Efficient Algorithms for k-Regret Minimizing Sets*

Pankaj K. Agarwal[1], Nirman Kumar[2], Stavros Sintos[3], and Subhash Suri[4]

1 **Department of Computer Science, Duke University, Durham, NC, USA**
pankaj@cs.duke.edu
2 **Department of Computer Science, University of Memphis, Memphis, TN, USA**
nkumar8@memphis.edu
3 **Department of Computer Science, Duke University, Durham, NC, USA**
ssintos@cs.duke.edu
4 **Department of Computer Science, UC Santa Barbara, Santa Barbara, CA, USA**
suri@cs.ucsb.edu

Abstract

A regret minimizing set Q is a small size representation of a much larger database P so that user queries executed on Q return answers whose scores are not much worse than those on the full dataset. In particular, a *k-regret minimizing set* has the property that the regret ratio between the score of the top-1 item in Q and the score of the top-k item in P is minimized, where the score of an item is the inner product of the item's attributes with a user's weight (preference) vector. The problem is challenging because we want to find a *single* representative set Q whose regret ratio is small with respect to *all possible* user weight vectors.

We show that k-regret minimization is NP-Complete for all dimensions $d \geq 3$, settling an open problem from Chester et al. [VLDB 2014]. Our main algorithmic contributions are two approximation algorithms, both with provable guarantees, one based on coresets and another based on hitting sets. We perform extensive experimental evaluation of our algorithms, using both real-world and synthetic data, and compare their performance against the solution proposed in [VLDB 14]. The results show that our algorithms are significantly faster and scalable to much larger sets than the greedy algorithm of Chester et al. for comparable quality answers.

1998 ACM Subject Classification H.2.8 Database Applications

Keywords and phrases regret minimizing sets, skyline, top-k query, coreset, hitting set

Digital Object Identifier 10.4230/LIPIcs.SEA.2017.7

1 Introduction

Multi-criteria decision problems pose a unique challenge for databases systems: how to present the space of possible answers to a user. In many instances, there is no single best answer, and often a very large number of incomparable objects satisfy the user's query. For instance, a database query for a car or a smart phone can easily produce an overwhelming number of potential choices to present to the user, with no obvious way to rank them. Top-k and the skyline operators are among the two main techniques used in databases to manage this kind of complexity, but each has its own shortcoming.

* Work by Agarwal and Sintos is supported by NSF under grants CCF-15-13816, CCF-15-46392, and IIS-14-08846, by ARO grant W911NF-15-1-0408, and by Grant 2012/229 from the U.S.-Israel Binational Science Foundation. Work by Suri and Kumar is supported by NSF under grant CCF-15-25817.

The top-k operator relies on the existence of a *utility function* that is used to rank the objects satisfying the user's query, and then selecting the top k by score according to this function. A commonly used utility function takes the inner product of the object attributes with a *weight vector*, also called the *user's preference*, thus forming a weighted linear combination of the different features. However, formulating the utility function is complicated, as users often do not know their preferences precisely, and, in fact, exploring the cost-benefit tradeoffs of different features is often the goal of database search.

The second approach of skylines is based on the principle of *pareto optimality*: if an object p is better than another object q on all features, then p is always preferable to q by any rational decision maker. This coordinate-wise dominance is used to eliminate all objects that are dominated by some other object. The *skyline* is the set of objects not dominated by any other object, and has proved to be a powerful tool in multi-criteria optimization. Unfortunately, while skylines are extremely effective in reducing the number of objects in low dimensions, their utility drops off quickly as the dimension (number of features) grows, especially when objects in the database have anti-correlated features (attributes). A related construct called k-skybands [15, 27] grows even more rapidly.

Regret minimization is a recent approach, proposed initially by Nanongkai et al. [31], to address the shortcomings of both the top k and skylines. It hybridizes top k and skylines by computing a small representative subset Q of the much larger database P so that *for any preference vector* the top ranked item in Q is a good approximation of the top ranked item in P. The hope is that the size of Q is much smaller than that of the skyline of P. The goal is to find a subset Q of small size whose approximation error is also small: posed in the form of a decision question, is there a subset of r objects so that every user's top-1 query can be answered within error at most $x\%$? In general, this is too stringent a requirement and motivated Chester et al. [12] to propose a more relaxed version of the problem, called the k-regret minimization.[1] In k-regret minimization, the quality of approximation is measured as the gap between the score of the top 1 item in Q and the top k item in P expressed as a ratio, so that the value is always between 0 and 1.

Problem formulation. An object is represented as a point $p = (p_1, \ldots, p_d)$ in $\mathbb{R}^d$ with non-negative attributes, i.e., $p_i \geq 0$ for every $i \leq d$. Let $\mathbb{X} = \{(p_1, \ldots, p_d) \in \mathbb{R}^d \mid p_i \geq 0 \ \forall i\}$ denote the space of all objects, and let $\mathsf{P} \subset \mathbb{X}$ be a set of n objects. A user preference is also represented as a point $u = (u_1, \ldots, u_d) \in \mathbb{X}$, i.e., all $u_i \geq 0$. Given a preference $u \in \mathbb{R}^d$, we define the *score* of an object p to be $\omega(u, p) = \langle u, p \rangle = \sum_{i=1}^{d} u_i p_i$.

For a preference $u \in \mathbb{X}$ and an integer $k \geq 1$, let $\varphi_k(u, \mathsf{P})$ denote the point p in P with the k-th largest score (i.e., there are less than k points of P with larger score than $\omega(u, p)$ and there are at least k points with score at least $\omega(u, p)$), and let $\omega_k(u, \mathsf{P})$ denote its score. Set $\Phi_k(u, \mathsf{P}) = \{\varphi_j(u, \mathsf{P}) \mid 1 \leq j \leq k\}$ to be the set of k top points with respect to preference u.[2] For brevity, we set $\omega(u, \mathsf{P}) = \omega_1(u, \mathsf{P})$. If P is obvious from the context, we drop P from the list of the arguments, i.e., we use $\omega_k(u)$ to denote $\omega_k(u, \mathsf{P})$ and so on.

[1] We should point out that the term k-regret is used to denote different things by Nanongkai et al. [31] and Chester et al. [12]. In the former, k-regret is the representative set of k objects, whereas in the latter, k-regret is used to denote the regret ratio between the scores of top 1 and top k. In our paper, we follow the convention of Chester et al. [12].

[2] If there are multiple objects with score $\omega_j(u, \mathsf{P})$, then either we include all such points in $\Phi_k(u, \mathsf{P})$ or break the tie in a consistent manner.

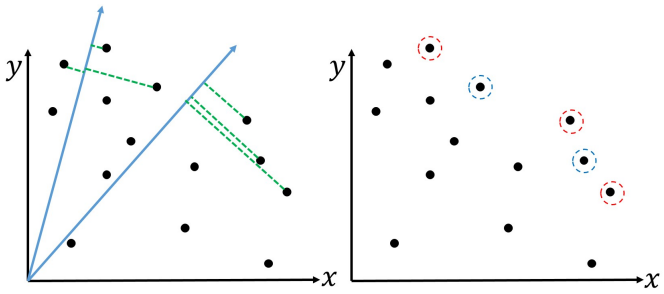


Figure 1 Left: top 3 points in two different preferences. Right: Set of points in the red circles is a $(1, 0)$-regret set. Set of points in the blue circles is a $(3, 0)$-regret set.

For a subset $\mathsf{Q} \subseteq \mathsf{P}$ and a preference u, define the regret of Q for preference u (w.r.t. P), denoted by $\ell_k(u, \mathsf{Q}, \mathsf{P})$, as

$$\ell_k(u, \mathsf{Q}, \mathsf{P}) = \frac{\max\{0, \omega_k(u, \mathsf{P}) - \omega(u, \mathsf{Q})\}}{\omega_k(u, \mathsf{P})}.$$

That is, $\ell_k(u, \mathsf{Q}, \mathsf{P})$ is the relative loss in the score of the k-th topmost object if we replace P with Q. We refer to the maximum regret of Q, $\ell_k(\mathsf{Q}, \mathsf{P}) = \max_{u \in \mathbb{X}} \ell_k(u, \mathsf{Q}, \mathsf{P})$ as the *regret ratio* of Q (w.r.t. P). If $\ell_k(\mathsf{Q}) \leq \epsilon$, we refer to Q as a (k, ϵ)-regret set (see Figure 1). By definition, a (k, ϵ)-regret set is also a (k', ϵ)-regret set for any $k' \geq k$. In particular, a $(1, \epsilon)$-regret set is a (k, ϵ)-regret set for any $k \geq 1$. However, there may exist a (k, ϵ)-regret set whose size is much smaller than any $(k-1, \epsilon)$-regret set, so the notion of (k, ϵ)-regret set is useful for all k.

Notice that $\ell_k(\mathsf{Q})$ is a monotonic decreasing function of its argument, i.e., if $\mathsf{Q}_1 \subseteq \mathsf{Q}_2$, then $\ell_k(\mathsf{Q}_1) \geq \ell_k(\mathsf{Q}_2)$. Furthermore, for any $t > 0$, $\omega(tu, p) = t\omega(u, p)$ but $\varphi_k(tu, \mathsf{P}) = \varphi_k(u, \mathsf{P})$, $\Phi_k(tu, \mathsf{P}) = \Phi_k(u, \mathsf{P})$, and $\ell_k(tu, \mathsf{Q}, \mathsf{P}) = \ell_k(u, \mathsf{Q}, \mathsf{P})$ (scale invariance).

Our goal is to compute a small subset $\mathsf{Q} \subseteq \mathsf{P}$ with small regret ratio, which we refer to as the *k-regret minimizing set* (k-RMS) problem[3]. Since the regret ratio can be decreased by increasing the size of the subset, there are two natural formulations of the RMS problem.

- **(i)** min-*error*: Given a set P of objects and a positive integer r, compute a subset of P of size r that minimizes the regret ratio, i.e., return a subset $\mathsf{Q}^* = \text{argmin}_{\mathsf{Q} \subseteq \mathsf{P}: |\mathsf{Q}| \leq r} \ell_k(\mathsf{Q})$, where we define $\ell_r = \ell_k(\mathsf{Q}^*)$.
- **(ii)** min-*size*: Given a set P of objects and a parameter $\epsilon > 0$, compute a smallest size subset with regret ratio at most ϵ, i.e., return $\mathsf{Q}^{\#} = \text{argmin}_{\mathsf{Q} \subseteq \mathsf{P}: \ell_k(\mathsf{Q}) \leq \epsilon} |\mathsf{Q}|$, and set $s_\epsilon = |\mathsf{Q}^{\#}|$.

Our results. The main results of our paper can be summarized as follows:

- **(I)** In Section 2, we show that the RMS problem is NP-Complete for all dimensions $d \geq 3$ and $k > 1$, answering an open problem from [12]. Previously, the problem was known to be solvable in polynomial time for $d = 2$ and intractable for $d = \Omega(n)$.
- **(II)** In Section 3, we present a coreset-based universal approximation, which shows that every $\mathsf{P} \subset \mathbb{X}$ admits an $O(\frac{1}{\epsilon^{(d-1)/2)}})$ size $(1, \epsilon)$-regret set, and thus also a (k, ϵ)-regret set, for any $k \geq 1$ and $\epsilon > 0$. The size of this regret-set is independent of the size of P, it can be computed in time $O(n + \frac{1}{\epsilon^{d-1}})$, and it can be dynamically updated per point insertion and deletion in time $O(\frac{\text{polylog}(n)}{\epsilon^{d-1}})$.

[3] We will refer to the k-RMS problem simply as RMS problem.

