also cheaper to build than its direct competitors (i.e., existing laptops in the gray area) by 7.2%–27.1%. The scenario of revamping an existing laptop is similar.

6.3 Fast Pruning of Non-top-k Options

As mentioned in Section 3.4, a straightforward first step in TopRR processing is to disregard options that cannot be among the top-k for any $\boldsymbol{w}$ in wR, since by definition they cannot affect the TopRR solution. The literature includes 4 approaches to compute a subset $\mathcal{D}' \subset \mathcal{D}$ such that $\mathcal{D}'$ is a guaranteed superset of all options that could rank among the top-k for any $\boldsymbol{w}$ in wR:

(i) *k-skyband* [34]: Defined in Section 2.3, it is guaranteed to include the top-k result for any weight vector in the preference space. Options in the k-skyband serve as set $\mathcal{D}'$.

(ii) *k-onion layers* [11]: It offers the same, general weight vector guarantee as the k-skyband, by placing into $\mathcal{D}'$ the first k layers of the convex hull of $\mathcal{D}$.

(iii) *r-skyband*: [14] defines that an option *r-dominates* another when it scores higher for any $\boldsymbol{w}$ in a given preference region wR. The r-skyband includes those options of $\mathcal{D}$ that are r-dominated by fewer than k others, and thus forms a set $\mathcal{D}'$ guaranteed to include the top-k result for any $\boldsymbol{w} \in$ wR.

(iv) UTK [30]: Reviewed in Section 2.2, UTK computes exactly those options that may appear in the top-k result for any $\boldsymbol{w} \in$ wR.

All 4 approaches take $\mathcal{D}$ and k as input, but r-skyband and UTK additionally take wR. The first 3 approaches produce a superset of the options that may appear in the top-k result for $\boldsymbol{w} \in$ wR, while UTK reports exactly those that do appear among the top-k for some $\boldsymbol{w} \in$ wR.

For our fast filtering purposes, the ideal choice would produce a small $\mathcal{D}'$ and yet take little computation time. To choose among the 4 alternatives, we conduct an experiment on our default dataset (IND) for default parameter values. Figure 8 illustrates the number of retained options, i.e., $|\mathcal{D}'|$, versus the computation time for each method, normalized by the maximum size and time taken, to provide perspective. The k-skyband and k-onion layers are unable to incorporate the constraint expressed by wR, thus failing to effectively reduce the dataset size and also taking more time. UTK achieves the optimal size reduction, but its computation time is two times higher than r-skyband. As such, we use r-skyband for fast filtering in all TopRR methods.

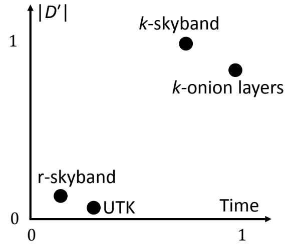


Figure 8: Trade-offs

6.4 Performance Evaluation

This is the main part of our experimental evaluation. First, we investigate the performance of TopRR methods (PAC, TAS, TAS*) with respect to different problem parameters, i.e., value k, preference region side-length σ, dataset size n, and dimensionality d.

Effect of value k: Figure 9(a) presents the average running time of the methods when varying k. TAS* achieves the best performance, beating PAC by up to two orders of magnitude. This is expected, as PAC relies on a UTK building block that was designed for a different problem, is computationally expensive, and produces numerous kIPRs. Since TAS* employs all optimization techniques in Section 5 (i.e., consistent top-scorer pruning, optimized region testing, smart splitting), it performs better than TAS, and the difference between them widens with k; the comparison attests to the strength and effectiveness of these optimizations (their individual effect is investigated in depth in Section 6.5). For completeness, in the chart embedded into Figure 9(a), we closer investigate small k values (i.e., $k \leq 5$); TAS* performs the best and PAC the worst in all cases, the former being 2.2 to 3.4 times faster than the latter.

