k-Regret Queries with Nonlinear Utilities

Taylor Kessler Faulkner
Denison University
Granville, OH, USA

Will Brackenbury*
Denison University
Granville, OH, USA

Ashwin Lall
Denison University
Granville, OH, USA

ABSTRACT

In exploring representative databases, a primary issue has been finding accurate models of user preferences. Given this, our work generalizes the method of regret minimization as proposed by Nanongkai et al. to include nonlinear utility functions. Regret minimization is an approach for selecting k representative points from a database such that every user's ideal point in the entire database is similar to one of the k points. This approach combines benefits of the methods top-k and skyline; it controls the size of the output but does not require knowledge of users' preferences. Prior work with k-regret queries assumes users' preferences to be modeled by linear utility functions. In this paper, we derive upper and lower bounds for nonlinear utility functions, as these functions can better fit occurrences such as diminishing marginal returns, propensity for risk, and substitutability of preferences. To model these phenomena, we analyze a broad subset of convex, concave, and constant elasticity of substitution functions. We also run simulations on real and synthetic data to prove the efficacy of our bounds in practice.

1. INTRODUCTION

When users make a request to a database, they are often unwilling to search through all of the results to find an item that best fits their preferences. Given this, multiple operators have been proposed to reduce output size for database queries while still effectively representing the entire database.

Top-k [13] and skyline [2, 10] operators have both been studied as possible solutions to the problem of reducing output to a manageable size. However, both of these operators have drawbacks. Top-k asks for users to input their utility functions and uses these functions to return a customized set of items to every user. However, this method requires users to both know and input their utility functions. Users may not know exactly how much they care about any given attribute of an item, or they may not want to take the time and effort to enter their utility function. The skyline operator eliminates any point if it is less preferable in every attribute than another point. However, this method does not control the exact number of items returned, leaving the possibility of it not sufficiently reducing the output size. To combat these issues, Nanongkai et al. [23] introduced the concept of regret, which we build upon in this paper.

*co-first author

Proceedings of the VLDB Endowment, Vol. 8, No. 13
Copyright 2015 VLDB Endowment 2150-8097/15/09.

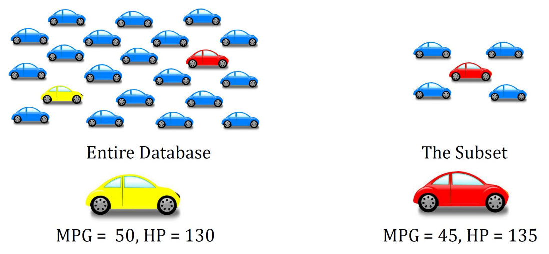


Figure 1: Example of Regret

Regret is informally defined in the following example. Assume a user, Alice, queries for a small set from a large database of cars. These cars have two attributes, horsepower (HP) and mileage per gallon (MPG). We will also assume that Alice has a utility function that can be determined by her attribute preferences. However, as we do not know what her utility function is, it is impossible to guarantee that we will output her ideal car in this small set. We can, however, provide her with a result that is close to her ideal car. The difference between the utility of her ideal car and the utility provided by the best car for her in the result set is her regret. We can find Alice's regret ratio by dividing regret by the utility of her ideal car. Finding a small set that bounds the regret ratio of any possible user is the goal of the k-regret query.

For example, assume that Alice's utility function is (MPG)^2 + 2(HP). Her ideal car from the entire database is the yellow car (see Figure 1), which gives her a utility of 2760. Her ideal car from the subset is the red car, which gives her a utility of 2295. Thus her regret ratio is $\frac{2760-2295}{2760} = .17$, or 17%. The goal of our algorithms is to minimize the regret ratio for Alice and all other possible users.

Previous work with the k-regret query has shown that the regret ratio can be bounded to a small quantity for linear utility functions, but the question was left open as to whether this was the case for broader classes of utility functions. Our work answers this affirmatively, showing theoretical bounds for a large subset of convex, concave, and constant elasticity of substitution (CES) utility functions. The addition of these classes of functions to our set of possible utility functions greatly enhances the power of the k-regret query, because it more accurately explains common phenomena in utility functions.

One such instance is diminishing marginal returns, the effect by which a user grows less satisfied with each additional unit of a good. It is most often represented by a concave function. Since this is a known effect, taking it into account when designing the database operator is critical so as to ensure the user's regret is not higher than anticipated by models that only include linear utility functions. For example, imagine that you, like Alice, would like to purchase a car. You want a car with high horsepower. If you were given a choice between car A, which has 260 HP, and car B, which has 280 HP, you would have a strong preference for car B. However, if you were given a choice between car C, which has 460 HP, and car D, which has 480 HP, car D would not seem much better to you than car C, as car C already has very high horsepower. Your desire for high horsepower can be represented by a concave function that puts less weight on differences in horsepower as horsepower gets higher. In addition, convex utility functions often represent the preferences of risk-loving individuals who will invest in a commodity despite a lower expected payout due to the risk.

CES utility functions prove useful by modeling a user's proclivity or aversion to variety amongst the attributes. For example, CES functions representing a low elasticity of substitution indicate a user who desires a diverse mix of criteria, since he or she believes the attributes to be complementary toward each other, while a function with a high elasticity of substitution shows a user who believes the varying attributes are, to a greater degree, interchangeable. Since all these effects occur frequently in behavioral analysis, we believe including them in our utility models should be a priority.

Ergo, our main contributions are as follows:

- We present the MinWidth algorithm which allows us to bound the maximum regret ratio for most independent, monotonically increasing convex, concave, and CES utility functions. These bounds are dependent solely on the number of points displayed and are independent of the number of tuples in the database, i.e., we can summarize databases of unlimited size with only a few points.
- We introduce two additional heuristic algorithms that give low regret ratios in practice.
- We give lower bounds for the convex, concave, and CES cases that are close to matching the abovementioned upper bounds.
- We show that regret can be very low in practice via simulations on both real and synthetic data.

After exploring related work in Section 2, we will formally define our problem in Section 3. We then provide an algorithm with worst-case upper bounds in Section 4, followed by two heuristic algorithms in Section 5. Next, we present our lower bounds in Section 6. In Section 7 we provide our experimental results and we conclude in Section 8.

2. RELATED WORK

Many others have proposed approaches to representative databases. There have been variations of top-k and skyline that attempt to fix the various issues that these techniques have. Several regret-minimization techniques have also been proposed.

Börzsönyi et al. [2] first introduced the concept of skyline. Barndorff-Nielsen and Sobel, Kung et al., Matousek, and Magnani et al. [1, 15, 20, 21] have all expanded the work on skyline. Das Sarma et al. [8] propose another approach to skyline. They assume users based on a probability distribution of utility functions instead of a fixed set of users. These users are modeled on threshold-based preferences which divide tuples into those which a user would or would not select. Lin et al. [18] and Tao et al. [26] propose an approach called k-representative skyline. Both find k skyline tuples that represent the skyline. Lin et al. [18] propose a method that finds the k skyline tuples that dominate the most tuples. This method is scale invariant; that is, it still outputs the same tuples even if we scale the input attribute values. Tao et al. [26] demonstrate that the approach that Lin et al. provide might yield a objectionable solution. They propose an alternate solution based on distances that solves the k-center problem on skyline points. This method is not scale-invariant, since it uses distances. There has also been work on constrained subspace skyline computation such as that by Dellis et al. [9] that propose range constraints specified by skyline queries. Magnani et al. [19] work with aggregate data rather than just single records. The method proposed returns desired groups of records based on their features.

Papadopoulos et al. and Yiu and Mamoulis [24,29] propose top-k dominating query. This method ranks each tuple by the number of other tuples that it dominates. Top-k dominating query combines benefits of both top-k and skyline, as it controls output size without asking users for utility functions. This query has been further explored by Lian and Chen [17] and Zhang et al. [30] using uncertain databases, and by Kontaki et al. [14] using continuous processing. Ilyas et al., Hristidis et al, Chang et al, and Tsaparas et al. [6,12,13,27] all use top-k operators; however, they ask for utility functions, which is a burden on the user. Goncalves and Vidal [11] propose the operators top-k skyline select and top-k skyline join. They ask users for criteria and utility functions. The operator then uses this information to return skyline tuples according to the given criteria and top k skyline tuples according to the utility functions. This method controls the size of the skyline, but still requires users to know and input their utility functions.

There have been several approaches to regret minimization. Nanongkai et al. [23] introduced the concept of regret minimization. This method controls output size while not requiring users to input their utility functions. However, the approach proposed by Nanongkai et al. only guarantees bounds for linear utility functions. Nanongkai et al. [22] incorporate user interaction into regret minimization by repeatedly returning selected tuples to users and asking them to select their favorite. They use this information to better understand which set of tuples each user desires. This process continues until the user decides to stop giving input. This method increases the happiness of each user but burdens users with feedback. Chester et al. [7] propose a method called relaxation to apply should there not be an appropriate subset of points to make each user acceptably happy. Peng et al. [25] use regret minimization, but first narrow down the database to a set of points called *happy points* using a geometric approach. Catallo et al. [3] also narrow down the points, but do so by using an algorithm to explore the points. This algorithm records the relevance and position of points in order to avoid needing to look through every relevant point in the database. None of the previously mentioned approaches incorporate nonlinear utility functions.

There have been other approaches to limit output size besides the ones mentioned above. Chan et al. [5] propose skyline frequency. This ranks points by counting the number of dimensions for which each point is in the skyline. Xia et al. [28] propose ϵ-skyline queries. This asks users to give weights to certain attributes of a point, and controls output size using ϵ. Lee et al. [16] ask users for partial rankings, which allows them to not ask for entire utility functions. Chan et al. [4] introduce a new definition of dominance; a point is dominant if it is better in no less than k dimensions. These methods all control output size; however, they do not minimize regret. None of the methods mentioned in this section both minimize regret and use nonlinear utility functions.