(III) In Section 4, we present an instance-specific approximation scheme, complementing the NP-Completeness of regret-set minimization. This is significant because the size of (k, ϵ)-regret set for a generic P can be much smaller than the coreset-based bound of $1/\epsilon^{\frac{d-1}{2}}$. In particular, our algorithm computes a $(k, 2\epsilon)$-regret set of P[4] whose size is within a log-factor of the optimal (k, ϵ)-regret set. With binary search, we can use this algorithm to also approximate the min-error version of the problem.

(IV) In Section 5, we describe our experimental results and evaluate the efficacy and the efficiency of our algorithms on both synthetic and real data sets. We compare our algorithms with the state of the art greedy algorithm for the k-regret minimization problem presented in [12]. Our hitting-set based algorithm is significantly faster than the previous known algorithms and the maximum regret ratios of the returned sets are very close, if not better, than the maximum regret ratios of the greedy algorithm. The coreset algorithm is significantly faster than hitting set and greedy algorithms. Although the (maximum) regret ratio of the set returned by the core-set based algorithm is worse than those of other algorithms, the regret in 90%–95% directions is roughly the same as that of the other two algorithms.

Remarks. While preparing our submission, we learned of two recent and independent discoveries with partial overlap with our work [8, 4]. In [8] the authors prove that RMS problem is NP-Hard for $k \geq 1$ and $d \geq 3$, and describe a coreset-based approximation algorithm. We present a simpler proof to show that RMS problem is NP-hard for $k \geq 2$ and $d \geq 3$. In addition, we show that RMS problem is NP-Complete which is not straightforward. Our coreset-based algorithm is related to their result, however, we first implemented it and run experiments for the RMS problem comparing the results with other competitive algorithms. In [4] the authors describe an efficient algorithm for 1-RMS problem in 2-d and a near-linear time approximation for the 1-RMS problem in higher dimensions, along with experimental evaluation. We give a more general, randomized approximation algorithm for the k-RMS problem with better approximation factor, and same running time with [4] up to logarithmic factors with high probability.

2 3D RMS is NP-Complete

In this section we show that the k-RMS problem is NP-Complete for $d = 3$ and $k \geq 2$.

Membership in NP. It turns out that due to bit-complexity issues, even establishing membership in NP is not straightforward in $d = 3$, and requires some non-trivial ideas. The starting point of our proof is a polynomial-time algorithm for computing the regret ratio of a subset $\mathsf{Q} \subseteq \mathsf{P}$.

Let $\Omega = \{p - q \mid p, q \in \mathsf{P}, p \neq q\}$ be the set of vectors in directions passing through a pair of points of P. For a vector $w \in \Omega$, let $\mathsf{h}_w : \langle x, w \rangle = 0$, be the plane normal to w passing through the origin. By construction, for $w = p - q$ the score of p is higher than that of q for all preferences in one of the open halfspaces bounded by h_w (namely, $\langle x, w \rangle > 0$), lower in the other halfspace, and equal for all preferences in h_w. Set $\mathsf{H} = \{\mathsf{h}_w \mid w \in \Omega\} \cup \{x_i = 0 \mid 1 \leq i \leq 3\}$, i.e., H includes all the planes h_w along with the coordinate planes. H induces a decomposition $A(\mathsf{H})$ of $\mathbb{R}^3$ into cells of various dimensions, where each cell is a maximal

[4] The approximation ratio 2 is not important. We can actually compute a $(k, t\epsilon)$-regret set for an arbitrary small constant $t > 1$.

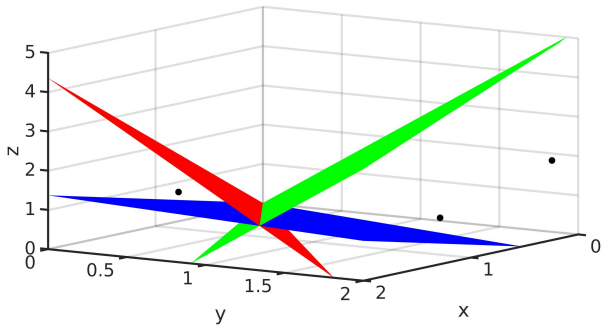


Figure 2 H for a set of 3 points in $\mathbb{R}^3$.

connected region of points lying in the same subset of hyperplanes of H (see Figure 2). It is well known that

(i) each cell of $A(\mathsf{H})$ is a polyhedral cone with the origin as its apex (i.e., each cell is the convex hull of a finite set of rays, each emanating from the origin), and

(ii) the only 0-dimensional cell of $A(\mathsf{H})$ is the origin itself, and the 1-dimensional cells are rays emanating from the origin. Let $\mathcal{C} \subseteq A(\mathsf{H})$ be the set of cells that lie in $\mathbb{X}$, the positive orthant.

For each cell $C \in \mathcal{C}$, let $\ell_k(C, \mathsf{Q}) = \max_{u \in C} \ell_k(u, \mathsf{Q})$ the regret ratio of Q within C. Then $\ell_k(\mathsf{Q}) = \max_{C \in \mathcal{C}} \ell_k(C, \mathsf{Q})$. The following lemma is useful in computing $\ell(C, \mathsf{Q})$.

▶ **Lemma 1.** *For each cell $C \in A(\mathsf{H})$ and for any $i \leq n$, $\varphi_i(u, \mathsf{P})$ (and thus $\varphi_i(u, \mathsf{Q})$) is the same for all $u \in C$.*

Proof. Suppose on the contrary, there are two points $u_1, u_2 \in C$, and $j \geq 0$ such that $\varphi_j(u_1, \mathsf{Q}) \neq \varphi_j(u_2, \mathsf{Q})$. Hence, there are two points p_1, $p_2 \in \mathsf{Q}$ such that $\langle u_1, p_1 \rangle \geq \langle u_1, p_2 \rangle$ and $\langle u_2, p_1 \rangle \leq \langle u_2, p_2 \rangle$, and at least one of the inequalities is strict. Let $\mathsf{h}_w \in \mathsf{H}$ be the plane that is normal to $p_1 - p_2$ and passes through the origin. It divides $\mathbb{R}^3$ into two halfspaces. Reference vectors u_1, u_2 lie in the opposite halfspaces of h_w, and at least one of the u_1, u_2 lies in the open halfspace. However, this is a contradiction because u_1, u_2 lie in the same cell of $A(\mathsf{H})$ and thus lie on the same side of each plane in H. ◀

Fix a cell C. Let $p_i = \varphi_1(u, \mathsf{Q})$ and $p_j = \varphi_k(u, \mathsf{P})$ for any $u \in C$ (from Lemma 1 we have that the ordering inside a cell is the same). Furthermore, let h_j be the plane $\langle x, p_j \rangle = 1$ and let $C^{\downarrow} = \mathsf{h}_j \cap C$. $C^{\downarrow}$ is a 2D polygon and each ray ρ in C intersects $C^{\downarrow}$ at exactly one point $\rho^{\downarrow}$. Since $\ell(u, \mathsf{Q})$ is the same for all points on ρ, $\ell_k(C, \mathsf{Q}) = \ell_k(C^{\downarrow}, \mathsf{Q})$. Furthermore, by Lemma 1, $\ell_k(C^{\downarrow}, \mathsf{Q})$ is either 0 for all $u \in C^{\downarrow}$ or

$$\ell_k(C^{\downarrow}, \mathsf{Q}) = \max_{u \in C^{\downarrow}} \frac{\omega(u, p_j) - \omega(u, p_i)}{\omega(u, p_j)} = \max_{u \in C^{\downarrow}} [1 - \omega(u, p_i)] = 1 - \min_{u \in C^{\downarrow}} \langle u, p_i \rangle.$$

Since $C^{\downarrow}$ is convex and $\langle u, p_i \rangle$ is a linear function, it is a minimum within $C^{\downarrow}$ at a vertex of $C^{\downarrow}$, so we compute $\langle u, p_i \rangle$ for each vertex $u \in C^{\downarrow}$ and choose the one with the minimum value. Repeating this step for all cells of $\mathcal{C}$ we compute $\ell_k(\mathsf{Q})$.

By a well known result in discrete geometry [3], the total number of vertices in $C^{\downarrow}$ over all cells $C \in \mathcal{C}$ is $O(|\mathsf{H}|^2) = O(n^4)$. Furthermore, if b bits are used to represent the coordinates of each point in P, each vertex of $C^{\downarrow}$ requires $O(b)$ bits. Finally, the algorithm extends to higher dimensions in a straightforward manner. The total running time in $\mathbb{R}^d$ is $O(n^{2d-1})$. We thus conclude the following.

▶ **Lemma 2.** *The RMS problem is in* NP.

NP-Hardness Reduction. We first show the hardness for $k = 2$ and $\epsilon = 0$, which is easily extended to other values of k, ϵ. Recall that a preference vector has only non-negative coordinates. For simplicity, however, we first consider all points in $\mathbb{R}^3$ as preference vectors and define $\ell_k(\mathsf{Q}) = \max_{u \in \mathbb{R}^3} \ell_k(u, \mathsf{Q})$, and later we describe how to restrict the preference vectors to $\mathbb{X}$.

Recall that the RMS problem for $\epsilon = 0$ and $k = 2$ asks: Is there a subset $\mathsf{Q} \subseteq \mathsf{P}$ of size r such that in every direction u, the point in Q with the highest score along u, i.e., $\varphi_1(u, \mathsf{Q})$, has score at least as much as that of the second best in P along u, i.e., of the point $\varphi_2(u, \mathsf{P})$?

Let Π be a strictly convex polytope in $\mathbb{R}^3$. The 1-*skeleton* of Π is the graph formed by the vertices and edges of Π. Given Π and an integer $r > 0$, the *convex-polytope vertex-cover* (CPVC) asks whether the 1-skeleton of Π has a vertex cover of size at most r, i.e., whether there is a subset C of vertices of Π of size r such that every edge is incident on at least one vertex of C. The CPVC problem is NP-Complete, as shown by Das and Goodrich [13]. Given Π with V as the set of its vertices, we construct an instance of the RMS problem for $k = 2$, as follows. First we translate Π so that the origin lies inside Π. Next we set $\mathsf{P} = V$. The next lemma proves the NP-hardness of the RMS problem for $k = 2$ and $\epsilon = 0$.

▶ **Lemma 3.** *$\mathsf{Q} \subseteq V$ is a vertex cover of Π if and only if Q is a $(2, 0)$-regret set for P.*

Proof. If Q is a vertex cover of Π, we show that Q is also a $(2, 0)$-regret set. Take a vector $u \in \mathbb{R}^3$ and assume that $q = \varphi_1(u, \mathsf{P})$ (if there is more than one point with rank one, we can let q be any one of them). If $q \in \mathsf{Q}$ then obviously $\omega_1(u, \mathsf{Q}) = \omega_1(u, \mathsf{P}) \geq \omega_2(u, \mathsf{P})$. Now, assume that $q \notin \mathsf{Q}$. Let $(q, q_1), \ldots, (q, q_g)$ be the edges in Π incident on q. Set $N_q = \{q_i \mid 1 \leq i \leq g\}$. Since Q is a vertex cover of Π and $q \notin \mathsf{Q}$, $N_q \subseteq \mathsf{Q}$. We claim that $\varphi_2(u, \mathsf{P}) \in N_q$, which implies that $\omega(u, \mathsf{Q}) \geq \omega_2(u, \mathsf{P})$. Hence, Q is a $(2, 0)$-regret set.

Indeed, since Π is convex, and q is maximal along direction u, the plane h on q vertical to u is a supporting hyperplane for Π. A plane h' parallel to h is translated toward the origin starting with its initial position at h. There are two cases. In the first case, where q and $\varphi_2(u, \mathsf{P})$ have the same score, they belong to the same face of Π that must be contained in h itself – in this case h also contains a point from N_q, since every face containing q and points other than q must contain a 1 dimensional face as well, and therefore a point in N_q. In the second case, as h' is translated, it must first hit one of the neighbors of q, by convexity. As a result, in any case, there will be a point in N_q that gives the rank-two point on u.