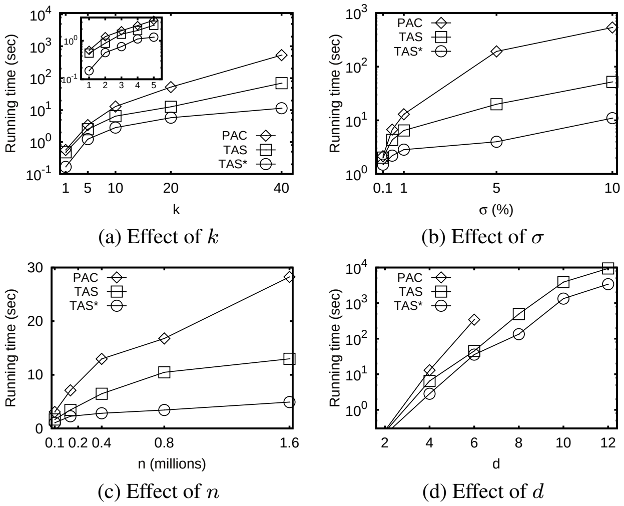
Effect of preference region side-length σ: In Figure 9(b), we evaluate the TopRR methods when varying σ, i.e., the side-length of the preference region wR. When σ increases, the volume of wR grows polynomially, and thus the number of kIPRs also rises quickly. Since both PAC and TAS require computing kIPRs, their performance degrades significantly with σ. In contrast, TAS* can terminate before reaching kIPRs, thanks to the optimized region testing in Section 5.2 (recall that it stops partitioning regions which meet the conditions of Lemma 7, without having to reduce them to kIPRs). TAS* also benefits from the effective splitting hyperplane selection technique in Section 5.3. The performance gap between TAS* and the other two methods widens with σ.

Effect of dataset size n: In Figure 9(c), we vary the dataset size n and report the running time of PAC, TAS and TAS*. All methods scale reasonably with n, with TAS* performing significantly better than its competitors. To provide some insight, at default setting, the number of processed options in TAS* is only 47.2% of those processed in TAS, which is attributed to the effectiveness of the techniques in Sections 5.1 and 5.2.

Effect of dimensionality d: We investigate the effect of dataset dimensionality d in Figure 9(d). The running time increases with d, as is commonly the case for problems that have a computational geometric nature [8]. PAC cannot terminate within 24 hours for $d \geq 8$. Nevertheless, our advanced method, TAS*, offers reasonable response times in all cases. That said, the standard top-k problem, and thus its derivatives too, suffer from the dimensionality curse [9], and are not meaningful for more than a handful of dimensions (except for hugely skewed or sparse datasets) because as d grows, the scores of all options in $\mathcal{D}$ quickly converge [52, 31].

Next, we study the robustness of TAS* with respect to different data distributions (COR, IND, ANTI), henceforth omitting the weaker methods TAS and PAC.

Effect of data distribution: Figure 10(a) examines the performance of TAS* on different data distributions as a function of k. The running time of TAS* is generally linear to k, regardless of the data distribution. For COR data, the option attributes are positively correlated, producing a small subset $\mathcal{D}'$ (since uncompetitive and consistent top-scoring options are easy to single out and disregard), thus requiring the fewest computations. On the other hand, for ANTI data, the remaining subset $\mathcal{D}'$ is large, hence requiring longer processing time.