3. PROBLEM DEFINITION

We are given a database (D) of n d-dimensional points, represented as tuples, and an output size k, where $d \leq k \leq n$. For every point $p \in D$, the value in any dimension i, represented as p_i, has the domain $\mathbb{R}_+$; each value represents the tuple's preferability in that dimension, where greater values are more preferable. The degree to which a point is favored by any given user is represented by the user's utility function.

DEFINITION 1. **Utility Function.** *A utility function f is a map $f : \mathbb{R}^d_+ \rightarrow \mathbb{R}_+$. We say that a user with utility function f has utility $f(p)$ for any point p.*

Since we assume that bigger values are better, all utility functions in this paper will be monotonically non-decreasing in each of their arguments.

DEFINITION 2. **Independent Utility Function.** *A utility function f is said to be independent if it can be expressed in the form $f(p) = \sum_{i=1}^{d} f_i(p_i)$, for some functions $f_i : \mathbb{R}_+ \rightarrow \mathbb{R}_+$, $1 \leq i \leq d$.*

EXAMPLE 1. *The end user Alice is shown a 2-dimensional point represented by the tuple $s = (.2, .8)$. Alice's preferences conform to the utility function $f(x_1, x_2) = 2x_1^2 + 3x_2^2$, which means that the given tuple yields a utility of 2. Note that Alice's utility function is independent. An example of a non-independent function is $g(x_1, x_2) = 2x_1^2 x_2^2 + x_2^3$.*

The reason for our focus on independent utility functions is because a user's preference for each attribute is usually unaffected by their preference for other attributes. In cases where this is not so, we use CES functions to describe the effect.

Without knowledge of Alice's utility function, though, one cannot accurately calculate the utility of a point as one does in a top-k query. As a solution, k-regret queries [23] seek to return points that minimize the regret ratio for the user.

DEFINITION 3. **Regret Ratio.** *If f represents a user's utility function, p is her optimal point in the database D, and s is her optimal point amongst the output S of k points shown to her, then we define her regret ratio to be $rr_D(S, f) = \frac{f(p)-f(s)}{f(p)}$.*

EXAMPLE 2. *Given all circumstances of our previous example, now assume Alice's optimal point p to be a point in the database with weights $(.6, .7)$. Alice is not shown this point, and therefore picks point s. Since p yields a utility of 2.19, and s gave a utility of 2, Alice's regret ratio is $\frac{.19}{2.19}$, or 8.68%.*

Since utility functions vary across users, however, one must not only minimize the regret ratio for Alice's utility function, but also for all other users. Thus, any algorithm for a k-regret query must minimize the maximum regret ratio across the class of utility functions, $\mathcal{F}$.

DEFINITION 4. **Maximum Regret Ratio.** *The maximum regret ratio for showing the set S instead of the entire database D to a set of users with utility functions from the class $\mathcal{F}$ is defined as $mrr_D(S, \mathcal{F}) = \sup_{f \in \mathcal{F}} rr_D(S, f)$. Note that we must take the supremum as $\mathcal{F}$ is an infinite set and the maximum may not exist.*

In all prior work, the class of functions $\mathcal{F}$ was restricted to ones with linear utility [7,22,23,25]. In this paper, we consider functions from the following very broad classes.

DEFINITION 5. **Convex function.** *A function f is said to be convex over $\mathbb{R}_+$ if for all $x_1, x_2 \in \mathbb{R}_+$ and $\lambda \in [0, 1]$,*

$$f(\lambda x_1 + (1 - \lambda)x_2) \leq \lambda f(x_1) + (1 - \lambda)f(x_2).$$

If f is twice differentiable, then it is convex if and only if $f''(x) \geq 0$ for all x.

DEFINITION 6. **Concave function.** *A function f is said to be concave if $-f$ is convex. If f is twice differentiable, then it is concave if and only if $f''(x) \leq 0$ for all x.*

DEFINITION 7. **Constant Elasticity of Substitution (CES) Functions.** *Constant elasticity of substitution functions are utility functions of the form $f(x) = (\sum_{i=1}^{d} a_i x_i^b)^{\frac{1}{b}}$, where $b \in \mathbb{R}$ and $a_i \in \mathbb{R}_+$ for all $1 \leq i \leq d$.*

Note that CES functions are not independent.

3.1 Problem Definition

When a user with an unknown utility function from the class $\mathcal{F}$ queries for k points from a database D with n d-dimensional points, return a set S of size k that minimizes $mrr_D(S, \mathcal{F})$. The classes of utility functions we consider are:

- $\mathcal{CONVEX} = \{f \mid f(x) = \sum_{i=1}^{d} f_i(x_i)$, where each f_i is a convex function$\}$;
- $\mathcal{CONCAVE} = \{f \mid f(x) = \sum_{i=1}^{d} f_i(x_i)$, where each f_i is a concave function$\}$; and
- $\mathcal{CES} = \{f \mid f(x) = (\sum_{i=1}^{d} a_i x_i^b)^{\frac{1}{b}}$, where $b \in \mathbb{R}_+$ and $a_i \in \mathbb{R}_+$ for all $1 \leq i \leq d\}$.

EXAMPLE 3. *A set of risk-loving financial advisors are presented with k options for various financial packages. Their utility functions are known to be of the form $f(x) = \sum_{i=1}^{d} a_i x_i^2$ (where each $a_i \in \mathbb{R}_+$). Because each $f_i(x) = a_i x_i^2$ is convex, our bounds for $\mathcal{CONVEX}$ will apply.*

EXAMPLE 4. *A group of consumers are shown k different vacation packages. They experience diminishing marginal returns on any given attribute of a package. Their utility functions are known to be of the form $f(x) = \sum_{i=1}^{d} a_i \sqrt{x_i}$ (where each $a_i \in \mathbb{R}_+$). Because each $f_i(x) = a_i \sqrt{x_i}$ is a concave function, our bounds for $\mathcal{CONCAVE}$ will apply.*

3.2 Stability and Scale Invariance

Nanongkai et al. [23] defined two very important properties called stability and scale invariance. Intuitively, stability is the property that the addition of a point that is non-optimal for all users should not change the result of the query. The k-regret query is stable since non-optimal points have no bearing on the computation of regret and hence the set of points returned will not be affected by non-optimal ones.

The scale invariance property [23] informally means that re-scaling each dimension of the database by a constant should not affect the result of the query. For example, it should not matter if the fuel efficiency of a car is given in miles per gallon or kilometers per liter. It was shown by Nanongkai et al. [23] that the k-regret query for linear utilities is scale invariant. We extend this result to a much broader class of utility functions.

THEOREM 1. *Let $\mathcal{F}$ be the class of functions of the form $f(x) = \sum_{i=1}^{d} f_i(x_i)$, where each f_i is an infinitely differentiable at zero. Then, the maximum regret ratio for any set of points remains unchanged after any dimension is rescaled by the same constant.*

PROOF. Assume f to be any function from the class $\mathcal{F}$ defined in the statement of the theorem. We know that f can be written in the form $f(x) = \sum_{i=1}^{d} f_i(x_i)$, where each f_i has a Taylor series expansion around zero. Hence, we can re-write the function as $f(x) = \sum_{i=1}^{d} \sum_{j=0}^{\infty} a_{i,j} x_i^j$. Now, let us say that the attributes of the database are rescaled by the constants $\lambda_1, \ldots, \lambda_d$, respectively, to give a new database D'. We define another utility function

$$g(x) = \sum_{i=1}^{d} \sum_{j=0}^{\infty} \frac{a_{i,j}}{\lambda_i^j} x_i^j$$

and note that, by construction, the value of the utility function of f on any point in D is equal to the value of the utility function g on the corresponding point in D'. Hence, the regret ratio of f in D will also be equal to the regret ratio of g in D'. Since our goal is to bound the regret ratio of the maximum across all functions, and g is a function in the class $\mathcal{F}$, the maximum regret ratio must be the same. □

4. UPPER BOUNDS

In this section, we will introduce the MinWidth algorithm and show how it provides theoretical upper bounds on the regret ratios for convex, concave, and CES functions. We will prove that the maximum regret ratio for a large subset of convex functions is $O(1/k^{\frac{1}{d-1}})$. We will also prove that the maximum regret ratio for all concave functions is also $O(1/k^{\frac{1}{d-1}})$. Finally, we will prove that the maximum regret ratio for CES functions is $O(1/bk^{\frac{b}{d-1}})$ for $0 < b < 1$ and $O(1/k^{\frac{1}{b(d-1)}})$ for $b > 1$. Note that all these bounds are independent of the number of tuples in the database and the distribution of these tuples.

4.1 MinWidth Algorithm

The MinWidth algorithm (Algorithm 1) is an adaptation of the Cube algorithm given in [23], with a few additions to increase its efficacy on sparse datasets. The algorithm works as follows: after adding the maximal points in each dimension to the result set, it divides each dimension except for the last into $t = O(k^{1/(d-1)})$ variable-sized intervals. For each combination of these intervals, MinWidth includes the point that is largest in the last dimension in its solution. See Algorithm 1 for details.

The difference from the Cube algorithm is the way in which MinWidth selects the breakpoints for the intervals. While the Cube algorithm evenly divides the points into equal-sized intervals, MinWidth selects the interval breakpoints in dimension i in such a manner so as to minimize the maximum distance (along dimension i) between any two points in the interval. More precisely, given the values of the points in the ith dimension $\{p_1[i], \ldots, p_n[i]\}$, we wish to pick breakpoints $s_0 < s_1 < s_2 < \ldots < s_{t-1} < s_t$ such that $\max_{l=0,\ldots,t-1} \max_{p_u[i],p_v[i] \in [s_l, s_{l+1}]} |p_u[i] - p_v[i]|$ is minimized. The interval width can thus be bounded by, and in fact can be considerably smaller than, the width of intervals for the Cube algorithm, which was c_i/t, where c_i is the maximum value in the i^{th} dimension.