Next, if Q is a $(2, 0)$-regret set, we show that Q is a vertex cover of Π. Suppose to the contrary Q is not a vertex cover of Π, i.e., there is an edge (q_1, q_2) in Π but $q_1, q_2 \notin \mathsf{Q}$. Since Π is a strictly convex polytope, there is a plane h tangent to Π at the edge (q_1, q_2) that does not contain any other vertex of Π. If we take the direction u normal to h then $\Phi_2(u, \mathsf{P}) = \{q_1, q_2\}$. If $q_1, q_2 \notin \mathsf{Q}$ then $\omega_1(u, \mathsf{Q}) < \omega_2(u, \mathsf{P})$, which contradicts the assumption that Q is a $(2, 0)$-regret set of P. ◀

By applying an affine transformation to the polytope Π, described in Appendix A, we can show that the RMS problem is NP-hard even when preferences are restricted to $\mathbb{X}$.

Choosing $\epsilon > 0$. While the above suffices to prove the hardness of the RMS problem for $\epsilon = 0$, it is possible that when $\epsilon > 0$ the problem is strictly easier. However, we show the stronger result that the RMS problem is NP-hard even when ϵ is required to be strictly positive. In order to get the NP-hardness of the RMS problem for $\epsilon > 0$ and $k = 2$, we find a small enough strictly positive value of ϵ with bounded bit complexity such that any $(2, \epsilon)$-regret set is also a $(2, 0)$-regret set, and vice versa. For each cell $C \in \mathcal{C}$, we take a

direction $u_C \in C$ and let $\lambda_C = 1 - \omega_3(u_C, \mathsf{P})/\omega_2(u_C, \mathsf{P}) > 0$. By defining $\epsilon = \frac{1}{2}\min_C \lambda_c$ we can conclude the result.

Larger values of k. By making $k-1$ copies of each point in the above construction it is straightforward to show that the RMS problem is NP-complete for any $k \geq 2$ and $d \geq 3$.

▶ **Theorem 4.** *The RMS problem is* NP*-Complete for $d \geq 3$ and for $k \geq 2$.*

3 Coreset-based Approximation

In this section, we present an approximation scheme for the RMS problem using coresets. The general idea of a coreset is to approximately preserve some desired characteristics of the full data set using only a tiny subset [2]. The particular geometric characteristic most relevant to our problem is the *extent* of the input data in any direction, which can be formalized as follows. Given a set of points P and a direction $u \in \mathbb{R}^d$, the *directional width* of P along u, denoted $\text{width}(u, \mathsf{P})$, is the distance between the two supporting hyperplanes of $\mathbb{R}^d$, one in direction u and the other in direction $-u$. The connection between k-regret and the directional width comes from the fact that the supporting hyperplane in a direction u is defined by the extreme point in that direction, and its distance from the origin is simply its score. Therefore, we have the equality: $\text{width}(u, \mathsf{P}) = \omega(u, \mathsf{P}) + \omega(-u, \mathsf{P})$.

We use coresets that approximate directional width to approximate k-regret sets. In particular, a subset $\mathsf{Q} \subseteq \mathsf{P}$ is called an *ϵ-kernel* coreset if $\text{width}(u, \mathsf{Q}) \geq (1-\epsilon)\,\text{width}(u, \mathsf{P})$, for all directions $u \in \mathbb{R}^d$. Showing that an ϵ-kernel coreset of P is also an $(1, \epsilon)$-regret set of P and using the results of [1, 10] to compute small ϵ-kernel coresets efficiently, we prove the following result.

▶ **Theorem 5.** *Given a set P of n points in $\mathbb{R}^d$, $\epsilon > 0$ and an integer $k > 0$, we can compute in time $O(n + \frac{1}{\epsilon^{d-1}})$ a subset $\mathsf{Q} \subseteq \mathsf{P}$ of size $O(\frac{1}{\epsilon^{(d-1)/2}})$ whose k-regret ratio is at most ϵ. The set Q can also be maintained under insertion/deletion of points in P in time $O(\frac{\log^d n}{\epsilon^{d-1}})$ per update.*

Proof. We first show that if $\mathsf{Q} \subseteq \mathsf{P}$ is an ϵ-kernel coreset of P then Q is also $(1, \epsilon)$-regret set of P. If Q is an ϵ-kernel coreset of P then $\text{width}(u, \mathsf{P}) - \text{width}(u, \mathsf{Q}) \leq \epsilon\,\text{width}(u, \mathsf{P}) \leq \epsilon\omega(u, \mathsf{P})$. The last inequality follows because $\omega(-u, \mathsf{P}) \leq 0$. Furthermore $\omega(-u, \mathsf{Q}) \leq \omega(-u, \mathsf{P})$. We thus have

$$\begin{aligned}
\omega(u, \mathsf{P}) - \omega(u, \mathsf{Q}) &= \omega(u, \mathsf{P}) + \omega(-u, \mathsf{P}) - \omega(u, \mathsf{Q}) - \omega(-u, \mathsf{P}) \\
&\leq \text{width}(u, \mathsf{P}) - \omega(u, \mathsf{Q}) - \omega(-u, \mathsf{Q}) \\
&= \text{width}(u, \mathsf{P}) - \text{width}(u, \mathsf{Q}) \\
&\leq \epsilon\omega(u, \mathsf{P}).
\end{aligned}$$

Hence, $\omega(u, \mathsf{Q}) \geq (1-\epsilon)\omega(u, \mathsf{P})$ and Q is an $(1, \epsilon)$-regret set of P. Chan [10] has described an algorithm for computing ϵ-kernel of size $O(\frac{1}{\epsilon^{(d-1)/2}})$ in time $O(n + \frac{1}{\epsilon^{d-1}})$. The dynamic update performance follows from the construction in [1]. ◀

By comparison, the algorithm in [31] computes a set Q with $\ell_1(Q) \leq \frac{d-1}{(r-d+1)\frac{1}{d-1}+d-1}$, which implies a $(1, \epsilon)$-regret set of size $O(\frac{1}{\epsilon^{d-1}})$. Thus, our result improves the bound of [31] significantly. In addition, if points in P lie uniformly on the unit sphere, then it is known that a valid ϵ-kernel Q can have size $\Omega(\frac{1}{\epsilon^{(d-1)/2}})$, and hence the result in [1] is optimal in the worst case. By a similar construction we can also show that the regret-set bound achieved in Theorem 5 is asymptotically optimal.

4 Regret Approximation using Hitting Sets

While Theorem 5 shows that every point set admits a (k, ϵ)-regret set of size $O(\frac{1}{\epsilon^{(d-1)/2}})$, a specific instance of P may have a much smaller regret set. In this section, we present an efficient scheme to approximate the optimal (k, ϵ)-regret set, by formulating the RMS problem as a hitting-set problem.

A range space (or set system) $\Sigma = (\mathsf{X}, \mathcal{R})$ consists of a set X of objects and a family $\mathcal{R}$ of subsets of X. A subset $H \subseteq \mathsf{X}$ is a hitting set of Σ if $H \cap R \neq \emptyset$ for all $R \in \mathcal{R}$. The hitting set problem asks to compute a hitting set of the minimum size. The hitting set problem is a classical NP-Complete problem, and a well-known greedy $O(\log n)$-approximation algorithm is known.

We construct a set system $\Sigma = (\mathsf{P}, \mathcal{R})$ such that a subset $\mathsf{Q} \subseteq \mathsf{P}$ is a (k, ϵ)-regret set if and only if Q is a hitting set of Σ. We then use the greedy algorithm to compute a small-size hitting set of Σ. A weakness of this approach is that the size of $\mathcal{R}$ could be very large and the greedy algorithm requires $\mathcal{R}$ to be constructed explicitly. Consequently, the approach is expensive even for moderate inputs say $d \sim 5$.

Inspired by the above idea, we propose a bicriteria approximation algorithm: given P and $\epsilon > 0$, we compute a subset $\mathsf{Q} \subseteq \mathsf{P}$ of size $O(s_\epsilon \log s_\epsilon)$ that is a $(k, 2\epsilon)$-regret set of P; the constant 2 is not important, it can be made arbitrarily small at the cost of increasing the running time. By allowing approximations to both the error and size concurrently, we can construct a much smaller range space and compute a hitting set of this range space.

The description of the algorithm is simpler if we assume the input to be well conditioned. We therefore transform the input set, without affecting an RMS, so that the score of the topmost point does not vary too much with the choice of preference vectors, i.e., the ratio $\frac{\max_{u \in \mathbb{X}} \omega(u, \mathsf{P})}{\min_{u \in \mathbb{X}} \omega(u, \mathsf{P})}$ is bounded by a constant that depends on d.

We transform P into another set P', so that (i) for any $u \in \mathbb{X}$, $\varphi_1(u, \mathsf{P}')$ does not lie close to the origin and (ii) for any (k, ϵ)-regret set $\mathsf{Q} \subseteq \mathsf{P}$, the corresponding subset $\mathsf{Q}' \subseteq \mathsf{P}'$ is a (k, ϵ)-regret set in P', and vice versa. The transformation is a non-uniform scaling of P. Nanongkai et al. [31] showed that such a scaling of P satisfies (ii). For each $1 \leq j \leq d$, let $m_j = \max_{p_i \in P} p_{ij}$ be the maximum value of the jth coordinate among all points. Let $B \subseteq \mathsf{P}$ contains the points corresponding to these m_j values. We refer to B as the *basis* of P. We divide the j-th coordinate of all points by m_j, for all $j = 1, 2, \ldots, d$. Let P' be the resulting set, and let B' be the transformation of B. We note that for each coordinate j there is a point $p'_i \in B'$ with $p'_{ij} = 1$. The different scaling factor in each coordinate can be represented by the diagonal matrix M where $M_{jj} = 1/m_j$, and so $\mathsf{P}' = M\mathsf{P}$. The key property of this affine transformation is the following lemma.

▶ **Lemma 6.** *Let M be the affine transformation described above and let $\mathsf{P}' = M\mathsf{P}$. Then, $\sqrt{d} \cdot \|u\| \geq \omega(u, \mathsf{P}') \geq \frac{1}{\sqrt{d}} \cdot \|u\|$, for all $u \in \mathbb{X}$.*

Proof. Since $\omega(\cdot, \cdot)$ is a linear function, without loss of generality consider a vector $u \in \mathbb{X}$ with $\|u\| = 1$. After the transformation M, for each coordinate j, we have $p'_j \leq 1$. Therefore, $\|p'\| \leq \sqrt{d}$ and also $\sqrt{d} \geq \omega(u, \mathsf{P}')$ because u is a unit vector. For the second inequality, we note that for any unit norm vector u we must have $u_j \geq \frac{1}{\sqrt{d}}$, for some j. Since our transform ensures the existence of a point $p' \in B'$ with $p'_j = 1$, we must have $\omega(u, \mathsf{P}') \geq \langle u, p' \rangle \geq \frac{1}{\sqrt{d}}$. This completes the proof. ◀

4.1 Approximation Algorithms

We first show how to formulate the min-size version of the RMS problem as a hitting set problem. Let P, k, and ϵ be fixed. For a vector $u \in \mathbb{X}$, let $R_u = \{p \in \mathsf{P} \mid \omega(u,p) \geq (1-\epsilon)\omega_k(u,p)\}$. Note that if $\epsilon = 0$, then $R_u = \Phi_k(u)$, the set of top-k points of P in direction u. Set $\mathcal{R}_u = \{R_u \mid u \in \mathbb{X}\}$. Although there are infinitely many preferences we show below that $|\mathcal{R}_u|$ is polynomial in $|\mathsf{P}|$. We now define the set system $\Sigma = (\mathsf{P}, \mathcal{R}_u)$.