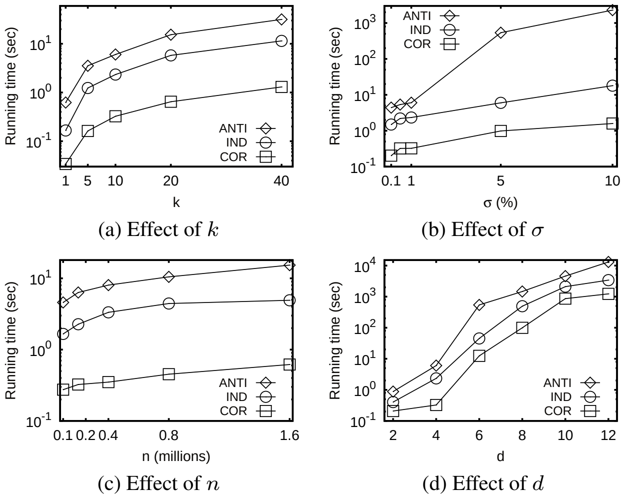
(a) Effect of k

(b) Effect of σ

(c) Effect of n

(d) Effect of d

Figure 9: Collective performance evaluation



(a) Effect of k

(b) Effect of σ

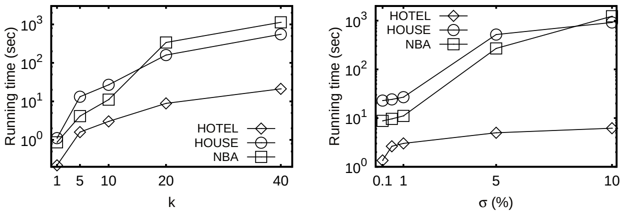
(c) Effect of n

(d) Effect of d

Figure 10: Performance of TAS* vs. data distribution

In Figure 10(b), we plot the running time of TAS* on different data distributions while varying σ. The running time increases at a faster rate on ANTI data, due to the factors explained in the previous paragraph. Figure 10(c) varies the dataset size n. For COR data, the running time is almost insensitive to n. Even for ANTI data, the running time grows sub-linearly; TAS* takes only 15.3 seconds for a dataset of 1.6 million options. Next, in Figure 10(d), we investigate the effect of dimensionality d for different data distributions. Although the running time increases with d, it remains practical even at the extremes tested.

Experiments on real datasets: We proceed by evaluating TAS* on the real datasets too, i.e., on HOTEL, HOUSE, and NBA, with dimensionality 4, 6, and 8, respectively. In Figure 11(a), we present the running time of TAS* when varying k. The comparison between HOUSE and NBA is the most interesting; when k is small (i.e., $k \leq 10$), TAS* performs slower on HOUSE because it is larger than NBA, yet for larger k, the situation is reversed due to the higher dimensionality of NBA. Using the same datasets, in Figure 11(b) we vary σ. Its effect is similar to Figure 10(b). Although we omit charts on space requirements for brevity, we note that TAS* has a very manageable footprint, requiring at maximum 648 MBytes in Figure 11(a), and 565 MBytes in Figure 11(b).



(a) Effect of k

(b) Effect of σ

Figure 11: Performance of TAS^* on real datasets

In Table 6, we compare the running time of TAS^* on real versus synthetic data. Each row corresponds to a real dataset, and to COR, IND, ANTI data of the same cardinality and dimensionality (for the default k and σ). Although the distribution of the real datasets does not necessarily follow a standard synthetic distribution, purely based on performance, HOTEL and HOUSE behave as slightly anticorrelated, while NBA as relatively correlated. However, they do not reach the extremes of ANTI and COR, respectively, which indicates that the synthetic benchmarks represent a broad spectrum of correlation degrees that encloses real data behavior.

Table 6: Comparison between real and synthetic datasets (TAS^*)

Dataset	Data characteristics	COR	IND	ANTI	Real
HOTEL	n = 418,843; d = 4	0.35s	2.76s	7.42s	3.02s
HOUSE	n = 315,265; d = 6	10.58s	21.27s	393.53s	26.87s
NBA	n = 21,960; d = 8	8.25s	101.23s	1068.21s	11.01s

In Table 7, we generate wR hyper-rectangles, where a (randomly chosen) side has length $\gamma \cdot \sigma$ (for a parameter γ), and the rest are equal but accordingly adjusted so that wR has the same volume as in the default setting. We vary γ from 0.25 to 4, and observe that TAS^* is not significantly affected. We hence deem our experiments for hyper-cubic wR (i.e., $\gamma = 1$) representative of the general case.