The breakpoints used by MinWidth are calculated as in Algorithm 2. FindG determines the breakpoints via a modified version of binary search. This adaption was added because one of Cube's drawbacks was that buckets would often be empty on sparse

Algorithm 1 MINWIDTH algorithm(D, k, n)

Input: D, a database of n d-dimensional points $\{p_1, p_2, ..., p_n\}$ and k, the number of points to output.
Output: A subset S of D with cardinality k.

```
1: For $i = 1, 2, ..., d-1$, let $c_i$ be the maximal value in the $i^{th}$
   dimension, and set $p_i^*$ to the corresponding point.
2: Let $S = \{p_i^*, p_2^*, ..., p_{d-1}^*\}$.
3: Let $t = \lfloor (k-d+1)^{\frac{1}{d-1}} \rfloor$.
4: **for** $i = 1, 2, ..., d-1$ **do**
5:     Let $bPoints[i] = \mathbf{findG}(D, t, n, i, c_i)$.
6: **for** each group of integers $0 \le j_1, \ldots, j_{d-1} < t$ **do**
7:     Let $S' = \{p \in D \mid bPoints[i][j_i] \le p[i] <
       bPoints[i][j_i + 1], 1 \le i < d\}$.
8:     Add the point in $S'$ that is maximal in the $d^{th}$ dimension to
       $S$.
9: **return** $S$.
```

datasets, and points would therefore be selected unevenly. FindG seeks to rectify this by determining the minimum distance between points that can be achieved with t buckets. FindG, after sorting the points based on their value in the i^{th} dimension, operates in a greedy fashion, continuously adding points to a possible bucket until the value of g as determined by the binary search has been exceeded. It then creates a "breakpoint" between the previous point and the point that exceeded the value of g. This ensures that no two points in the same interval are further apart than g. If there exists a feasible selection of such points with value g, then the greedy algorithm will find them as shown by the following argument.

Say that there exists a feasible solution for g, call it $s_0^* < s_1^* < \ldots < s_t^*$. Further, let us denote by $s_0 < s_1 < \ldots < s_t$ the first $t+1$ breakpoints selected via our greedy method. Note that $s_0^* = s_0 = \min_{p \in D} p[i]$ since we must include the smallest point in the i^{th} dimension between the breakpoints. We will show that the greedy solution is also feasible by a replacement argument. Let us say that the greedy and feasible solution are identical up to some index $j \ge 0$. The greedy algorithm selects s_{j+1} as the point with the largest value in the i^{th} dimension such that $s_{j+1}[i] - s_j[i] \le g$. Hence, for s^* to be feasible, it must be that $s_{j+1}^*[i] \le s_{j+1}[i]$. Since this is true, we can replace $s_{j+1}^*[i]$ with $s_{j+1}[i]$ in the feasible solution and it would still remain feasible. This means that we have a new feasible solution that agrees with the greedy solution up to the first $j+1$ breakpoints. Repeating this argument at most t times gives us that the greedy solution must also be feasible.

If the value of g cannot be achieved with t, a new value of g is found by binary search, and the process is repeated until the precision of g is arbitrarily sufficient. In our implementation, we executed the binary search for at least twenty iterations.

Once MinWidth has been given the breakpoints for every dimension, it then places all the points into $t^{d-1} \le (k-d+1)$ "buckets" and picks the point maximal in the last dimension in each bucket, adding them to the result set. This result set is then returned to the user. Given the value of t, MinWidth returns k points to the user. The choice of maximal points in each dimension, as well as the constraints on the maximum width of any given bucket ($\frac{c_i}{t}$ for dimension i, in the worst case), lead to our theoretical bounds for convex, concave, and CES functional types in the following sections. Although other algorithms may provide lower regret in practice, these properties of MinWidth allow us to prove the following bounds.

Running time: Step 1 of the algorithm takes $O(nd)$ time. Steps

Algorithm 2 FINDG algorithm(D, t, n, i, c_i)

Input: D, a database of d-dimensional points $\{p_1, p_2, ..., p_n\}$ sorted on the i^{th} dimension; t, the desired number of intervals; n, the total number of points; i, the dimension for which to assign boundaries; and c_i, the maximal value in the i^{th} dimension.
Output: $bPoints$, a set of $t+1$ boundaries in the i^{th} dimension.

```
1: Set $lowerbound = 0$ and $upperbound = \frac{c_i}{t}$.
2: while the precision of $g$ is insufficient, do
3:   $bPoints[0] = p_1$.
4:   Set $g$ to the average of $lowerbound$ and $upperbound$.
5:   Let $m = 1, j = 2$, and $h = 1$.
6:   while $p_j < c_i$ and $h < t$ do
7:     while $p_j - p_m \leq g$ do
8:       $j = j + 1$
9:     $m = j - 1$
10:    Set $bPoints[h] = p_m$.
11:    Set $h = h + 1$.
12:  if $c_i - p_m > g$ then
13:    $lowerbound = g$.
14:  else
15:    Set $bPoints[t] = p_n$.
16:    $upperbound = g$.
17:    Store this copy of $bPoints$.
18: return the last stored copy of $bPoints$.
```

4-5 can be accomplished in $O(n)$ time since the greedy algorithm (lines 3-17 of the FindG algorithm) takes linear time, assuming that we use constant precision for g (e.g., 0.01%). Lastly, steps 6-8 can be executed in $O(nd)$ time by computing which bucket each point falls into in $O(d)$ time. Hence, the overall running time is $O(nd)$. Note that any algorithm that processes all the data (n points of d dimensions each) will need $\Omega(nd)$ time to execute. Also note that we have assumed that the data is static; the case for dynamically changing data is outside the scope of this paper.

4.2 Proofs

All of the proofs in this section start with an arbitrary function in the class of functions being studied. We then bound the regret for the result of the MinWidth algorithm by comparing the optimal point for that function with the point picked in the same interval as the optimal. Finally, we show that including the extremal point in each of the first $d-1$ dimensions gives us a bound on the regret ratio via the following lemma.

LEMMA 2. *If the regret of the MinWidth algorithm for the function $f(x) = \sum_{j=1}^{d} f_j(x_j)$ is bounded by $\sigma * \max_{i \leq d-1} \sum_{j=1}^{d} f_j(p_i^*[j])$ for some $\sigma > 0$, then the regret ratio for the function is at most $\frac{\sigma}{1+\sigma}$.*

PROOF. Let f be a utility function $f(x) = \sum_{j=1}^{d} f_j(x_j)$ with regret $r_D(S, f) \leq \sigma * \max_{i \leq d-1} \sum_{j=1}^{d} f_j(p_i^*[j])$ when shown the result of the MinWidth algorithm S rather than the entire database D. Since $p_1^*, p_2^*, ...p_{d-1}^*$ are in S we know that $\max_{p \in S} f(p) \geq \max_{i \leq d-1} \sum_{j=1}^{d} f_j(p_i^*[j])$; thus, $r_D(S, f)/\max_{p \in S} f(p) \leq \sigma$. The regret ratio of f is hence bounded by

$$\begin{aligned} rr_D(S, f) &= \frac{r_D(S, f)}{\max_{p \in S} f(p) + r_D(S, f)} \\ &= \frac{1}{\max_{p \in S} f(p)/r_D(S, f) + 1} \\ &\leq \frac{1}{1/\sigma + 1} = \frac{\sigma}{1+\sigma}. \end{aligned}$$

□

4.3 Convex Upper Bound

The MinWidth Algorithm allows us to bound the regret for a large subset of $\mathcal{CONVEX}$. In this section, we will prove the following theorem:

THEOREM 3. *For any k, the regret ratio for monotonically increasing convex utility functions of the form $f(p) = \sum_{i=1}^{d} f_i(p_i)$ where each f_i is of the form $f_i(x) = x^b g_i(x)$, where g_i is a differentiable, non-increasing function, is at most $\frac{b(d-1)}{b(d-1)+t}$.*

PROOF. Let f be an arbitrary function of the form $f(p) = \sum_{i=1}^{d} f_i(p_i)$ where each f_i is of the form $f_i(x) = x^b g_i(x)$, where g_i is a differentiable, non-increasing function. For all of the proofs in this section, we will define p and s as follows. Let p be a point in D such that $f(p) = \max_{x \in D} f(x)$; that is, p maximizes f. Let s be the point selected by MinWidth in the same combination of intervals as p (see Line 8 of Algorithm 1).

We will first prove that $f(p) - f(s) \leq \frac{b(d-1)}{t} \max_{i \leq d-1} \sum_{j=1}^{d} f_j(p_i^*[j])$. We will show this through the following inequalities:

$$\begin{aligned} f(p) - f(s) &= \sum_{i=1}^{d} (f_i(p_i) - f_i(s_i)) \\ &\leq \sum_{i=1}^{d-1} (p_i - s_i) f_i'(p_i) && (1) \\ &\leq \sum_{i=1}^{d-1} \left(\frac{c_i}{t}\right) f_i'(c_i) \\ &\leq \frac{d-1}{t} \max_{i \leq (d-1)} (c_i) f_i'(c_i) \\ &\leq \frac{d-1}{t} \max_{i \leq (d-1)} b f_i(c_i) && (2) \\ &\leq \frac{b(d-1)}{t} \max_{i \leq (d-1)} \sum_{j=1}^{d} f_j(p_i^*[j]). \end{aligned}$$

The transition to equation 1 can be explained as follows: based on the Mean Value Theorem, we can be assured that there is a point a_i on the curve where $f_i'(a_i) = \frac{f(p_i) - f(s_i)}{p_i - s_i}$. Thus, $f_i'(a_i) * (p_i - s_i) = f_i(p_i) - f_i(s_i)$. Further, since a_i is between p_i and s_i, and $f'(x)$ is a monotonically increasing function for all convex functions, $f_i'(p_i) \geq f_i'(a_i)$. Inequality 2 follows from the fact that for all functions of the form $f_i(x) = x^b g_i(x)$, where g_i is a differentiable, non-increasing function, we have that $x f_i'(x) = b x^b g_i(x) + x g_i'(x) \leq b f_i(x)$ (since $g'(x) \leq 0$).