▶ **Lemma 7.**
(i) $|\mathcal{R}_u| = O(n^d)$.
(ii) *A subset* $\mathsf{Q} \subseteq \mathsf{P}$ *is a hitting set of* Σ *if and only if* Q *is a* (k,ϵ)*-regret set of* P.

Proof.
(i) Note that R_u is a subset of P that is separated from $\mathsf{P} \setminus R_u$ by the hyperplane $\mathsf{h}_u : \langle u, x\rangle \geq (1-\epsilon)\omega_k(u,\mathsf{P})$. Such a subset is called linearly separable. A well-known result in discrete geometry [3] shows that a set of n points in $\mathbb{R}^d$ has $O(n^d)$ linearly separable subsets. This completes the proof of (i).
(ii) First, by definition any (k,ϵ)-regret set Q has to contain a point of R_u for all $u \in \mathbb{X}$ because otherwise $\ell_k(u,\mathsf{Q}) > \epsilon$. Hence, Q is a hitting set of Σ. Conversely, if $\mathsf{Q} \cap R_u \neq \emptyset$, then $\ell_k(u,\mathsf{Q}) \leq \epsilon$. If Q is a hitting set of Σ, then $\mathsf{Q} \cap R_u \neq \emptyset$ for all $u \in \mathbb{X}$, so Q is also a (k,ϵ)-regret set. ◀

We can thus compute a small-size (k,ϵ)-regret set of P by running the greedy hitting set algorithm on Σ. In fact, the greedy algorithm in [7] returns a hitting set of size $O(s_\epsilon \log s_\epsilon)$. As mentioned above, the challenge is the size of $\mathcal{R}_u$. Even for small values of k, $|\mathcal{R}_u|$ can be $\Omega(n^{\lfloor d/2 \rfloor})$ [3]. Next, we show how to construct a much smaller set system.

Recall that $\ell_k(u,\mathsf{Q})$ is independent of $\|u\|$ so we focus on unit preference vectors, i.e., we assume $\|u\| = 1$. For the analysis, we also assume that $\mathsf{P} = M\mathsf{P}$ from Lemma 6. Let $\mathbb{U} = \{u \in \mathbb{X} \mid \|u\| = 1\}$ be the space of all unit preference vectors; $\mathbb{U}$ is the portion of the unit sphere restricted to the positive orthant. For a given parameter $\delta > 0$, a set $\mathsf{N} \subset \mathbb{U}$ is called a *δ-net* if the spherical caps of radius δ around the points of N cover $\mathbb{U}$, i.e. for any $u \in \mathbb{U}$, there is a point $v \in \mathsf{N}$ with $\langle u, v\rangle \geq \cos(\delta)$. A δ-net of size $O(\frac{1}{\delta^{d-1}})$ can be computed by drawing a "uniform" grid on $\mathbb{U}$. In practice, it is simpler and more efficient to choose a random set of $O(\frac{1}{\delta^{d-1}} \log \frac{1}{\delta})$ directions – this will be a δ-net with probability at least $1/2$. Let N be a $\frac{\delta}{2d}$-net of $\mathbb{U}$, and let $\mathcal{R}_\mathsf{N} = \{R_u \mid u \in \mathsf{N}\}$.

Set $\Sigma_\mathsf{N} = (\mathsf{P}, \mathcal{R}_\mathsf{N})$. Note that $|\mathcal{R}_\mathsf{N}| = O(\frac{1}{\delta^{d-1}})$. Our main observation, stated in the lemma below and proven in Appendix B, is that it suffices to compute a hitting set of Σ_N. Recall that basis B of P is the subset of at most d points, one per coordinate, corresponding to the points with the highest value per coordinate.

▶ **Lemma 8.** *Let* Q' *be a hitting set of* Σ_N*, and let* B *be the basis of* P*. Then* $\mathsf{Q} = \mathsf{Q}' \cup B$ *is a* $(k, \epsilon + \delta - \delta\epsilon)$*-regret set of* P.

Algorithm 1 summarizes the algorithm. Greedy_HS is the greedy algorithm in [7] for computing a hitting set. Scale(P) is the procedure that scales the set P according to the transformation M in Lemma 6. Basis(P) is the method to find the basis B.

Analysis. The correctness of the algorithm follows from Lemma 8. Since a hitting set of Σ is also a hitting set of Σ_N, Σ_N has a hitting set of size at most s_ϵ. The greedy algorithm in [7] returns a hitting set of size $O(s_\epsilon \log s_\epsilon)$ for $d \geq 4$ and of size $O(s_\epsilon)$ for $d \leq 3$. Therefore $|\mathsf{Q}| = O(s_\epsilon \log s_\epsilon)$ for $d \geq 4$ and $O(s_\epsilon)$ for $d = 3$. Computing the set B takes $O(n)$ time. N

Algorithm 1 RMS_HS

Input: P: Input points, $k \geq 1$: rank, $\epsilon, \delta \in [0,1]$: error parameters.
Output: Q a $(k, \epsilon + \delta - \delta\epsilon)$-regret set

1: $B := \textsc{Basis}(\mathsf{P})$
2: $\mathsf{P} := \textsc{Scale}(\mathsf{P})$
3: $\mathsf{N} := \frac{\delta}{2d}$-net of $\mathbb{U}$
4: $R_u := \{p \in \mathsf{P} \mid \omega(u,p) \geq (1-\epsilon)\omega_k(u,\mathsf{P})\}$
5: $\mathcal{R}_{\mathsf{N}} := \{R_u \mid u \in \mathsf{N}\}$
6: $\mathsf{Q}' := \textsc{Greedy\_HS}(\mathsf{P}, \mathcal{R}_{\mathsf{N}})$
7: Return $\mathsf{Q} := \mathsf{Q}' \cup B$

can be constructed in $O(|\mathsf{N}|)$ time and we can compute R_u for each $u \in \mathsf{N}$ in $O(n)$ time. The greedy algorithm in [7] takes $O(\frac{n}{\delta^{d-1}} \log n \log \frac{1}{\delta})$ expected time (the bound on the running time also holds with high probability).

Putting everything together and setting $\delta = \epsilon$, we obtain the following:

▶ **Theorem 9.** *Let $\mathsf{P} \subset \mathbb{X}$ be a set of n points in $\mathbb{R}^d$, $k \geq 1$ an integer, and $\epsilon > 0$ a parameter. Let s_ϵ be the minimum size of a (k,ϵ)-regret set of P. A subset $\mathsf{Q} \subseteq \mathsf{P}$ can be computed in $O\left(\frac{n}{\epsilon^{d-1}} \log(n) \log\left(\frac{1}{\epsilon}\right)\right)$ expected time such that Q is a $(k, 2\epsilon)$-regret set of P. The size of Q is $O(s_\epsilon \log s_\epsilon)$ for $d \geq 4$ and $O(s_\epsilon)$ for $d \leq 3$.*

Notice that we can improve the running time of Theorem 9 to $O\left(\frac{n}{\epsilon^{d-1}}\right)$ using the simple greedy algorithm for the hitting set problem. In this case, the size of Q is $O(s_\epsilon \log \frac{1}{\epsilon})$.

min-error RMS. Recall that the min-error problem takes as input a parameter r, and returns a subset $\mathsf{Q} \subseteq \mathsf{P}$ of size at most r such that $\ell_k(\mathsf{Q}) \leq \ell_r$, where ℓ_r is the minimum regret ratio of a subset of P of size at most r. We propose a bicriteria approximation algorithm for the min-error problem using Algorithm 1.

Let $E = \{1 - \omega_j(u,\mathsf{P})/\omega_k(u,\mathsf{P}) \mid k < j \leq n, u \in \mathsf{N}\}$, and let $\epsilon_0 \in E$ be the smallest value such that $s_{\epsilon_0} \leq r$. Since N is a $\frac{\delta}{2d}$-net it can be shown (similar to the proof of Lemma 8) that $\varepsilon_0 \leq (1-\delta)\ell_r + \delta$, so we can solve the min-error problem approximately by performing a binary search on the values in E. However, testing whether $s_\epsilon \leq r$ for a given ϵ is hard, so we use an approximate decision procedure as follows: For a value $\epsilon \in E$, we run Algorithm 1. If it returns a subset of size larger than $cr \log r$, where $c > 0$ is an appropriate constant, we search among the values larger than ε, and among the smaller values otherwise. In the end, we return a set Q of size $O(r \log r)$. Notice that if $\epsilon > \ell_r$, then Algorithm 1 always returns a set of size less than $cr \log r$. The following theorem summarizes the results of the min-error version of the problem.

▶ **Theorem 10.** *Let $\mathsf{P} \subset \mathbb{X}$ be a set of n points in $\mathbb{R}^d$, $k \geq 1$ an integer, and $r > 0$, $0 < \delta < 1$ two parameters. A subset $\mathsf{Q} \subseteq \mathsf{P}$ can be computed in $O(\frac{n}{\delta^{d-1}} \log(n) \log\left(\frac{1}{\delta}\right) \log\left(\frac{n}{\delta}\right))$ expected time such that $\ell_k(\mathsf{Q}) \leq (1-\delta)\ell_r + \delta$. The size of Q is $O(r \log r)$ for $d \geq 4$ and $O(r)$ for $d \leq 3$.*

Table 1 Summary of datasets used in experiments.

ID	Description	d	n	Skyline
BB	Basketball	5	21961	200
ElNino	Oceanographic	5	178080	1183
Colors	Colors	9	68040	674
AntiCor	Anti-correlated points	4	10000	657
Sphere	Points on unit sphere	4	15000	15000
SkyPoints	Many points close to skyline	3	500	100

5 Experiments

We have implemented our algorithms as well as the current state of the art, namely, the greedy algorithms described in [31, 12], and experimentally evaluated their relative performance.[5]

Algorithms. In particular, the four algorithms we evaluate are the following:

RRS is the ***R***andomized ***R***egret ***S***et algorithm, based on coresets, described in Section 3. In our implementation, rather than choosing $O(\frac{1}{\epsilon^{(d-1)/2}})$ random preferences all at once, we choose them in stages and maintain a subset Q until $\ell_k(\mathsf{Q}) \leq \epsilon$.

HS is the ***H***itting ***S***et algorithm presented in Section 4. Notice that for $k = 1$, the optimum solution of the RMS problem will always be a subset of the skyline of P. Hence, to reduce the running time we only run the algorithm for $k = 1$ on skyline points. Furthermore, instead of choosing $O(\frac{1}{\epsilon^{d-1}})$ directions in one step and find a hitting set, we can sample in stages and maintain a hitting set until we find a (k, ϵ)-regret set.

NSLLX is the greedy algorithm for 1-RMS problem described in [31], which iteratively finds the preference u with the maximum regret using an LP algorithm and adds $\varphi_1(u, \mathsf{P})$ to the regret set. We use Gurobi software [17] to solve the LP problems efficiently. We remark that this algorithm, as a preprocessing step, removes all data points that are not on the skyline.

CTVW is the extension of the NSLLX algorithm for $k > 1$, proposed by [12], and it is the state of the art for the k-RMS problem. In [12] they discard all the points not on the skyline as preprocessing to run the experiments. The CTVW algorithm solves many (in the worst-case, $\Omega(n)$) instances of large LP programs to add the next point to the regret set. The number of LP programs is controlled by a parameter T – a larger T increases the probability of adding a good point to the regret set, but also leads to a slower algorithm. In the original paper, the authors suggested a value of T that is exponential in k; for instance, $T \geq 2.4 \times 10^7$ for $k = 10$, which is clearly not practical. In practice, Chester et al. [12] used $T = 54$ for $k = 4$, which is also the value we adopted in our experiments for comparison. Indeed, using $T > 54$ increases the running time but does not lead to significantly better regret sets.