Table 7: Effect of wR elongation (TAS^*)

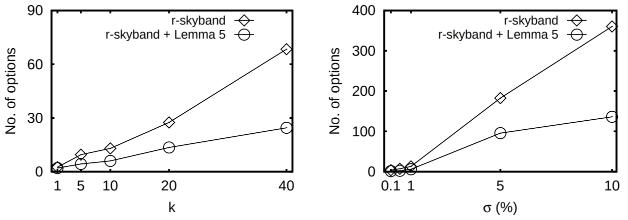
γ	HOTEL	HOUSE	NBA
0.25	3.05s	26.93s	11.06s
0.5	3.13s	26.81s	11.12s
1	3.02s	26.87s	11.01s
2	3.17s	28.29s	11.23s
4	3.22s	26.89s	11.10s

6.5 Effect of Optimizations

The next few experiments examine the individual effectiveness of the optimizations in Section 5.

First, we consider Section 5.1 and assess the power of Lemma 5, by comparing the number of options in $\mathcal{D}'$ after filtering the dataset only by r-skyband, versus r-skyband+Lemma 5. We use the default dataset (IND) for the default n and d. Figures 12(a) and 12(b) show the average $|\mathcal{D}'|$ when varying k and σ, respectively. Pruning by r-skyband+Lemma 5 is much more effective than r-skyband alone. At $k = 40$, for example, r-skyband+Lemma 5 produces a set $\mathcal{D}'$ that is 2.8 times smaller than r-skyband, i.e., Lemma 5 lets through 2.8 times fewer options for subsequent processing.

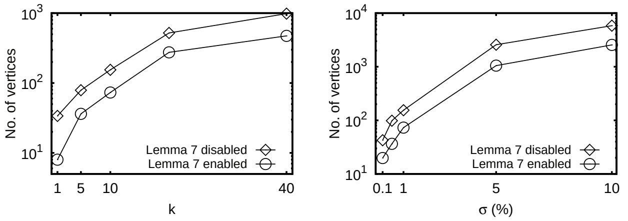
Next, we consider the optimized region testing in Section 5.2. We compare TAS^* with Lemma 7 enabled, versus TAS^* with Lemma 7 disabled. Figures 13(a) and 13(b) plot $|\mathcal{V}_{all}|$ (i.e., the final number of vertices in $\mathcal{V}_{all}$) when varying k and σ, respectively. Lemma 7 reduces $|\mathcal{V}_{all}|$ by 1.9 to 4.2 times, indicating that it indeed avoids the unnecessary splitting of many non-kIPR regions.



(a) Varying k

(b) Varying σ

Figure 12: Effect of pruning consistent top-scorers (Section 5.1)



(a) Varying k

(b) Varying σ

Figure 13: Effect of optimized region testing (Section 5.2)

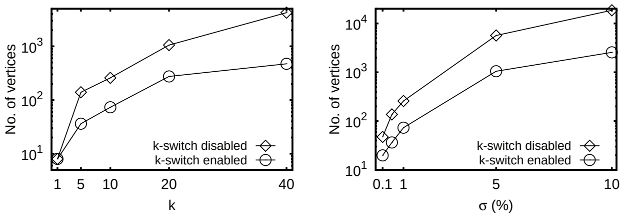
Finally, we investigate the effectiveness of the splitting hyperplane selection technique in Section 5.3. We compare two versions of TAS^*, with the k-switch strategy enabled/disabled. Figures 14(a) and 14(b) plot $|\mathcal{V}_{all}|$ for different values of k and σ, respectively. The results reveal that the k-switch strategy leads to 8.9 times fewer vertices in $\mathcal{V}_{all}$, attesting to its effectiveness in avoiding poor splitting, and in turn, reducing the number of splits performed.

7. CONCLUSION

In this paper, we introduce the *top-ranking region* problem (TopRR), which computes the maximal region oR in option space where any new option $\boldsymbol{o}$ would rank among the top-k for every weight vector in a target preference region wR. The problem finds application in competitive product placement and competitiveness enhancement. We propose a test-and-split approach, enhanced with a suite of powerful optimizations, and demonstrate the practicality and effectiveness of our techniques on large benchmark datasets. As future work, we plan to develop pre-computation techniques to further expedite processing, as well as to explore parallelism.