By Lemma 2, $rr_D(S, P) \leq \frac{b(d-1)}{b(d-1)+t}$. This bound is guaranteed for any subset of this sub-class of convex functions when we use the MinWidth algorithm. □

Consider a concrete example of this upper bound. Let $d = 2$ and let $f_1(x), f_2(x) \in \{c_0 + c_1 x + c_2 x^2 \mid c_0, c_1, c_2 \geq 0\}$. Thus $b = 2$.

If we let $k = 20$, then $t = k^{\frac{1}{d-1}} = 20$, and thus the regret ratio is at most $\frac{b(d-1)}{b(d-1)+t} = \frac{2}{2+20} < 10\%$.

Another example is when $f_1(x), f_2(x) \in \{cx^\beta \mid c > 0, 1 \leq \beta \leq 3\}$ ($d = 2$, $b = 3$). If $k = 20$, then the regret ratio is at most $\frac{b(d-1)}{b(d-1)+t} = \frac{3}{3+20} < 14\%$.

4.4 Concave Upper Bound

We can also prove a bound for concave functions. We will show the following theorem to be true:

THEOREM 4. *For any k, the regret ratio for utility functions of the form $f(p) = \sum_{i=1}^{d} f_i(p_i)$ where each f_i is some monotonically increasing,infinitely differentiable concave function such that each $f_i(0) \geq 0$ is at most $\frac{d-1}{(d-1)+t}$.*

PROOF. Let f be an arbitrary utility function of the form $f(p) = \sum_{i=1}^{d} f_i(p_i)$, where each f_i is some monotonically increasing, concave function with $f(0) \geq 0$.

Let p be the optimal point for f in D and let s be the point in the same interval, as in the previous proof. We will bound the regret using the Taylor expansion of the f_i functions and the following inequalities:

$$\begin{aligned}
f(p) - f(s) &= \sum_{i=1}^{d} (f_i(p_i) - f_i(s_i)) \\
&\leq \sum_{i=1}^{d-1} \sum_{n=0}^{\infty} \frac{f_i^n(0)}{n!}(p_i)^n - \sum_{n=0}^{\infty} \frac{f_i^n(0)}{n!}(s_i)^n \\
&= \sum_{i=1}^{d-1} \sum_{n=0}^{\infty} \frac{f_i^n(0)}{n!}(p_i)^n - \frac{f_i^n(0)}{n!}(s_i)^n \\
&= \sum_{i=1}^{d-1} \sum_{n=0}^{\infty} \frac{f_i^n(0)}{n!}((p_i)^n - (s_i)^n) \\
&\leq \sum_{i=1}^{d-1} \sum_{n=0}^{\infty} \frac{f_i^n(0)}{n!}(\frac{c_i}{t})(2c_i)^{n-1} \qquad (3) \\
&\leq \frac{(d-1)}{t} \max_{i \leq d-1} \sum_{n=0}^{\infty} \frac{f_i^n(0)}{n!}(c_i)(2c_i)^{n-1} \\
&\leq \frac{(d-1)}{2t} \max_{i \leq d-1} \sum_{n=0}^{\infty} \frac{f_i^n(0)}{n!}(2c_i)^{n} \\
&= \frac{d-1}{2t} \max_{i \leq d-1} f_i(2c_i) \\
&\leq \frac{d-1}{2t} \max_{i \leq d-1} 2 * f_i(c_i) \qquad (4) \\
&= \frac{d-1}{t} \max_{i \leq d-1} f_i(c_i) \\
&\leq \frac{d-1}{t} \max_{i \leq d-1} \sum_{j=1}^{d} f_j(p_j^*[j]).
\end{aligned}$$

The leap to 3 may be explained by factoring the previous statement into $(p_i - s_i)(p_i + s_i)^n$, and label 4 is readily explained by the property of subadditivity.

By Lemma 2, $rr_D(S, Q) \leq \frac{d-1}{(d-1)+t}$. This bound holds for all concave functions. $\square$

For example, let $d = 2$ and $k = 20$. Therefore $t = k^{\frac{1}{d-1}} = 20$. Thus the regret ratio is at most $\frac{d-1}{(d-1)+t} = \frac{1}{1+20} < 5\%$.

4.5 CES Upper Bound

The MinWidth algorithm can also be used to prove a bound for the CES functions. We will first prove the following theorem for CES functions with $0 < b < 1$.

THEOREM 5. *For any k, the regret ratio for CES utility functions with $0 < b < 1$ is at most $\frac{d^{\frac{1}{b}}}{d^{\frac{1}{b}} + bt^b}$.*

PROOF. Let f be the utility function of a user, where $f(p) = \left(\sum_{i=1}^{d} a_i p_i^b\right)^{\frac{1}{b}}$, $0 < b < 1$; that is, $f(p)$ is a CES function with $0 < b < 1$.

Let p and s be defined as in the previous proofs. We will first prove that $f(p) - f(s) \leq \frac{(d-1)^{\frac{1}{b}}}{bt^b} \left(\max_{i \leq d} \sum_{j=1}^{d} a_j p_{i[j]}^{*b}\right)^{\frac{1}{b}}$. We show this through the following inequalities:

$$\begin{aligned}
f(p) - f(s) &= \left(\sum_{i=1}^{d} a_i p_i^b\right)^{\frac{1}{b}} - \left(\sum_{i=1}^{d} a_i s_i^b\right)^{\frac{1}{b}} \\
&\leq \left(\sum_{i=1}^{d} a_i p_i^b - \sum_{i=1}^{d} a_i s_i^b\right) * \frac{1}{b} * \left(\sum_{i=1}^{d} a_i p_i^b\right)^{\frac{1}{b}-1} \qquad (5) \\
&\leq \frac{1}{b} * \left(\sum_{i=1}^{d} a_i (p_i^b - s_i^b)\right) \left(\sum_{i=1}^{d} a_i p_i^b\right)^{\frac{1}{b}-1} \\
&\leq \frac{1}{b} * \left(\sum_{i=1}^{d-1} a_i (p_i^b - s_i^b)\right) \left(\sum_{i=1}^{d-1} a_i p_i^b\right)^{\frac{1}{b}-1} \\
&\leq \frac{1}{b} \left(\sum_{i=1}^{d-1} a_i (p_i - s_i)^b\right) \left(\sum_{i=1}^{d-1} a_i p_i^b\right)^{\frac{1}{b}-1} \qquad (6) \\
&\leq \frac{1}{b} \left(\sum_{i=1}^{d-1} a_i \left(\frac{c_i}{t}\right)^b\right) \left(\sum_{i=1}^{d-1} a_i p_i^b\right)^{\frac{1}{b}-1} \\
&= \frac{d-1}{bt^b} \left(\max_{i \leq d-1} a_i c_i^b\right) \left((d-1) \max_{i \leq d} a_i c_i^b\right)^{\frac{1}{b}-1} \\
&= \frac{(d-1)^{\frac{1}{b}}}{bt^b} \left(\max_{i \leq d-1} a_i c_i^b\right)^{\frac{1}{b}} \\
&\leq \frac{(d-1)^{\frac{1}{b}}}{bt^b} \left(max_{i \leq d-1} \sum_{j=1}^{d} a_j p_{i[j]}^{*b}\right)^{\frac{1}{b}}.
\end{aligned}$$

Equation 5 is proved by the following: $g(x) = x^{\frac{1}{b}}$ is a convex function for $0 < b < 1$. Because of the convexity of g, we know that $g(x) - g(y)$ is bounded by $(x - y)f'(x)$, its first order Taylor approximation, due to Jensen's inequality.

Equation 6 is true since $h(x) = x^b$ is a concave function when $0 < b < 1$, and thus $h(x) - h(y) \leq h(x - y)$. By a proof similar to that of Lemma 2, $rr_D(S, C) \leq \frac{(d-1)^{\frac{1}{b}}}{(d-1)^{\frac{1}{b}} + bt^b}$. $\square$

For an example of this upper bound for CES utility functions with $0 < b < 1$, let $b = 0.75$, $d = 2$, and $k = 20$. Thus $t = k^{\frac{1}{d-1}} = 20$. Therefore the regret ratio is at most $\frac{d^{\frac{1}{b}}}{d^{\frac{1}{b}} + bt^b} = \frac{2^{4/3}}{2^{4/3} + 0.75 * 20^{0.75}} < 27\%$.

Finally, we will prove the upper bound for CES functions with $b > 1$.