The algorithms are implemented in C++ and we run on a 64-bit machine with four 3600 MHz cores and 16GB of RAM with Ubuntu 14.04. In evaluating the quality $\ell_k(\mathsf{Q})$ of a regret set $\mathsf{Q} \subseteq \mathsf{P}$, we compute the regret for a large set of random preferences (for example for $d = 3$ we take 20000 preferences), and use the maximum value found as our estimate. In fact, this approach gives us the distribution of the regret over the entire set of preference vectors.

[5] All data sets that we used and our implementation can be found on `https://users.cs.duke.edu/~ssintos/kRMS_SEA`

Datasets. We use the following datasets in our experiments, which include both synthetic and real-world.

BB (`databasebasketball.com`) is the basketball dataset that has been widely used for testing algorithms for skyline computation, top-k queries, and the k-RMS problem, [12, 21, 24, 25, 39]. Each point in this dataset represents a basketball player and its coordinates contain five statistics (points, rebounds, blocks, assists, fouls) of the player.

ElNino (`archive.ics.uci.edu/ml/datasets/El+Nino`) is the ElNino dataset containing oceanographic data such as wind speed, water temperature, surface temperature etc. measured by buoys stationed in the Pacific ocean, and also used in [12]. This dataset has some missing values, which we have filled in with the minimum value of a coordinate for the point. If some values are negative they are replaced by the absolute value.

Colors (`www.ics.uci.edu/ mlearn/MLRepository.html`) is a data set containing the mean, standard deviation, and skewness of each H, S, and V in the HSV color space of a color image. This set is also a popular one for evaluating skylines and regret sets (see [5, 31]).

AntiCor is a synthetic set of points with *anti-correlated* coordinates. Specifically, let h be the hyperplane with normal $\mathsf{n} = (1, \ldots, 1)$, and at distance 0.5 from the origin. To generate a point p, we choose a random point $\widetilde{p}$ on $\mathsf{h} \cap \mathbb{X}$, a random number $t \backsim \mathcal{N}(0, \sigma^2)$, for a small standard deviation σ, and $p = \widetilde{p} + t\mathsf{n}$. If $p \in \mathbb{X}$ we keep it, otherwise discard p. By design, many points lie on the skyline and the top-k elements can differ significantly even for nearby preferences. This data set is also widely used for testing top-k queries or skyline computation (see [6, 31, 39, 29]). For our experiments we set $\sigma = 0.1$ and generate 10000 points.

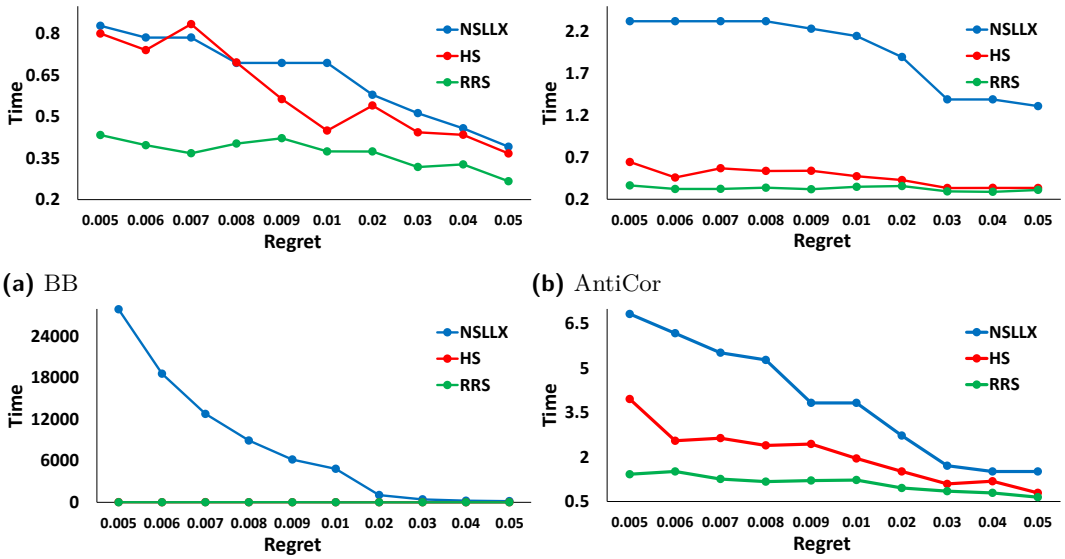
Sphere is a set of points uniformly distributed on the unit sphere inside $\mathbb{X}$, in which clearly all the points lie on the skyline. We generate the Sphere dataset with 15000 points for $d = 4$ (all points lie on the skyline).

SkyPoints is a modification of the Sphere data set. We choose a small fraction of points from the Sphere data set and for each point p add, say, 20 points that lie very close to p but are dominated by p. For larger value of k, say $k > 5$, considering only the skyline points is hard to decide which point is going to decrease the maximum regret ratio in the original set. We generate SkyPoints data set for $d = 3$, 500 points; with 100 points on the skyline.

In evaluating the performance of algorithms, we focus on two main criteria, the runtime and the regret ratio, but also consider a number of other factors that influence their performance such as the value of k, the size of the skyline etc.

RRS and HS are both randomized algorithms so we report the average size of the regret sets and the average running time computed over 5 runs. For $k = 1$, we use the NSLLX algorithm, and for $k = 10$, we use its extension, the CTVW algorithm. In some plots there are missing values for the CTVW algorithm, because we stopped the execution after running it on a data set for 2 days.

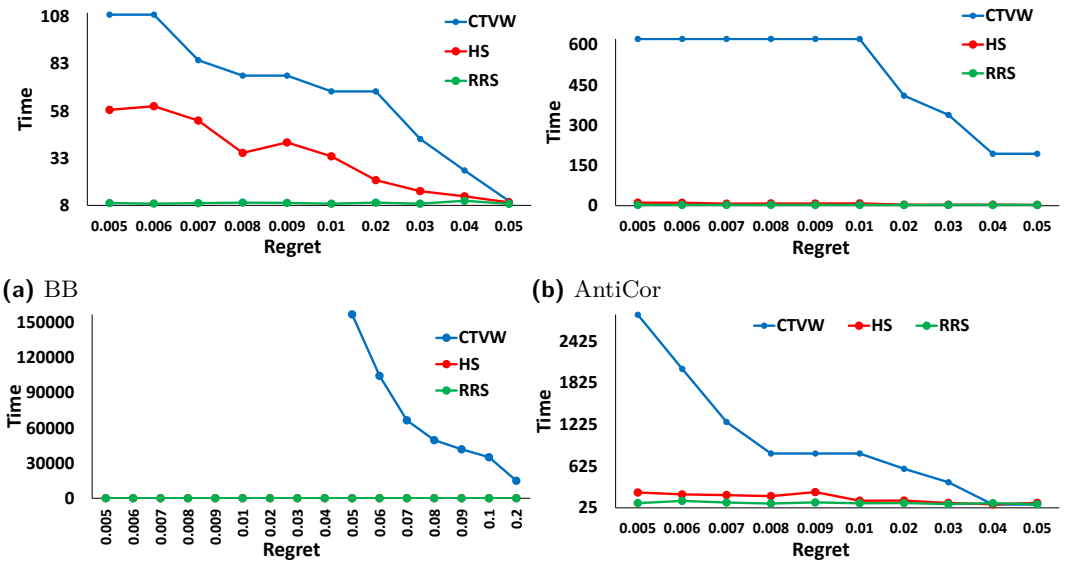
Running time. We begin with the runtime efficiency of the four algorithms, which is measured in the number of seconds taken by each to find a regret set, given a target regret ratio. Figure 3 shows the running times of NSLLX, HS, and RRS for $k = 1$. The algorithm RRS is the fastest. For some instances, the running time of HS and RRS are close but in some other instances HS is up to three times slower. The NSLLX algorithm is the slowest, especially for smaller values of the regret ratio. The relative advantage of our algorithms is quite significant for datasets that have large skylines, such as AntiCor and Sphere. Even for $k = 1$, NSLLX is 7 times slower than HS on AntiCor data set and 480 times slower on Sphere data set, for regret ratio ≤ 0.01.



(a) BB **(b)** AntiCor

(c) Sphere **(d)** ElNino

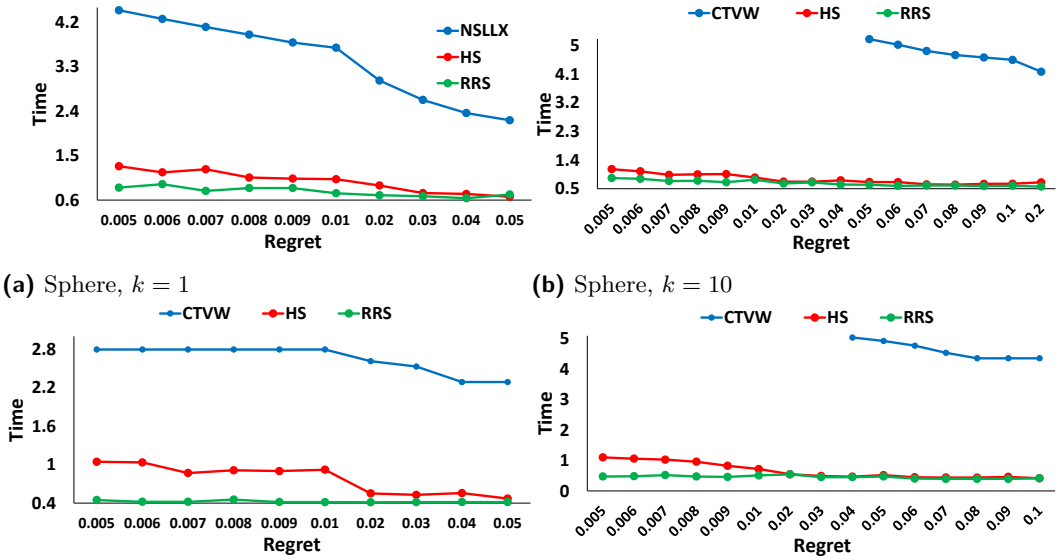
Figure 3 Running time for $k = 1$.



(a) BB **(b)** AntiCor

(c) Sphere **(d)** ElNino

Figure 4 Running time for $k = 10$.



(a) Sphere, $k = 1$

(b) Sphere, $k = 10$

(c) AntiCor, $\sigma = 0.1$, $k = 10$

(d) AntiCor, $\sigma = 0.01$, $k = 10$

Figure 5 $\log_{10}$-scale running time.