8. ACKNOWLEDGMENTS

Bo Tang was supported by the Science and Technology Innovation Committee Foundation of Shenzhen (Grant No. JCYJ20180302174301157) and the National Science Foundation of China (NSFC No. 61802163). Kyriakos Mouratidis was supported by the Singapore Management University Lee Kong Chian Fellowship. Man Lung Yiu was supported by grant P0009703 (YBXT) from the Hong Kong Polytechnic University.



(a) Varying k

(b) Varying σ

Figure 14: Effect of k-switch hyperplane selection (Section 5.3)

9. REFERENCES

[1] *Hotel dataset*, 2018. www.hotels-base.com.

[2] *House dataset*, 2018. www.ipums.org.

[3] *Laptop dataset*, 2018. www.cnet.com.

[4] *NBA dataset*, 2018. www.basketball-reference.com.

[5] P. K. Agarwal, B. Aronov, T. M. Chan, and M. Sharir. On levels in arrangements of lines, segments, planes, and triangles. *Discrete & Computational Geometry*, 19(3):315–331, 1998.

[6] A. Asudeh, A. Nazi, N. Koudas, and G. Das. Assisting service providers in peer-to-peer marketplaces: Maximizing gain over flexible attributes. *arXiv preprint arXiv:1705.03028*, 2017.

[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] M. d. Berg, O. Cheong, M. v. Kreveld, and M. Overmars. *Computational geometry: algorithms and applications*. Springer-Verlag TELOS, 2008.

[9] K. S. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft. When is "nearest neighbor" meaningful? In *ICDT*, pages 217–235, 1999.

[10] S. Börzsönyi, D. Kossmann, and K. Stocker. The skyline operator. In *ICDE*, pages 421–430, 2001.

[11] 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.

[12] B. Chazelle. An optimal convex hull algorithm in any fixed dimension. *Discrete & Computational Geometry*, 10(4):377–409, 1993.

[13] S. Chester, A. Thomo, S. Venkatesh, and S. Whitesides. Computing k-regret minimizing sets. *PVLDB*, 7(5):389–400, 2014.

[14] P. Ciaccia and D. Martinenghi. Reconciling skyline and ranking queries. *PVLDB*, 10(11):1454–1465, 2017.

[15] G. Das, D. Gunopulos, N. Koudas, and N. Sarkas. Ad-hoc top-k query answering for data streams. In *VLDB*, pages 183–194, 2007.

[16] M. Das, G. Das, and V. Hristidis. Leveraging collaborative tagging for web item design. In *KDD*, pages 538–546, 2011.

[17] R. Fagin, A. Lotem, and M. Naor. Optimal aggregation algorithms for middleware. *JCSS*, 66(4):614–656, 2003.

[18] 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.

[19] 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.

[20] P. Godfrey. Skyline cardinality for relational processing. In *FoIKS*, pages 78–97, 2004.

[21] Z. He and E. Lo. Answering why-not questions on top-k queries. *IEEE Trans. Knowl. Data Eng.*, 26(6):1300–1315, 2014.

[22] 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.

[23] 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), 2008.

[24] J. Koh, C. Lin, and A. L. P. Chen. Finding k most favorite products based on reverse top-t queries. *VLDB J.*, 23(4):541–564, 2014.

[25] C. Li, B. C. Ooi, A. K. H. Tung, and S. Wang. DADA: a data cube for dominant relationship analysis. In *SIGMOD Conference*, pages 659–670, 2006.

[26] Q. Liu, Y. Gao, G. Chen, B. Zheng, and L. Zhou. Answering why-not and why questions on reverse top-k queries. *VLDB J.*, 25(6):867–892, 2016.

[27] H. Lu and C. S. Jensen. Upgrading uncompetitive products economically. In *ICDE*, pages 977–988, 2012.

[28] M. Miah, G. Das, V. Hristidis, and H. Mannila. Determining attributes to maximize visibility of objects. *IEEE Trans. Knowl. Data Eng.*, 21(7):959–973, 2009.

[29] 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.

[30] K. Mouratidis and B. Tang. Exact processing of uncertain top-k queries in multi-criteria settings. *PVLDB*, 11(8):866–879, 2018.