THEOREM 6. *For any k, the regret ratio for CES utility functions with $b > 1$ is at most $\frac{(b(d-1))^{\frac{1}{b}}}{(b(d-1))^{\frac{1}{b}} + t^{\frac{1}{b}}}$.*

PROOF. Let f be a CES function with $b > 1$. Let p and s be as in the previous proofs. We will first prove that $f(p) - f(s) \leq \left(\frac{b(d-1)}{t}\right)^{\frac{1}{b}} \left(\max_{i \leq d-1} \sum_{j=1}^{d} a_i {p^*_{i[j]}}^b\right)^{\frac{1}{b}}$. We show this through the following inequalities:

$$\begin{aligned}
f(p) - f(s) &= \left(\sum_{i=1}^{d} a_i p_i^b\right)^{\frac{1}{b}} - \left(\sum_{i=1}^{d} a_i s_i^b\right)^{\frac{1}{b}} \\
&\leq \left(\sum_{i=1}^{d-1} a_i (p_i^b - s_i^b)\right)^{\frac{1}{b}} \\
&\leq \left(\sum_{i=1}^{d-1} a_i (p_i - s_i)(b p_i^{b-1})\right)^{\frac{1}{b}} \qquad (7) \\
&\leq \left(\sum_{i=1}^{d-1} a_i \left(\frac{c_i}{t} b c_i^{b-1}\right)\right)^{\frac{1}{b}} \\
&\leq \left(\frac{d-1}{t} \max_{i \leq d-1} a_i b c_i^b\right)^{\frac{1}{b}} \\
&\leq \left(\frac{b(d-1)}{t} \max_{i \leq d-1} \sum_{j=1}^{d} a_i {p^*_{i[j]}}^b\right)^{\frac{1}{b}} \\
&= \left(\frac{b(d-1)}{t}\right)^{\frac{1}{b}} \left(\max_{i \leq d-1} \sum_{j=1}^{d} a_i {p^*_{i[j]}}^b\right)^{\frac{1}{b}}
\end{aligned}$$

where 7 and its consequent equations follow from the convex bound proof. We use two facts to prove the transition. First, notice that $a_d s_d^b \geq a_d p_d^b$, and let $a_d s_d^b - a_d p_d^b = z$ where $z \geq 0$. Second, since $g(x) = x^{\frac{1}{b}}$ is a concave function, $g(x) + g(y) \geq g(x + y)$; thus $g(x) - g(y) \leq g(x - y)$. Hence $x^{\frac{1}{b}} - y^{\frac{1}{b}} \leq (x + y)^{\frac{1}{b}}$. Using these two facts, the following inequalities are true:

$$\begin{aligned}
&\left(\sum_{i=1}^{d-1} a_i p_i^b + a_d p_d^b\right)^{\frac{1}{b}} - \left(\sum_{i=1}^{d-1} a_i s_i^b + a_d s_d^b\right)^{\frac{1}{b}} \\
\leq\ &\left(\sum_{i=1}^{d-1} a_i p_i^b + a_d p_d^b - \sum_{i=1}^{d-1} a_i s_i^b - a_d s_d^b\right)^{\frac{1}{b}} \\
\leq\ &\left(\sum_{i=1}^{d-1} a_i p_i^b - \sum_{i=1}^{d-1} a_i s_i^b - z\right)^{\frac{1}{b}} \\
\leq\ &\left(\sum_{i=1}^{d-1} a_i p_i^b - \sum_{i=1}^{d-1} a_i s_i^b\right)^{\frac{1}{b}}.
\end{aligned}$$

By a proof similar to that of Lemma 2, $rr_D(S, M) \leq \frac{(b(d-1))^{\frac{1}{b}}}{(b(d-1))^{\frac{1}{b}} + t^{\frac{1}{b}}}$. □

Consider a concrete example of an upper bound for CES utility functions with $b > 1$. Let $b = 2$, $d = 2$, and $k = 20$. Thus $t = k^{\frac{1}{d-1}} = 20$. Therefore the regret ratio is at most $\frac{(b(d-1))^{\frac{1}{b}}}{(b(d-1))^{\frac{1}{b}} + t^{\frac{1}{b}}} = \frac{\sqrt{2}}{\sqrt{2} + \sqrt{20}} < 25\%$.

Algorithm 3 AREA-GREEDY algorithm(D, k, n)

Input: D, a database of n d-dimensional points $\{p_1, p_2, ..., p_n\}$ and k, the number of points to output.
Output: A subset S of D with cardinality k.

```
1:  For $i = 1, 2, ..., d-1$, let $p_i^*$ be the point with the largest value
    in the $i^{th}$ dimension.
2:  Let $S = \{p_i^*, p_2^*, ..., p_{d-1}^*\}$.
3:  while $||S|| < k$ do
4:      $A^* = 0$,
5:      for each $p \in D$ do
6:          Let $A$ be the area bounded by the axes and $S \cup \{p\}$
7:          if $p \notin S$ and $A > A^*$ then
8:              Let $A^* = A$ and $p^\omega = p$
9:      $S = S \cup \{p^\omega\}$
10: return $S$.
```

Note that for $b = 1$, CES functions become linear functions. From the proofs in this section, it follows that the MinWidth algorithm is capable of bounding the regret for a large set of convex functions and concave functions as well as all CES functions with $b > 0$.

5. HEURISTICS

In addition to the theoretical guarantee of MinWidth, we provide here two heuristics which yield low regret-ratios when evaluated.

5.1 Area-Greedy

The first heuristic is Area-Greedy (Algorithm 3), a greedy-style heuristic, which can be informally described as follows: firstly, as in MinWidth, it picks the maximal points in every dimension and adds them to the solution. Once this is done, it greedily adds the point to the solution that greatest increases the area under the points in the solution, repeating this process until a subset of k points is returned. Due to the limited number of points, it is possible to find these integrals via simple calculations of the quadrature of areas, generalizing this to higher dimensions the same way one generalizes finding the area of a rectangle to finding the area of a rectangular prism.

This primary rationale behind this algorithm's design is that it effectively creates a solution set dependent on the distance between points, which we will show is an important aspect for the lower bound to this problem (see Section 6). Other heuristics have proposed similar ideas, but they have fallen short due to their issues with the earlier-discussed property of stability. Area-Greedy neatly avoids this problem, and therefore is able to give very low regret-ratios in practice, especially on CES utility functions.
Running time: The Area-Greedy algorithm is the most computationally intensive of the ones studied in this paper. It iterates k times over the n points and takes $O(kd)$ time to compute the area in each iteration, leading to a total running time of $O(ndk^2)$.

5.2 Angle

The second heuristic is called Angle. First, the Angle algorithm uses polar coordinates to divide every dimension into t equidistant angles with the x-axis, where t is initialized to $k^{\frac{1}{d-1}}$. It then searches through every combination for each angle and identifies the farthest point in that direction, adding it to the solution. See Algorithm 4 for details. Unfortunately, it is often the case that this algorithm gives too few distinct points as some points will be the far-

Algorithm 4 ANGLEPOINTS algorithm(D, n, t)

Input: D, a database of n d-dimensional points $\{p_1, p_2, ..., p_n\}$ and t, an integer parameter.
Output: A subset S of D

```
1: Let $S = \emptyset$
2: for each $0 \leq b_2, \ldots, b_d < t$ do
3:     Let $\vec{v}$ be the vector defined by setting the angle between the
       $i$th dimension and the x-axis to $\frac{\pi \cdot b_i}{2t}$, for each $1 < i \leq d$.
4:     Let $p^*$ be the point in $D$ that is farthest in the direction $\vec{v}$.
5:     Let $S = S \cup \{p^*\}$.
6: return $S$
```

Algorithm 5 ANGLE algorithm(D, k, n)

Input: D, a database of n d-dimensional points $\{p_1, p_2, ..., p_n\}$ and k, the number of points to output.
Output: A subset S of D with cardinality k.

```
1: Let $t = \lfloor k^{\frac{1}{d-1}} \rfloor$.
2: Let $S = \text{AnglePoints}(D, n, t)$.
3: while $||S|| < k$ do
4:     $t = t + 1$.
5:     Let $S = \text{AnglePoints}(D, n, t)$.
6: return S
```

thest in many different directions. To get around this, we increase the value of t until we get k distinct points (see Algorithm 5).

This heuristic is very useful because its design leads it to naturally pick points that yield low regret ratios for convex functions. Convex functions heavily favor points that have close to the maximum value in some (or perhaps multiple) dimensions, and since these points are farthest out in any given direction, these are the points Angle tends to select. Contrast this with MinWidth and Area-Greedy, which attempt to optimize for an even spread of points. Based on this reasoning, one expects Angle to outperform them both with regards to convex functions.

Running time: The Angle algorithm takes $O(ndk)$ time to execute as it finds the optimal point in each of k different directions. This algorithm, unlike the others, has the advantage that it is easy to update when new tuples are inserted—each insertion takes just $O(kd)$ time to check if should replace one of the k points.

6. LOWER BOUNDS

In this section, we present maximum regret ratio lower bounds for convex, CES, and concave utility functions. In each case we show that the maximum regret ratio must be at least some function of the number of points, k. We start with the result for convex functions:

THEOREM 7. *For any $d \geq 2, b \geq 1$, there exists a database of d-dimensional points such that any algorithm that displays k points must have a regret ratio of at least $\frac{1}{2}\left(\frac{1}{2^d k}\right)^{2b/(d-1)}$ for $\mathcal{F} = \{f(x) = \sum_{i=1}^d a_i x_i^b\}_{a_i \in \mathbb{R}_+}$, a subset of the convex class of functions.*

PROOF. Fix any $d \geq 2, b \geq 1$ and define $\mathcal{F} = \{f(x) = \sum_{i=1}^d a_i x_i^b\}_{a_i \in \mathbb{R}_+}$. Note that $\mathcal{F}$ is a subset of the class of convex functions as $b \geq 1$. Our database consists of the set of points in the set $D = \{(p_1, \ldots, p_d) \in \mathbb{R}_+^d \mid \sum_{i=1}^d p_i^{2b} = 1\}$. In other words, D is the set of d-dimensional points on the positive part of the unit super-sphere of degree $2b$ in d dimensions.