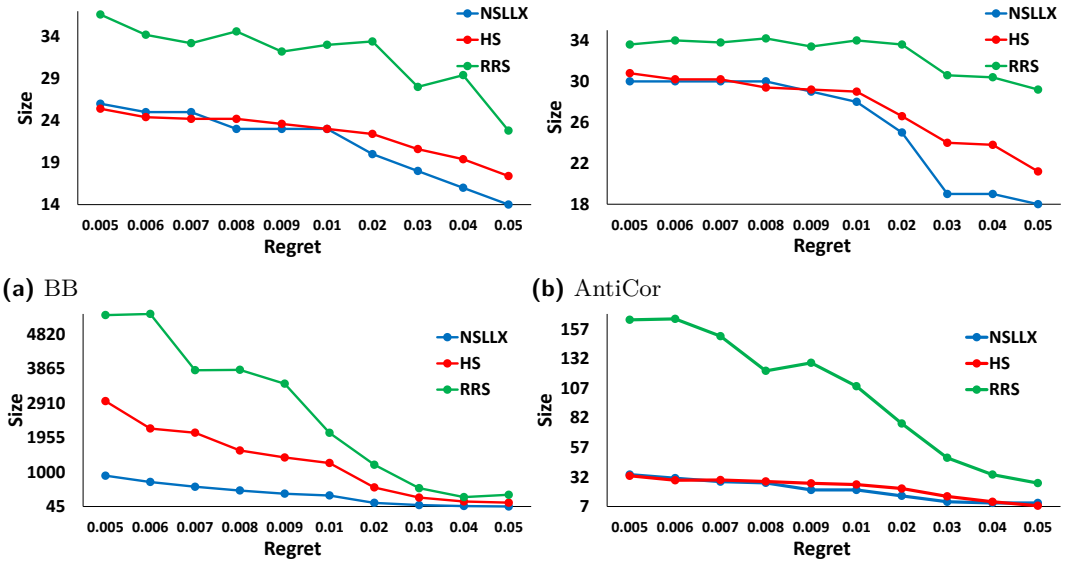
The speed advantage of RRS and HS algorithms over CTVW becomes much more pronounced for $k = 10$, as shown in Figure 4. Recall that CTVW discards all points that are not on the skyline. The running time is significantly larger if one runs this algorithm on the entire point set or when the skyline is large. For example, for the AntiCor and Sphere data sets, which have large size skylines, the CTVW algorithm is several orders of magnitude slower than ours. If we set the parameter $\sigma = 0.01$ for AntiCor data set, and generate 10000 points (the skyline has 8070 points in this case) the running time of CTVW is much higher as can be seen in Figure 9b. Because of the high running time of NSLLX and CTVW algorithms, in Figure 5, we show the running time in the log scale with base 10.

Regret ratio. We now compare the quality of the regret sets (size) computed by the four algorithms. Figures 6 and 7 show the results for $k = 1$ and for $k = 10$, respectively.

The experiments show that in general the HS algorithm finds regret sets comparable in size to NSLLX and CTVW. This is also the case for AntiCor data set if we set $\sigma = 0.01$ as can be seen in Figure 9a. The RRS algorithm tends to find the largest regret set among the four algorithms, but it does have the advantage of dynamic udpates: that is, RRS can maintain a regret set under insertion/deletion of points. However, since the other algorithms do not allow efficient updates, we do not include experiments on dynamic updates.

The sphere data set is the worst-case example for regret sets since every point has the highest score for some direction. As such, the size of the regret set is much larger than for the other data sets. HS and RRS algorithm rely on random sampling on preference vectors instead of choosing vectors adaptively to minimize the maximum regret, it is not surprising that for Sphere data sets CTVW does 1.5-3 times better than the HS algorithm. Nevertheless, as we will see below the regret of HS in 95% directions is close to that of CTVW.

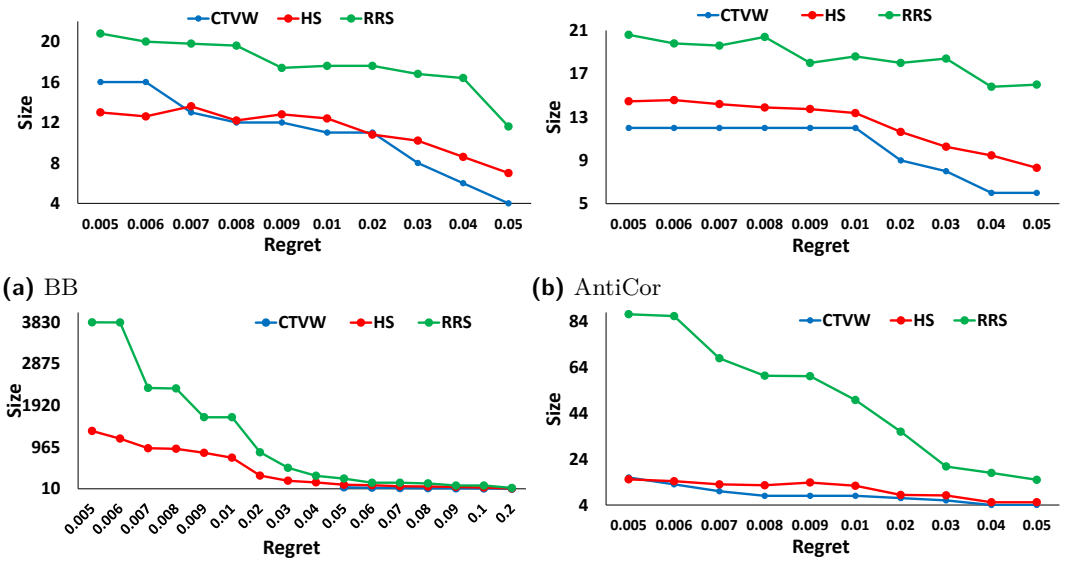
Regret distribution. The regret ratio only measures the *largest* relative regret over all preference vectors. A more informative measure could be to look at the entire distribution of the regret over all preference vectors.



(a) BB **(b)** AntiCor

(c) Sphere **(d)** ElNino

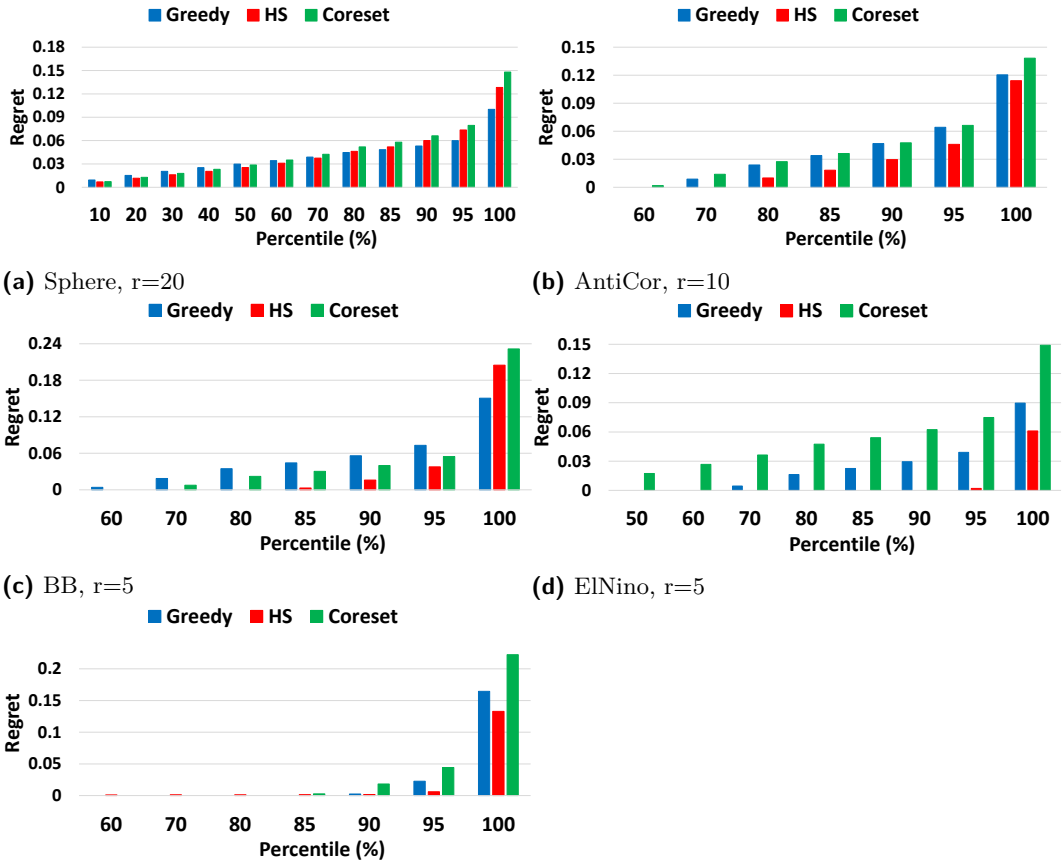
Figure 6 Maximum regret ratio for $k = 1$.



(a) BB **(b)** AntiCor

(c) Sphere **(d)** ElNino

Figure 7 Maximum regret ratio for $k = 10$.



(a) Sphere, r=20 **(b)** AntiCor, r=10

(c) BB, r=5 **(d)** ElNino, r=5

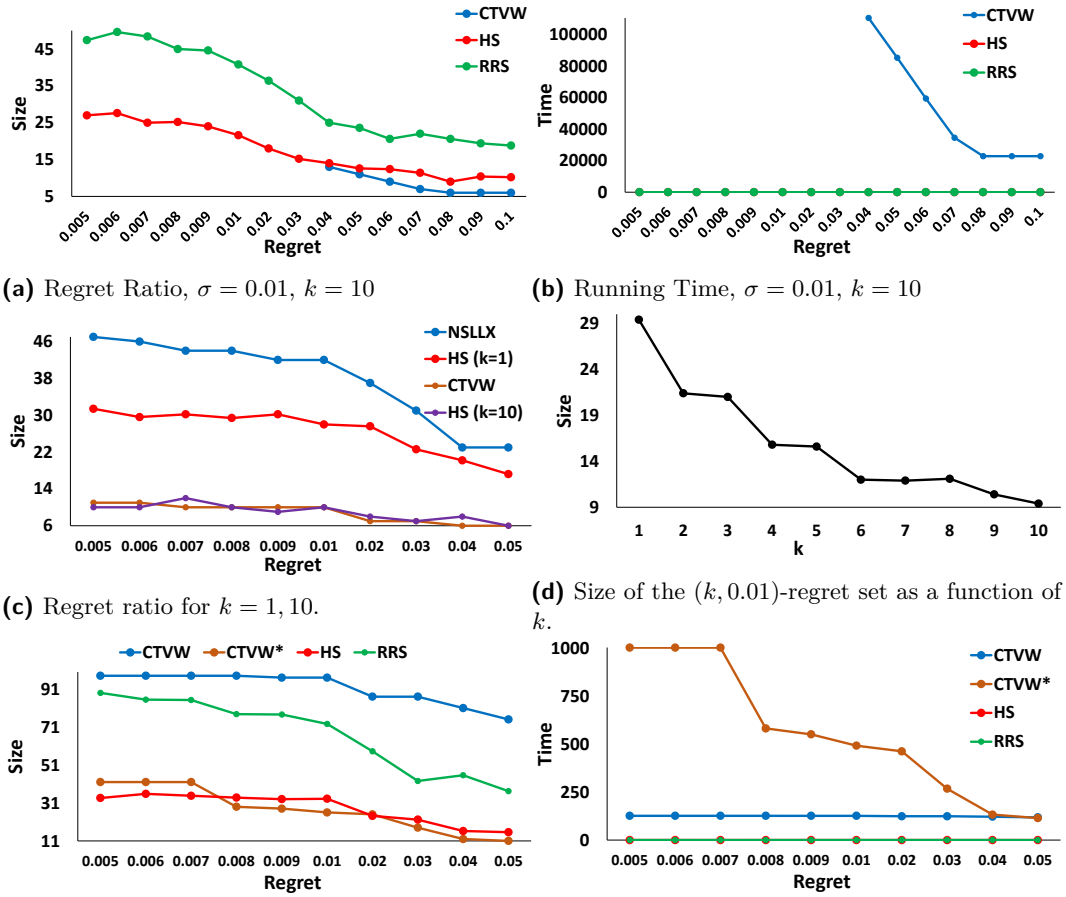
(e) Color, r=5

Figure 8 Regret distributions, $k = 1$.

We first show the results for the two synthetic main data sets, namely, Sphere and AntiCor. See Figures 8a, 8b. In this experiment, we fixed the regret set size to 20 for the Sphere dataset and 10 for the AntiCor dataset. We observe that the differences in the regret ratios in 95% of the directions are much smaller than the differences in the maximum regret ratios. For example, the difference of the maximum regret ratio between RRS and NSLLX in Sphere data set is 0.048, while the difference in the 95% (85%) of the directions is 0.019 (0.0096). As we can see in Figures 8c, 8d, 8e we get similar results for the real data sets. For the real datasets we fix the regret size to 5 because for higher values we found that 95% of the directions had 0-regret ratio for all algorithms.