[31] K. Mouratidis, J. Zhang, and H. Pang. Maximum rank query. *PVLDB*, 8(12):1554–1565, 2015.

[32] D. Nanongkai, A. D. Sarma, A. Lall, R. J. Lipton, and J. J. Xu. Regret-minimizing representative databases. *PVLDB*, 3(1):1114–1124, 2010.

[33] V. Padmanabhan, S. Rajiv, and K. Srinivasan. New products, upgrades, and new releases: A rationale for sequential product introduction. *Journal of Marketing Research*, 34(4):456–472, 1997.

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

[35] P. Peng and R. C. Wong. k-hit query: Top-k query with probabilistic utility function. In *SIGMOD Conference*, pages 577–592, 2015.

[36] Y. Peng, R. C. Wong, and Q. Wan. Finding top-k preferable products. *IEEE Trans. Knowl. Data Eng.*, 24(10):1774–1788, 2012.

[37] L. Qian, J. Gao, and H. V. Jagadish. Learning user preferences by adaptive pairwise comparison. *PVLDB*, 8(11):1322–1333, 2015.

[38] R. T. Rockafellar. Lagrange multipliers and optimality. *SIAM Review*, 35(2):183–238, 1993.

[39] R. Simons. Choosing the right customer. *Harvard Business Review*, 92(3):48–55, 2014.

[40] 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.

[41] B. Tang, K. Mouratidis, and M. L. Yiu. Determining the impact regions of competing options in preference space. In *SIGMOD Conference*, pages 805–820, 2017.

[42] Y. Tao, V. Hristidis, D. Papadias, and Y. Papakonstantinou. Branch-and-bound processing of ranked queries. *Inf. Syst.*, 32(3):424–445, 2007.

[43] J. Viner. Cost curves and supply curves. *Journal of Economics*, 3(1):23–46, 1931.

[44] A. Vlachou, C. Doulkeridis, Y. Kotidis, and K. Nørvåg.

Monochromatic and bichromatic reverse top-k queries. *IEEE Trans. Knowl. Data Eng.*, 23(8):1215–1229, 2011.

[45] A. Vlachou, C. Doulkeridis, K. Nørvåg, and Y. Kotidis. Identifying the most influential data objects with reverse top-k queries. *PVLDB*, 3(1):364–372, 2010.

[46] A. Vlachou, C. Doulkeridis, K. Norvag, and Y. Kotidis. Branch-and-bound algorithm for reverse top-k queries. In *SIGMOD Conference*, pages 481–492, 2013.

[47] Q. Wan, R. C. Wong, I. F. Ilyas, M. T. Özsu, and Y. Peng. Creating competitive products. *PVLDB*, 2(1):898–909, 2009.

[48] 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.

[49] G. Yang and Y. Cai. Querying improvement strategies. In *EDBT*, pages 294–305, 2017.

[50] J. Yang, Y. Zhang, W. Zhang, and X. Lin. Influence based cost optimization on user preference. In *ICDE*, pages 709–720, 2016.

[51] A. Yu, P. K. Agarwal, and J. Yang. Processing a large number of continuous preference top-*k* queries. In *SIGMOD Conference*, pages 397–408, 2012.

[52] A. Yu, P. K. Agarwal, and J. Yang. Top-k preferences in high dimensions. *IEEE Trans. Knowl. Data Eng.*, 28(2):311–325, 2016.

[53] V. A. Zeithaml, R. T. Rust, and K. N. Lemon. The customer pyramid: creating and serving profitable customers. *California Management Review*, 43(4):118–142, 2001.

[54] J. Zhang, K. Mouratidis, and H. Pang. Global immutable region computation. In *SIGMOD Conference*, pages 1151–1162, 2014.

[55] Z. Zhang, C. Jin, and Q. Kang. Reverse k-ranks query. *PVLDB*, 7(10):785–796, 2014.

[56] Z. Zhang, Y. Yang, R. Cai, D. Papadias, and A. K. H. Tung. Kernel-based skyline cardinality estimation. In *SIGMOD Conference*, pages 509–522, 2009.