Let $\epsilon > 0$ be the maximum regret ratio for a set of k points S. We claim that for every point p in the database, there must exist some point q among S such that $\sum_{i=1}^d (p_i - q_i)^{2b} \leq 2\epsilon$. We prove this claim using contradiction by assuming that it is not the case. Fix a point p in the database but not in the set of k points and another point q that is in the set of k points. We know from the definition of D that $\sum_{i=1}^d p_i^{2b} = 1$, $\sum_{i=1}^d q_i^{2b} = 1$, and by our assumption that $\sum_{i=1}^d (p_i - q_i)^{2b} > 2\epsilon$. We consider the utility function $f(x) = \sum_{i=1}^d p_i^b x_i^b$ from $\mathcal{F}$. The utility of p is $f(p) = \sum_{i=1}^d p_i^{2b} = 1$. The utility of q on the other hand can be bounded as:

$$
\begin{aligned}
f(q) &= \sum_{i=1}^d q_i^b p_i^b \leq \sum_{i=1}^d \frac{1}{2}(p_i^{2b} + q_i^{2b} - (p_i - q_i)^{2b}) \text{ (induction on } b) \\
&= \frac{1}{2}(2 - \sum_{i=1}^d (p_i - q_i)^{2b}) < 1 - \epsilon.
\end{aligned}
$$

The regret for point p is hence greater than $(f(p) - f(q))/f(p) = 1 - (1 - \epsilon) = \epsilon$, which is a contradiction, proving our claim.

Now that we have established our claim that no point in the database can be too far (in terms of ℓ_{2b} distance) from one of the k points, we can establish a bound on the number of points needed. It is known that the a d-dimensional n-degree super-sphere of radius r has a surface area of $C_{d,n} r^{d-1}$, for some constant $C_{d,n}$ that depends only on d and n (and not r). The surface area of the points in the database D is hence $C_{d,n}/2^d$ since it consists of the all positive points in the unit super-sphere. For the k points to be not too far from the points in the database, the surface area of the super-sphere of radius $(2\epsilon)^{1/2b}$ centered at each of the k points must cover the entire database. The surface area of each of these is $C_{d,n}(2\epsilon)^{(d-1)/2b}$. Hence, there should be at least

$$
k \geq \frac{C_{d,n}/2^d}{C_{d,n}(2\epsilon)^{(d-1)/2b}} = \frac{1}{2^d (2\epsilon)^{(d-1)/2b}}
$$

points shown. Solving for ϵ, we get that using k points can only guarantee a regret of $\frac{1}{2}\left(\frac{1}{2^d k}\right)^{2b/(d-1)}$, completing the proof. □

Observe that when b and d are constant in the proof above, the regret ratio is effectively $\Omega(\frac{1}{k^{2b/(d-1)}})$. Compare this to the upper bound for convex functions from Section 4 of $O(\frac{1}{k^{1/(d-1)}})$ (once again assuming constant b and d). We leave as future work the problem of closing the $2b$ gap in the exponent.

Next, we show that for the class of CES utility functions with exponent b we cannot hope to bound the regret ratio to less than $O(1/bk^2)$ in the two-dimensional case. This is demonstrated by the following theorem.

THEOREM 8. *For any k, there is a database of two-dimensional points such that for any subset of k points in the database the regret ratio for CES utility functions with exponent $b > 0$ is at least $\Omega(1/bk^2)$.*

PROOF. Consider the database of points $D = \{(x, y) \mid x^{2b} + y^{2b} = 1, x, y \geq 0\}$. These points can also be represented parametrically as $\{(\cos^{1/b}\theta, \sin^{1/b}\theta) \mid \theta \in [0, \pi/2]\}$. Now, consider any set S of k of these points defined by their angles: $0 \leq \theta_1 < \theta_2 < \ldots < \theta_k \leq \pi/2$. We also define the angles $\theta_0 = 0$ and $\theta_{k+1} = \pi/2$. It must be the case that for some $0 \leq i \leq k$, $\theta_{i+1} - \theta_i \geq \frac{\pi}{2(k+1)}$. This is because there are $(k+1)$ angles that all sum to $\pi/2$, so they all cannot be strictly less than $\frac{\pi}{2(k+1)}$. Fix i to be some value such that $\theta_{i+1} - \theta_i \geq \frac{\pi}{2(k+1)}$ and let $\theta = (\theta_{i+1} + \theta_i)/2$. We will show that there exists a CES function for which we will have high regret when not including the point in the database corresponding to the angle θ.

Let $(\hat{x}, \hat{y})$ be the point corresponding to the θ defined above, i.e., $\hat{x} = \cos^{1/b}\theta, \hat{y} = \sin^{1/b}\theta$. Consider the CES utility function $f(x, y) = \left(\frac{\hat{x}^b}{\hat{x}^b+\hat{y}^b} \cdot x^b + \frac{\hat{y}^b}{\hat{x}^b+\hat{y}^b} \cdot y^b\right)^{1/b}$. The utility for the point $(\hat{x}, \hat{y})$ is $f(\hat{x}, \hat{y}) = \left(\frac{\hat{x}^{2b}}{\hat{x}^b+\hat{y}^b} + \frac{\hat{y}^{2b}}{\hat{x}^b+\hat{y}^b}\right)^{1/b} = (\frac{1}{\hat{x}^b+\hat{y}^b})^{1/b}$. In contrast, the utility obtained from any point $(\cos^{1/b}\theta_j, \sin^{1/b}\theta_j)$, where $|\theta_j - \theta| = \Delta \geq \frac{\pi}{4(k+1)}$, will be

$$\begin{aligned} & f(\cos^{1/b}\theta_j, \sin^{1/b}\theta_j) \\ = & \left(\frac{\hat{x}^b}{\hat{x}^b+\hat{y}^b} \cdot (\cos^{1/b}\theta_j)^b + \frac{\hat{y}^b}{\hat{x}^b+\hat{y}^b} \cdot (\sin^{1/b}\theta_j)^b\right)^{1/b} \\ = & \left(\frac{1}{\hat{x}^b+\hat{y}^b}\right)^{1/b} (\cos\theta\cos\theta_j + \sin\theta\sin\theta_j)^{1/b} \\ = & \left(\frac{1}{\hat{x}^b+\hat{y}^b}\right)^{1/b} (\cos\theta\cos(\theta+\Delta) + \sin\theta\sin(\theta+\Delta))^{1/b} \\ = & \left(\frac{1}{\hat{x}^b+\hat{y}^b}\right)^{1/b} (\cos^2\theta\cos\Delta + \sin^2\theta\cos\Delta)^{1/b} \\ = & \left(\frac{1}{\hat{x}^b+\hat{y}^b}\right)^{1/b} (\cos\Delta)^{1/b}. \end{aligned}$$

Hence, the regret ratio for this point will be

$$\left(\left(\frac{1}{\hat{x}^b+\hat{y}^b}\right)^{1/b} - \left(\frac{1}{\hat{x}^b+\hat{y}^b}\right)^{1/b} (\cos\Delta)^{1/b}\right) / \left(\frac{1}{\hat{x}^b+\hat{y}^b}\right)^{1/b} = 1 - (\cos\Delta)^{1/b}.$$

We can now use the MacLaurin series expansion of $\cos^n x$:

$$\cos^n x = 1 - \frac{nx^2}{2} + \frac{n(3n-2)x^4}{24} - \ldots$$

to bound this regret by $\frac{\Delta^2}{2b} - O(\Delta^4/b^2) = \Omega(\Delta^2/b)$. Lastly, we use the fact that $\Delta \geq \frac{\pi}{4(k+1)}$ to get that this regret ratio is $\Omega(1/bk^2)$. $\square$

Lastly, for the concave case we have the following result:

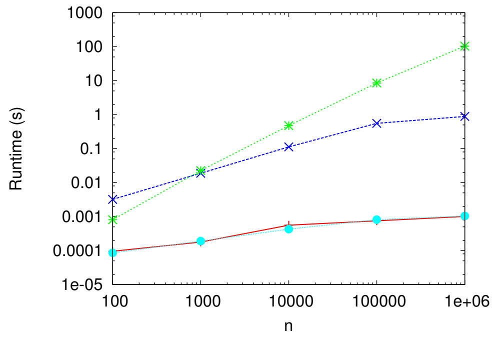
THEOREM 9. *For any k, there exists a database of two-dimensional points such that for any subset of k points in the database the regret ratio of the class $\mathcal{F} = \{f(x) = \sum_{i=1}^{d} a_i x_i^b\}_{a_i \in \mathbb{R}_+}$ $(0 < b < 1)$, a subset of the concave class of functions, is at least $\Omega(\frac{1}{k^2})$.*

The proof of this theorem is nearly identical to that of the theorem for the CES case and is omitted to avoid redundancy. This bound is off by a quadratic factor from the $O(\frac{1}{k})$ bound (for the case $d = 2$) from Section 4, and we leave the closing of this gap to future work.

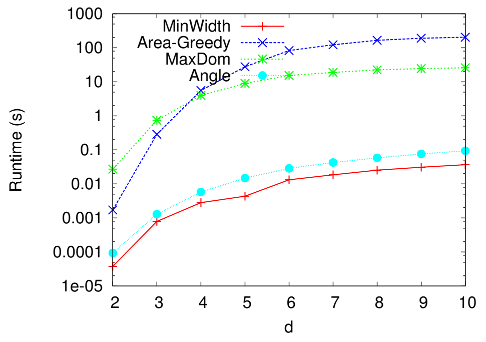
7. EXPERIMENTAL EVALUATION

In Section 4, we proved bounds on concave, convex, and CES functions using the MinWidth algorithm. These proofs show that, when using the MinWidth algorithm, the regret can always be bounded. While it is important to be able to prove these bounds in theory, we also want to show that regret can be quite small in practice. To this end, we provide experimental results in this section.