Impact of larger k. We remarked in the introduction that the size of (k, ϵ)-regret set can be smaller for some datasets than their $(1, \epsilon)$-regret set, for $k > 1$. We ran experiments to confirm this phenomenon, and the results are shown in Figures 9c, 9d for Colors data set. As Figure 9c shows, the size of 1-regret set is 3.5 times larger than 10-regret sets for some values of the regret ratio. Figure 9d shows how the size of the regret set computed by the HS algorithm decreases with k, for a fixed value of the regret ratio 0.01.

Skyline effect. In order to improve its running time, the algorithm CTVW [12] removes all the non-skyline points, as a preprocessing step, before computing the regret set. While expedient, this strategy also risks finding directions with high k-regret ratio, and as a result



(a) Regret Ratio, $\sigma = 0.01$, $k = 10$

(b) Running Time, $\sigma = 0.01$, $k = 10$

(c) Regret ratio for $k = 1, 10$.

(d) Size of the $(k, 0.01)$-regret set as a function of k.

(e) Regret ratio, SkyPoints

(f) Running time, SkyPoints

Figure 9 Figures 9a, 9b: AntiCor. Figures 9c, 9d: Colors. Figures 9e, 9f: SkyPoints.

may lead to worse regret set. In this experiment, we used the Skypoint dataset to explore this cost/benefit tradeoff. In particular, the modified version of CTVW that does not remove non-skyline points is called CTVW*. The results are shown in Figure 9e, which confirm that removal of non-skyline points can cause significant increase in the size of the regret set, for a given target regret ratio. (In this experiment, the regret size differences are most pronounced for small values of regret ratio. When large values of regret ratio are acceptable, the loss of good candidate points is no longer critical.) Of course, while CTVW* finds nearly as good a regret set as HS, its running time is much worse than that of HS, or CTVW, because of this change, as shown in Figure 9f.

6 Related Work

The work on regret minimization was inspired by preference top-k and skyline queries. Both of these research topics try to help a user find the "best objects" from a database. Top-k queries assign scores to objects by some method, and return the objects with the topmost k scores while the skyline query finds the objects such that no other object can be strictly better. Efficiently answering top-k queries has seen a long line of work, see e.g. [14, 16, 18, 19, 26, 28, 35, 36, 40, 41, 43] and the survey [20]. In earlier work, the ranking

of points was done by weight, i.e., ranking criterion was fixed. Recent work has considered the specification of the ranking as part of the query. Typically, this is specified as a preference vector u and the ranking of the points is by linear projection on u see e.g. [14, 19, 41]. Another ranking criterion is based on the distance from a given query point in a metric space i.e., the top-k query is a k-nearest neighbor query [37].

In general, preference top-k queries are hard, and this has led to approximate query answering [11, 41, 42]. Motivated by the need of answering preference top-k queries, Nanongkai et. al. [31] introduced the notion of a 1-regret minimizing set (RMS) query. Their definition attempted to combine preference top-k queries and the concept of skylines. They gave upper and lower bounds on the regret ratio if the size of the returned set is fixed to r. Moreover, they proposed an algorithm to compute a 1-regret set of size r with regret ratio $O\left(\frac{d-1}{(r-d+1)^{1/(d-1)}+d-1}\right)$, as well as a greedy heuristic that works well in practice.

Chester et. al. [12] generalized the definition of 1-RMS to the k-RMS for any $k \geq 1$. They showed that the k-RMS problemis NP-hard when the dimension d is also an input to the problem, and they provided an exact polynomial algorithm for $d = 2$. There has been more work on the 1-RMS problem see [9, 30, 34], including a generalization by Faulkner et. al. [22] that considers non-linear utility functions.

The 1-regret problem can be easily addressed by the notion of ϵ-kernel coresets, first introduced by Agarwal et al. [1]. Later, faster algorithms were proposed to construct a coreset [10].

The 1-RMS problem is also closely related to the problem of approximating the Pareto curve (or skyline) of a set of points. Papadamitriou and Yannakakis [32, 33] considered this problem and defined an approximate Pareto curve as a set of points whose $(1 + \epsilon)$ scaling dominates every point on the skyline. They showed that there exists such a set of polynomial size [32, 33]. However, computing such a set of the smallest size is NP-Complete [23]. See also [38].

References

1 P. K. Agarwal, S. Har-Peled, and K. R. Varadarajan. Approximating extent measures of points. *Journal of the ACM (JACM)*, 51(4):606–635, 2004.

2 P. K. Agarwal, S. Har-Peled, and K. R. Varadarajan. Geometric approximation via coresets. *Combinatorial and computational geometry*, 52:1–30, 2005.

3 P. K. Agarwal and M. Sharir. Arrangements and their applications. *Handbook of computational geometry*, pages 49–119, 2000.

4 A. Asudeh, A. Nazi, N. Zhang, and G. Das. Efficient computation of regret-ratio minimizing set: A compact maxima representative. In *Proc. SIGMOD*, 2017. To appear.

5 I. Bartolini, P. Ciaccia, and M. Patella. Efficient sort-based skyline evaluation. *ACM Transactions on Database Systems (TODS)*, 33(4):31, 2008.

6 S. Börzsönyi, D. Kossmann, and K. Stocker. The skyline operator. In *Proc. 17th Int. Conf. Data Eng.*, pages 421–430, 2001.

7 H. Brönnimann and M. T. Goodrich. Almost optimal set covers in finite vc-dimension. *Discrete & Computational Geometry*, 14(4):463–479, 1995.

8 W. Cao, J. Li, H. Wang, K. Wang, R. Wang, R. Chi-Wing Wong, and W. Zhan. k-regret minimizing set: Efficient algorithms and hardness. In *ICDT 2017-20th International Conference on Database Theory*, pages 11:1–11:19, 2017.

9 I. Catallo, E. Ciceri, P. Fraternali, D. Martinenghi, and M. Tagliasacchi. Top-k diversity queries over bounded regions. *ACM Transactions on Database Systems (TODS)*, 38(2):10, 2013.

10 T. M. Chan. Faster core-set constructions and data stream algorithms in fixed dimensions. In *Proceedings of the twentieth annual symposium on Computational geometry*, pages 152–159. ACM, 2004.

11 D. Chen, G.-Z. Sun, and N. Z. Gong. Efficient approximate top-k query algorithm using cube index. In *Asia-Pacific Web Conference*, pages 155–167. Springer, 2011.

12 S. Chester, A. Thomo, S. Venkatesh, and S. Whitesides. Computing k-regret minimizing sets. *Proceedings of the VLDB Endowment*, 7(5):389–400, 2014.

13 G. Das and M. T. Goodrich. On the complexity of optimization problems for 3-dimensional convex polyhedra and decision trees. *Computational Geometry*, 8(3):123–137, 1997.

14 G. Das, D. Gunopulos, N. Koudas, and N. Sarkas. Ad-hoc top-k query answering for data streams. In *Proceedings of the 33rd International Conference on Very Large Data Bases*, VLDB'07, pages 183–194, 2007.

15 Z. Gong, G.-Z. Sun, J. Yuan, and Y. Zhong. Efficient top-k query algorithms using k-skyband partition. In *International Conference on Scalable Information Systems*, pages 288–305. Springer, 2009.

16 U. Güntzer, W. Balke, and W. Kiessling. Optimizing multi-feature queries for image databases. In *Proceedings of the 26th International Conference on Very Large Data Bases*, VLDB'00, pages 419–428, 2000.

17 Inc. Gurobi Optimization. Gurobi optimizer reference manual, 2015. URL: `http://www.gurobi.com`.

18 J.-S. Heo, J. Cho, and K.-Y. Whang. The hybrid-layer index: A synergic approach to answering top-k queries in arbitrary subspaces. In *2010 IEEE 26th International Conference on Data Engineering (ICDE 2010)*, pages 445–448. IEEE, 2010.

19 V. Hristidis, N. Koudas, and Y. Papakonstantinou. Prefer: A system for the efficient execution of multi-parametric ranked queries. *SIGMOD Rec.*, 30(2):259–270, May 2001.

20 I. F. Ilyas, G. Beskales, and M. A. Soliman. A survey of top-k query processing techniques in relational database systems. *ACM Computing Surveys (CSUR)*, 40(4):11, 2008.

21 S. Jasna and M. J. Pillai. An algorithm for retrieving skyline points based on user specified constraints using the skyline ordering. *International Journal of Computer Applications*, 104(11), 2014.

22 T. Kessler Faulkner, W. Brackenbury, and A. Lall. k-regret queries with nonlinear utilities. *Proceedings of the VLDB Endowment*, 8(13):2098–2109, 2015.

23 V. Koltun and C. H. Papadimitriou. Approximately dominating representatives. *Theor. Comput. Sci.*, 371(3):148–154, February 2007.

24 R. D. Kulkarni and B. F. Momin. Skyline computation for frequent queries in update intensive environment. *Journal of King Saud University-Computer and Information Sciences*, 2015.

25 R. D. Kulkarni and B. F. Momin. Parallel skyline computation for frequent queries in distributed environment. In *Computational Techniques in Information and Communication Technologies (ICCTICT), 2016 International Conference on*, pages 374–380. IEEE, 2016.

26 C. Li, Kevin K. C.-C. Chang, and I. F. Ilyas. Supporting ad-hoc ranking aggregates. In *Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data*, SIGMOD'06, pages 61–72, 2006.

27 Q. Liu, Y. Gao, G. Chen, Q. Li, and T. Jiang. On efficient reverse k-skyband query processing. In *International Conference on Database Systems for Advanced Applications*, pages 544–559. Springer, 2012.

28 A. Marian, N. Bruno, and L. Gravano. Evaluating top-k queries over web-accessible databases. *ACM Trans. Database Syst.*, 29(2):319–362, June 2004.

29 M. Morse, J. M. Patel, and W. I. Grosky. Efficient continuous skyline computation. *Information Sciences*, 177(17):3411–3437, 2007.

30 D. Nanongkai, A. Lall, A. Das Sarma, and K. Makino. Interactive regret minimization. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, pages 109–120. ACM, 2012.

31 D. Nanongkai, A. D. Sarma, A. Lall, R. J. Lipton, and J. Xu. Regret-minimizing representative databases. *Proceedings of the VLDB Endowment*, 3(1-2):1114–1124, 2010.

32 C. H. Papadimitriou and M. Yannakakis. On the approximability of trade-offs and optimal access of web sources. In *Proceedings of the 41st Annual Symposium on Foundations of Computer Science*, FOCS'00, pages 86–92, 2000.

33 C. H. Papadimitriou and M. Yannakakis. Multiobjective query optimization. In *Proceedings of the Twentieth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS'01, pages 52–59, 2001.

34 P. Peng and R. C.-W. Wong. Geometry approach for k-regret query. In *2014 IEEE 30th International Conference on Data Engineering*, pages 772–783. IEEE, 2014.

35 S. Rahul and Y. Tao. Efficient top-k indexing via general reductions. In *Proceedings of the 35th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, PODS'16, pages 277–288, 2016.

36 M. Theobald, G. Weikum, and R. Schenkel. Top-k query evaluation with probabilistic guarantees. In *Proceedings of the 300th International Conference on Very Large Data Bases*, VLDB'04, pages 648–659, 2004.

37 E. Tiakas, G. Valkanas, A. N. Papadopoulos, Y. Manolopoulos, and D. Gunopulos. Processing top-k dominating queries in metric spaces. *ACM Trans. Database Syst.*, 40(4):23:1–23:38, January 2016.

38 S. Vassilvitskii and M. Yannakakis. Efficiently computing succinct trade-off curves. *Theor. Comput. Sci.*, 348(2):334–356, December 2005.

39 A. Vlachou, C. Doulkeridis, Y. Kotidis, and K. Nørvåg. Reverse top-k queries. In *2010 IEEE 26th International Conference on Data Engineering (ICDE 2010)*, pages 365–376. IEEE, 2010.

40 D. Xin, C. Chen, and J. Han. Towards robust indexing for ranked queries. In *Proceedings of the 32Nd International Conference on Very Large Data Bases*, VLDB'06, pages 235–246, 2006.

41 A. Yu, P. K. Agarwal, and J. Yang. Processing a large number of continuous preference top-k queries. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, pages 397–408. ACM, 2012.

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

43 Z. Zhang, S. w. Hwang, K. C.-C. Chang, M. Wang, C. A. Lang, and Y. c. Chang. Boolean + ranking: Querying a database by k-constrained optimization. In *Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data*, SIGMOD'06, pages 359–370, 2006.

A Affine transformation of polytope Π

From Lemma 3 we know that RMS problem is NP-Hard when the preferences vectors are in $\mathbb{R}^3$. Here, we show that RMS problem is NP-Hard even when preferences are restricted to $\mathbb{X}$. The reduction is similar to the reduction proposed in Lemma 3. The only difference is that polytope Π needs to have two additional properties:

(i) All vertices of Π must lie in the first orthant.

(ii) For any edge (v_1, v_2) of Π, where v_1, v_2 are vertices of P, there is a direction $u \in \mathbb{X}$ such that v_1, v_2 are the top vertices in direction u.

It is easy to satisfy property (i) because the translation of the vertices of a polytope does not change the rank of the points in any direction. On the other hand, property (ii) is not guaranteed by the construction in [13].

We show that there is an affine transformation of Π that can be computed and applied in polynomial time, to get a polytope Π′ with the same combinatorial structure as Π, but that also satisfies properties (i), and (ii). The fact that the polytope has the same combinatorial structure implies that the underlying graph is the same, and therefore a vertex cover will also be a $(2, 0)$-regret set of Π. Next, we describe the details of the transformation.

Construction. First, we translate Π such that the origin o is inside Π. Then, we compute the polar dual Π^*[6]. Let v be a vertex of Π^*. Translate Π^* such that v becomes the origin. Then take a rotation such that polytope Π^* does not intersect the negative orthant – i.e., the set of points in $\mathbb{R}^3$ which have all coordinates strictly negative; we can always do it because Π^* is convex. Let u_1, u_2, u_3 be the three directions emanating from the origin such that the cone defined by them, contains the entire polytope Π^*. Such directions always exist and can be found in polynomial time. It is known that we can find in polynomial time an affine transformation such that u_1 is mapped to the direction $e_1 = (1, 0.01, 0.01)$, u_2 to direction $e_2 = (0.01, 1, 0.01)$ and u_3 to $e_3 = (0.01, 0.01, 1)$ (we can do it by first transforming u_1, u_2, u_3 to the unit axis vectors and then transform them to e_1, e_2, e_3). Apply this affine transformation to Π^* to get $\hat{\Pi}^*$. Polytope $\hat{\Pi}^*$ lies in the first orthant, except of vertex v which is at the origin. Shift this polytope slightly such that the origin lies in the interior of the polytope, v lies in the negative orthant, and all the other vertices are still in the first orthant. Such a translation can be computed in polynomial time by subtracting from all coordinates a quantity $m/2$, where m is the value of the minimum coordinate over all points except v. Hence, after the translation, $v = (-m/2, -m/2, -m/2)$. Finally we compute the polar dual of $\hat{\Pi}^*$; call this $\hat{\Pi}$. Translate $\hat{\Pi}$ until all vertices have positive coordinates, and let Π′ denote the new polytope.

▶ **Lemma 11.** *Polytope* Π′ *is combinatorially equivalent to* Π *and satisfies properties* (i), (ii).

Proof. We start by mapping property (ii) in the dual space. Consider a polytope G and its dual G^* (where the origin lies inside them). It is well known that any vertex v of G corresponds to a hyperplane h_v in the dual space that defines a facet of G^*. An edge between two vertices in G corresponds to an edge between the two corresponding faces in G^*. Furthermore, if a vertex v of G is the top-k vertex of G in a direction u, then the

[6] The polar dual of a polytope containing the origin o is defined as the intersection of all hyperplanes $\langle x, p\rangle \leq 1$ where $p \in P$, and it can be equivalently defined as the intersection of the dual hyperplanes $\langle x, v\rangle \leq 1$ for all the vertices v of P.

corresponding hyperplane h_v is the k-th hyperplane (among the n dual hyperplanes) that is intersected by the ray ou, where o is the origin. From the above it is straightforward to map property (ii) in the dual space: (ii') For any edge (f_1, f_2) where f_1, f_2 are faces of G^* there is a direction $u \in \mathbb{X}$ such that the first two hyperplanes that are intersected by the ray ou are h_1, h_2, where h_1 is the hyperplane that contains the face f_1 and h_2 the hyperplane that contains the face f_2.

We now show how these properties can be guaranteed in $\hat{\Pi}^*$. Notice that from the construction of $\hat{\Pi}^*$, the origin lies inside $\hat{\Pi}^*$ and all faces of $\hat{\Pi}^*$ have non empty intersection with the positive octant. By convexity, $\hat{\Pi}^*$ satisfies property (ii') because for any edge $e = (f_1, f_2)$ of $\hat{\Pi}^*$ there is a ray emanating from the origin that first intersects the edge e, and hence the hyperplanes h_1, h_2 are the first hyperplanes that are intersected by the ray. So, its dual polytope $\hat{\Pi}$ satisfies property (ii). In addition, Π^* is combinatorially equivalent to Π, by duality. Since we apply an affine transformation $\hat{\Pi}^*$ is also combinatorially equivalent to Π^*. Finally, the polytope $\hat{\Pi}$ is combinatorially equivalent to $\hat{\Pi}^*$ (its dual). Notice that translation does not change the combinatorial structure of a polytope or the ordering of the points in any direction, so Π' satisfies property (ii), property (i) by definition, and is also combinatorially equivalent to Π. ◀

The first part of the NP-Hardness proof is the same with the case of all directions in $\mathbb{R}^3$ in Lemma 3, if Q is a vertex cover of Π' then it is also a $(2, 0)$-regret set. Using property (ii) of Π', it is straightforward to show the other direction.

B Proof of Lemma 8

▶ **Lemma 8.** *Let Q' be a hitting set of Σ_{N}, and let B be the basis of P. Then $\mathsf{Q} = \mathsf{Q}' \cup B$ is a $(k, \epsilon + \delta - \delta\epsilon)$-regret set of P.*

Proof. It suffices to show that for any direction $u \in \mathbb{U}$ there is a point $q \in \mathsf{Q}$ for which $\omega(u, q) \geq (1 - \delta)(1 - \epsilon)\omega_k(u, \mathsf{P})$ (because $1 - (\epsilon + \delta - \delta\epsilon) = (1 - \delta)(1 - \epsilon)$).

We first consider the case when $\omega_k(u, \mathsf{P}) \leq \frac{1}{(1-\epsilon)\sqrt{d}}$. In this case, by the proof of Lemma 6 the set B is guaranteed to contain a point q with $\omega(u, q) \geq \frac{1}{\sqrt{d}}$, which proves the claim. So let now assume that $\omega_k(u, \mathsf{P}) > \frac{1}{(1-\epsilon)\sqrt{d}}$. Let $\bar{u} \in \mathsf{N}$ be a direction in the net N such that, $\widehat{(u, \bar{u})} \leq \delta/2d$, where $\widehat{(u, \bar{u})}$ is the angle between u and $\bar{u}$. Such a direction exists because N is a $\frac{\delta}{2d}$-net on $\mathbb{U}$. Observe that,

$$\|u - \bar{u}\| = \sqrt{2 - 2\cos(\widehat{(u, \bar{u})})} = 2\sin\left(\frac{\widehat{(u, \bar{u})}}{2}\right) \leq \frac{\delta}{2d},$$

where we have used first the cosine rule, the identity $1 - \cos\theta = 2\sin^2\left(\frac{\theta}{2}\right)$, as well as the inequality $\sin\theta \leq \theta$ for $\theta \geq 0$ in the final step. Also, observe that for any $p \in \mathsf{P}$ we have,

$$|\omega(u, p) - \omega(\bar{u}, p)| \leq \frac{\delta}{2\sqrt{d}}. \tag{1}$$

This follows because,

$$|\omega(u, p) - \omega(\bar{u}, p)| = |\langle u, p\rangle - \langle \bar{u}, p\rangle| = |\langle u - \bar{u}, p\rangle| \leq \|u - \bar{u}\| \cdot \|p\| \leq \frac{\delta}{2d} \cdot \sqrt{d} = \frac{\delta}{2\sqrt{d}},$$

where we have used the Cauchy-Schwarz inequality for the first inequality, the upper bound on $\|u - \bar{u}\|$ derived earlier, along with $\|p\| \leq \sqrt{d}$ (by Lemma 6) for the second inequality.

Let $x_1, x_2, \ldots, x_k \in \mathsf{P}$ be the top-k points along direction u, i.e., $x_i = \varphi_i(u, \mathsf{P})$. Also, let y_k be the top-k point along direction $\bar{u}$. As remarked we can assume, $\omega(u, x_i) \geq \omega(u, x_k) \geq \frac{1}{(1-\epsilon)\sqrt{d}}$. Now, for any $i = 1, 2, \ldots, k$ we have that,

$$\begin{aligned}\omega(\bar{u}, x_i) &\geq \omega(u, x_i) - \frac{\delta}{2\sqrt{d}} \geq \omega(u, x_i) - \frac{(1-\epsilon)\delta}{2}\omega(u, x_i) \\ &= \omega(u, x_i)\left(1 - \frac{(1-\epsilon)\delta}{2}\right) \geq \omega(u, x_k)\left(1 - \frac{(1-\epsilon)\delta}{2}\right).\end{aligned}$$

The first inequality follows by *Equation* 1, and the second inequality holds since $\omega(u, x_i) \geq \omega(u, x_k) > \frac{1}{(1-\epsilon)\sqrt{d}}$. This implies that there are k points whose scores are each at least $\omega(u, x_k)\left(1 - \frac{(1-\epsilon)\delta}{2}\right)$, and therefore the k-th best score along $\bar{u}$, i.e., $\omega(\bar{u}, y_k)$, is at least $\omega(u, x_k)\left(1 - \frac{(1-\epsilon)\delta}{2}\right)$. Now, the algorithm guarantees that there is a point $q \in Q$ such that $\omega(\bar{u}, q) \geq (1-\epsilon)\omega(\bar{u}, y_k)$. We claim that this q "settles" direction u as well, up-to the factor $(1-\delta)(1-\epsilon)$. Indeed,

$$\begin{aligned}\omega(u, q) &\geq \omega(\bar{u}, q) - \frac{\delta}{2\sqrt{d}} \geq (1-\epsilon)\omega(\bar{u}, y_k) - \frac{\delta}{2\sqrt{d}} \\ &\geq (1-\epsilon)\left(1 - \frac{(1-\epsilon)\delta}{2}\right)\omega(u, x_k) - \frac{\delta}{2\sqrt{d}} \\ &\geq (1-\epsilon)\left(1 - \frac{(1-\epsilon)\delta}{2}\right)\omega(u, x_k) - \frac{(1-\epsilon)\delta}{2}\omega(u, x_k) \\ &= (1-\epsilon)(1 - \delta + \delta\epsilon/2)\omega(u, x_k) \geq (1-\delta)(1-\epsilon)\omega(u, x_k)\end{aligned}$$

This completes the proof. ◀