To evaluate our functions across convex, concave, and CES functions, we randomly generate 10,000 different functions of each type and compute the regret ratio for the chosen algorithm, taking the greatest of these as the maximum regret ratio. Although this method does not guarantee a bound, in practice we have found that it approximates the maximum regret ratio to a high degree of accuracy.



(a) Varying n, d = 3



(b) Varying d, n = 10,000

Figure 2: Runtimes, Anti-correlated, k = 20 (Note: y-axis is in logscale)

We implemented all algorithms in C++, and ran the runtime tests on a 64-bit 3.20GHz HP EliteDesk 800 machine which was running Ubuntu 3.13. Since there was some variation between each trial, we ran every experiment one hundred times and averaged the data over these trials.

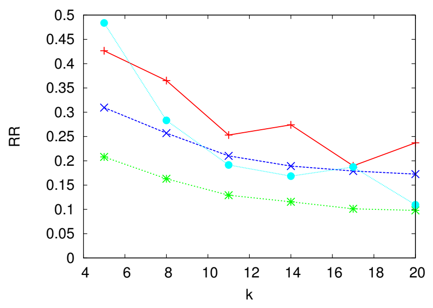
7.1 Datasets

We chose to run our experiments on both real-world and synthetic anti-correlated data created using the dataset generator of Börzsönyi et al. [2]. Our synthetic datasets have between 2 and 10 dimensions, 100 and 1,000,000 points, and our queries returned anywhere from 4 to 20 points. The two real-world datasets we have chosen are "NBA", which is a 5-dimensional dataset consisting of 17,264 points, and "Color", which is a 9-dimensional dataset containing 68,040 points.

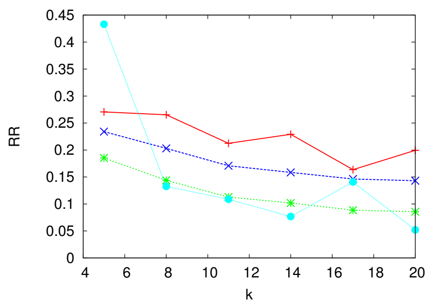
7.2 Algorithms

In our tests, we included

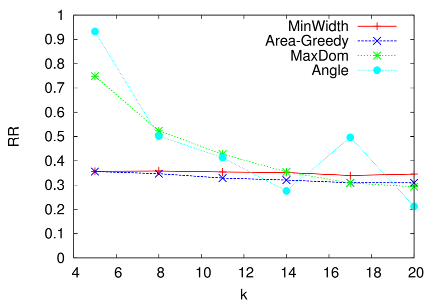
1. the MinWidth algorithm (Algorithm 1) that has provable bounds, and functions effectively across all types of datasets,
2. our Area-Greedy heuristic (Algorithm 3), which provides strongest results on convex utility functions, due to its usage of a distance-based heuristic to avoid outliers in the data,
3. our Angle heuristic (Algorithm 5), which functions well across all types of functions, particularly convex functions, when given a sufficient k value, and
4. the Max-Dom Greedy heuristic which was shown to perform well for linear utilities by Nanongkai et al. [23]. Originally



(a) CES

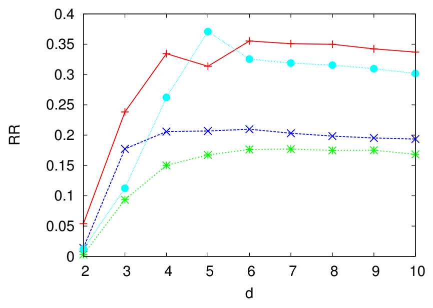


(b) Concave

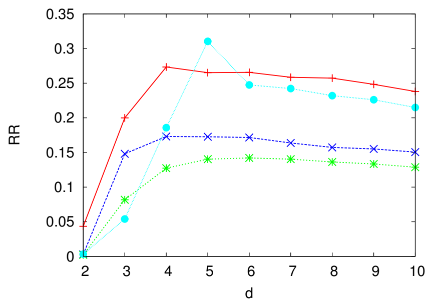


(c) Convex

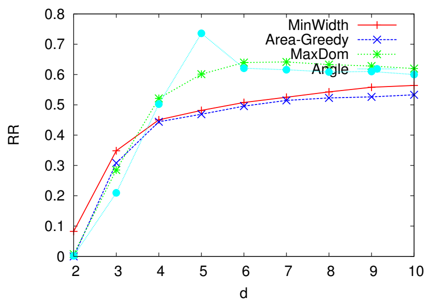
Figure 3: Anti-correlated, 3D, n = 10,000, Varying k



(a) CES



(b) Concave



(c) Convex

Figure 4: Anti-correlated, Varying d, n = 10,000, k = 20

proposed by Lin et al. [18], it seeks to pick the points on the skyline that dominate the most points overall, and returns k of those points. Although the algorithm does not exhibit the property of stability, it seems to work well in practice. Additionally, we found that, because of Max-Dom Greedy's tendency to select ideal points from the center of the dataset versus the edges, it performs especially well on Concave and CES utility functions.

7.3 Running Times

Our MinWidth and Angle algorithms have negligible running time, consistently completing in less than a second even for very large inputs. However, Max-Dom Greedy and Area-Greedy have more problematic running times. As can be seen in Figure 2(a), Max-Dom Greedy takes significantly longer as the database size increases, since it performs a skyline-style operation that is computationally expensive for additional points. Area-Greedy runs very slowly in higher dimensions (Figure 2(b)) due to its method of integrating higher-dimensional spaces. Therefore, while both algorithms are high-performance, their running time proves to be a significant drawback. In further work, however, the runtime of the Area-Greedy algorithm could be greatly reduced by incorporating dynamic programming techniques, and so this barrier to implementation could be eliminated in the future.

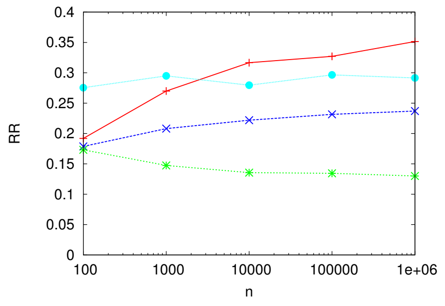
7.4 Maximum Regret Ratios

Referring back to the upper bounds on regret ratios from Section 4, we can see that the experimental results verify these bounds by providing even lower regret ratios in practice than was predicted by theory. The results demonstrate the expected trends within a slight range of error, particularly on convex functions, which displayed the greatest variation between regret ratios on similar datasets.

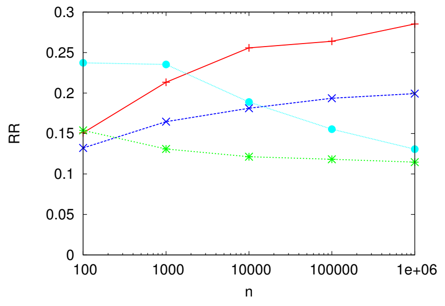
As expected from the proven bounds as well as previous experimental results, all the algorithms in Figure 3 show lower regret ratios with higher k. The lone aberration at some points is Angle, which occasionally demonstrates somewhat erratic behavior with respect to k. This is because the points that Angle picks can vary drastically with even a slight increase in k, since altering k changes the directions in which Angle chooses its points. As k became larger, however, Angle eventually proved more effective than the other algorithms across all function types.

Interestingly, in Figures 4(a) and 4(b), both Area-Greedy and Max-Dom Greedy retained very similar regret ratios, despite the increase in d. Angle and MinWidth, however, diverged early on, with Angle producing its anticipated wild behavior. All of the algorithms in Figure 4, though, produced an interesting result: there was a clear increase in regret ratios as d increased to 5, but beyond that point, there was only a slight upward inflection among the algorithms, i.e. the rate of growth leveled out. This trend can be explained by the fact that, as the number of dimensions increases, extreme points in the dataset are the user's ideal point less often, due to the fact that there are so many more attributes that must be optimized. Ergo, the utility lost by not choosing the ideal point is diminished, and so the regret ratios grow more slowly as d becomes larger.

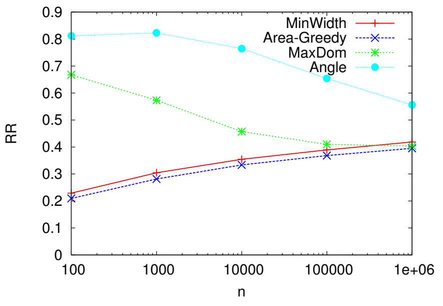
Figure 5 displays a few interesting trends: while the regret ratios are relatively stable, which is consistent with our upper bounds that are independent of database size, Angle and Max-Dom Greedy both decrease with respect to n. This is especially notable in Figure 5(a) and Figure 5(c). The reason for this is the algorithm's method of point selection. Max-Dom Greedy selects the points



(a) CES

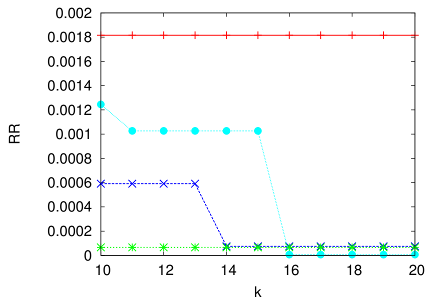


(b) Concave

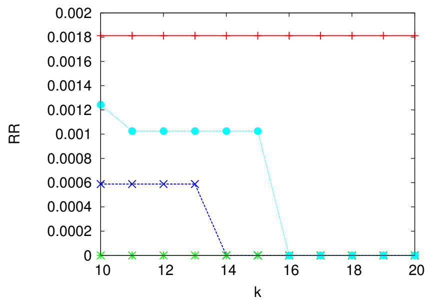


(c) Convex

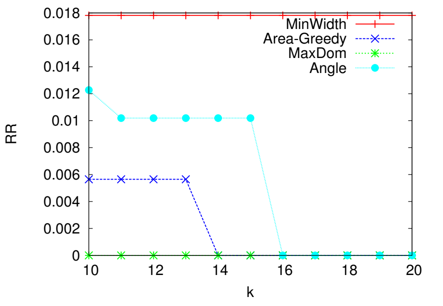
Figure 5: Anti-correlated, 3D, Varying n, k = 10



(a) CES



(b) Concave



(c) Convex

Figure 6: NBA, 5D, n = 17,264, varying k

which dominate the most other points in the dataset. When there are comparatively few points, this heuristic is less reliable, but as the number of points increases, its reliability increases. Angle exhibits similar tendencies, as each iteration through it seeks to pick the maximal point in a specific direction. When there are fewer points, however, Angle must choose a point that is close to, but not truly in the path of, its ideal direction. With more points, however, this concern is negated.

All of our algorithms performed extremely well on the NBA dataset (Figure 6). None of the regret ratios ever went higher than 2%. The highest regret ratio was for convex regret ratios (Figure 6(c)), though even this was for MinWidth, not any of the heuristics which proved effective in practice. The efficacy of our operators on this dataset can best be explained by the limited number of points in the skyline (253) and the closeness of such points. Given that this approximates a real-world scenario, however, we can count this as a great success, showing that even convex regret ratios can be usefully bounded in practice.

The Color dataset (Figure 7) shows that the differences between algorithms can be significant. For example, even as k is increased, Figure 7(a) shows very little variation, and there is a consistent difference of at least 10% between MinWidth and Area-Greedy. A similar effect holds true for the concave regret ratios in Figure 7(b), as well as for the convex ratios in Figure 7(c). In this last case, however, Max-Dom Greedy is clearly defeated for convex utility functions by Angle, Area-Greedy, and even MinWidth, which it performed as well or better than in synthetic data.

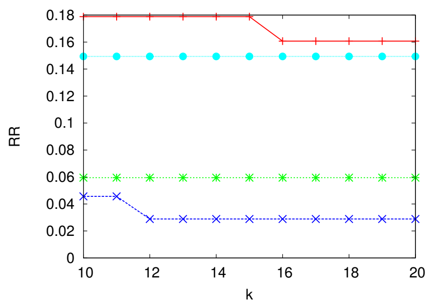
Overall, the regret ratios from convex functions were still the most significant, as per our original expectations. Angle proved most efficacious at minimizing these regret ratios when given sufficient points, although Area-Greedy and MinWidth proved effective no matter the k value. Max-Dom Greedy, however, was the best performing heuristic for concave and CES utilities. In all cases, we were able to show that the maximum regret ratio can be much smaller than the theoretical bounds.

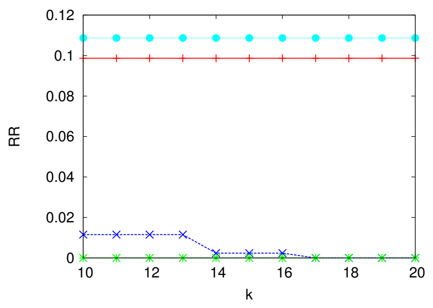
8. CONCLUSIONS

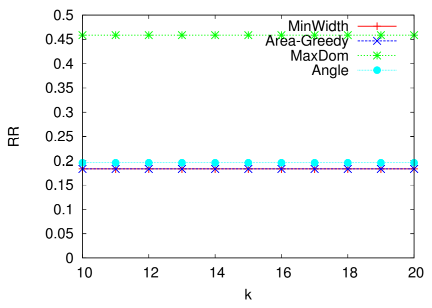
In this paper, we have expanded on previous research on representative databases and k-regret queries. We generalized the models of user preference, showing that the maximum regret ratio can be bounded for large classes of non-linear utility functions independent of the number of tuples in the database. In light of these results, we proposed several new algorithms which give very low regret for a broad range of non-linear utility functions both in theory and practice. We also provided new lower bounds for this problem that take into account nonlinear functions.

We performed simulations on real and synthetic data, evaluating the maximum regret ratios of our new classes of functions. We showed that the majority of the algorithms perform well against concave utility functions, but there was much greater variation amongst CES and convex utility functions, with Area-Greedy and Angle proving superior in these areas.

Based on our results, we believe there is the possibility of further research in expanding the class of utility functions to non-independent concave and convex utility functions, such as Cobb-Douglas functions, exponential functions, and submodular utility functions. There is also interesting work to be done considering the regret ratios of independent functions containing both concave as well as convex terms. Furthermore, it is possible that similar techniques to those shown in this paper can extend other representative database techniques that assume linear utility functions.


(a) CES


(b) Concave



(c) Convex

Figure 7: Color, 9D, n = 68,040, varying k

9. ACKNOWLEDGMENTS

This work was supported in part by Anderson-Bowen and Hodgson Summer Research Assistantships. The authors would like to thank the anonymous reviewers for their comments that improved the quality of the paper.

10. REFERENCES

[1] O. Barndorff-Nielsen and M. Sobel. On the distribution of the number of admissible points in a vector random sample. *Theory of Probability and its Applications*, 11(2):249–269, 1966.

[2] S. Börzsönyi, D. Kossmann, and K. Stocker. The skyline operator. In *ICDE*, 2001.

[3] 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.

[4] C. Y. Chan, H. V. Jagadish, K.-L. Tan, A. K. H. Tung, and Z. Zhang. Finding k-dominant skylines in high dimensional space. In *SIGMOD*, 2006.

[5] C. Y. Chan, H. V. Jagadish, K.-L. Tan, A. K. H. Tung, and Z. Zhang. On high dimensional skylines. In *EDBT*, 2006.

[6] Y.-C. Chang, L. D. Bergman, V. Castelli, C.-S. Li, M.-L. Lo, and J. R. Smith. The onion technique: Indexing for linear optimization queries. In *SIGMOD*, 2000.

[7] S. Chester, A. Thomo, S. Venkatesh, and S. Whitesides. Computing k-regret minimizing sets. *VLDB*, 7(5), 2014.

[8] A. Das Sarma, A. Lall, D. Nanongkai, R. J. Lipton, and J. Xu. Representative skylines using threshold-based preference distributions. In *ICDE*, 2011.

[9] E. Dellis, A. Vlachou, I. Vladimirskiy, B. Seeger, and Y. Theodoridis. Constrained subspace skyline computation. In *CIKM*, pages 415–424. ACM, 2006.

[10] P. Godfrey, R. Shipley, and J. Gryz. Algorithms and analyses for maximal vector computation. *VLDB J.*, 16(1):5–28, 2007. Also appeared in VLDB'05.

[11] M. Goncalves and M.-E. Vidal. Top-k skyline: A unified approach. In *OTM Workshops*, 2005.

[12] V. Hristidis, N. Koudas, and Y. Papakonstantinou. Prefer: A system for the efficient execution of multi-parametric ranked queries. In *SIGMOD*, 2001.

[13] I. F. Ilyas, G. Beskales, and M. A. Soliman. A survey of top-*k* query processing techniques in relational database systems. *ACM Comput. Surv.*, 40(4), 2008.

[14] M. Kontaki, A. N. Papadopoulos, and Y. Manolopoulos. Continuous top-k dominating queries in subspaces. In *Panhellenic Conference on Informatics*, 2008.

[15] H. T. Kung, F. Luccio, and F. P. Preparata. On finding the maxima of a set of vectors. *J. ACM*, 22(4):469–476, 1975. Also in FOCS'74.

[16] J. Lee, G. won You, and S. won Hwang. Personalized top-k skyline queries in high-dimensional space. *Inf. Syst.*, 34(1):45–61, 2009.

[17] X. Lian and L. C. 0002. Top-k dominating queries in uncertain databases. In *EDBT*, 2009.

[18] X. Lin, Y. Yuan, Q. Zhang, and Y. Zhang. Selecting stars: The k most representative skyline operator. In *ICDE*, 2007.

[19] M. Magnani and I. Assent. From stars to galaxies: skyline queries on aggregate data. In *EDBT*. ACM, 2013.

[20] M. Magnani, I. Assent, and M. L. Mortensen. Taking the big picture: representative skylines based on significance and diversity. *The VLDB Journal*, pages 1–21, 2014.

[21] J. Matousek. Computing dominances in e^n. *Inf. Process. Lett.*, 38(5):277–278, 1991.

[22] D. Nanongkai, A. Lall, A. Das Sarma, and K. Makino. Interactive regret minimization. In *SIGMOD*, pages 109–120. ACM, 2012.

[23] 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.

[24] A. N. Papadopoulos, A. Lyritsis, A. Nanopoulos, and Y. Manolopoulos. Domination mining and querying. In *DaWaK*, 2007.

[25] P. Peng and R. C.-W. Wong. Geometry approach for k-regret query. In *ICDE*, pages 772–783. IEEE, 2014.

[26] Y. Tao, L. Ding, X. Lin, and J. Pei. Distance-based representative skyline. In *ICDE*, 2009.

[27] P. Tsaparas, T. Palpanas, Y. Kotidis, N. Koudas, and D. Srivastava. Ranked join indices. In *ICDE*, 2003.

[28] T. Xia, D. Zhang, and Y. Tao. On skylining with flexible dominance relation. In *ICDE*, 2008.

[29] M. L. Yiu and N. Mamoulis. Multi-dimensional top-dominating queries. *VLDB J.*, 18(3):695–718, 2009. Also VLDB'07.

[30] W. Zhang, X. Lin, Y. Zhang, J. Pei, and W. Wang. Threshold-based Probabilistic Top-k Dominating Queries. *VLDB J.*, 19(2):283–305, 2009.