Representative Skylines using Threshold-based Preference Distributions

Atish Das Sarma [#1], Ashwin Lall [#2], Danupon Nanongkai [#3], Richard J. Lipton [#4], Jim Xu [#5]

[#] *College of Computing, Georgia Institute of Technology*
266 Ferst Dr., Atlanta, GA-30332, USA

[1] dassarma@google.com (current affiliation is Google Research)
[2] lalla@denison.edu (current affiliation is Denison University)
[3] danupon@cc.gatech.edu
[4] rjl@cc.gatech.edu
[5] jx@cc.gatech.edu

***Abstract*— The study of skylines and their variants has received considerable attention in recent years. Skylines are essentially sets of most interesting (undominated) tuples in a database. However, since the skyline is often very large, much research effort has been devoted to identifying a smaller subset of (say k) "representative skyline" points. Several different definitions of representative skylines have been considered. Most of these formulations are intuitive in that they try to achieve some kind of clustering "spread" over the entire skyline, with k points. In this work, we take a more principled approach in defining the representative skyline objective. One of our main contributions is to formulate the problem of displaying k representative skyline points such that the probability that a random user would click on one of them is maximized.**

Two major research questions arise naturally from this formulation. First, how does one mathematically model the likelihood with which a user is interested in and will "click" on a certain tuple? Second, how does one negotiate the absence of the knowledge of an explicit set of target users; in particular what do we mean by "a random user"? To answer the first question, we model users based on a novel formulation of threshold preferences which we will motivate further in the paper. To answer the second question, we assume a probability distribution of users instead of a fixed set of users. While this makes the problem harder, it lends more mathematical structures that can be exploited as well, as one can now work with probabilities of thresholds and handle cumulative density functions.

On the theoretical front, our objective is NP-hard. For the case of a finite set of users with known thresholds, we present a simple greedy algorithm that attains an approximation ratio of $(1 - 1/e)$ of the optimal. For the case of user distributions, we show that a careful yet similar greedy algorithm achieves the same approximation ratio. Unfortunately, it turns out that this algorithm is rather involved and computationally expensive. So we present a threshold sampling based algorithm that is more computationally affordable and, for any fixed $\epsilon > 0$, has an approximation ratio of $(1 - 1/e - \epsilon)$. We perform experiments on both real and synthetic data to show that our algorithm significantly outperforms previously proposed approaches.

I. Introduction

Extracting a few tuples from the database to support multi-criteria decision making is an important functionality for database systems. This is required in many application domains where the end-users are more interested in the most important query answers in the potentially huge answer space. Skyline queries [1], [2] are well-studied tools in such settings. The study of computing skylines and related quantities has received considerable attention in recent years.

Consider the following example. A tourist is searching for hotels that are both cheap and of high quality, on a hotel reservation system. Although hotels that have higher quality tend to be more expensive, there could be instances where a hotel A is more expensive and of inferior quality than a hotel B. Clearly, in this scenario, hotel A is *dominated* by hotel B, and A should not be shown to the tourist. The set of *undominated* tuples forms the skyline set, and clearly only this set of hotels is of interest to the tourist.

In many real-life examples (e.g., cars, laptops, hotels etc.), however, all items tend towards the skyline because they would otherwise not be "economically viable". However, the size of the skyline is usually too large. Theoretically, the size of the skyline could be arbitrarily large even when there are only two criteria involved (such as price and quality in this case). Moreover, when there are many criteria involved, the skyline size grows exponentially in the number of criteria even when the values are randomly generated [3], [4].

Due to the often gigantic size of a skyline, significant research effort has been devoted to identifying a much smaller subset of (say k) most interesting skyline points, known as a *representative skyline*. This is motivated by examples such as E-commerce websites, where the number of skyline merchandises may be large, but one can display only a small number of them to the user. Specifically, in the hotel reservation system example, displaying all the skyline hotels to the tourist would be impractical, and overwhelming. What subset of these skyline tuples should the hotel reservation system display, in the absence of much information about the tourist user? In particular, the website wishes to display results such that the user (or equivalently a random user) is most likely to be interested in (and thereby click on) at least one of them.

Several approaches have been suggested for displaying a subset of size k of all the skyline tuples. In each of these works, a different definition of what is the *best* subset of size k has been adopted. One [5] defines a notion of clustering on the skyline points and displays the cluster centers, while

another assigns the quality of a subset based on the number of points they dominate [6]. Some other approaches [7], [8] relax the notion of domination so that the number of *relaxed skylines* is reduced, while another [9] assumes users have linear utility functions and displays results to minimize the minimum utility over all users. We will discuss more about these approaches, their merits and drawbacks, and a few other approaches in the related work section. While all of these approaches (and formulations/definitions of representatives skylines therein) seem intuitively reasonable, which of these should one adopt? They all define the objective to attain some kind of a *spread* over all skyline tuples. Is there one *canonical* objective?

In this paper, we undertake a more principled approach in defining the quality of a subset of k tuples. Our goal is to display k skyline points such that the probability that a random user would click on (i.e. like) one of the displayed k results is maximized. To achieve this goal, we define the quality of a subset of k tuples, our objective function to be maximized, to be exactly the probability with which a random user, from among a set of users (or a distribution over types of users), will click on one of the tuples. This definition and formulation is indeed one of the main contributions of this paper.

Two questions arise to fully describe the definition based on maximizing probability of a random user liking one of the displayed k results. First, how does one model the probability with which a user click on a result tuple? Second, how does one negotiate the absence of the knowledge of an explicit set of users; in particular what do we mean by a random user? We model users based on threshold preferences. This is a very simplistic assumption, yet perhaps not far from reality. It also turns out that threshold preference functions fully *characterize* skylines in the sense that there is a direct correspondence between threshold functions and skylines. A specific user exploring hotels may look at any hotel that is at least 4 stars and costs at most 150 dollars a night, and not look at anything that does not satisfy these thresholds. To answer the second question, we assume a distribution of users instead of a fixed set of them. While this makes the problem harder, it lends more structure as well, as one can now work with probabilities of thresholds and handle cumulative density functions. Furthermore, distributions become crucial in the absence of explicit knowledge of a fixed set of users. The threshold based model, while simplistic, may be learned from a variety of sources such as market surveys or user polls even in the absence of rich query logs. Compared to previous approaches, our approach assumes preference distributions and obtains better representative skylines.

On the theoretical front, this problem is NP-hard (for ≥ 3 dimensions). For a finite set of users, there is a simple greedy algorithm that attains an approximation ratio of $(1-1/e)$ of the optimal solution. For the case of user distributions, we show how to perform a similar greedy algorithm carefully to achieve the same approximation ratio. Unfortunately, this algorithm is rather involved and computationally expensive. We therefore present a threshold sampling based algorithm that, for any fixed $\epsilon > 0$, the approximation ratio guaranteed is $(1-1/e-\epsilon)$; here the sampling complexity depends on ϵ.

Additionally, we perform experiments with the sampling based algorithm on both real and synthetic data. For synthetic data, we generate skyline points in an anti-correlated manner (as is standard in literature), and generate thresholds as Gaussians, i.e. Normal distributions, independently along each attribute. We compare the performance of our algorithm with previous approaches for varying number of dimensions, points, and parameters of the distribution. Our algorithm outperforms other approaches on our metric, which is arguably expected since other techniques are oblivious to the threshold distributions. However, this does suggest that if one can access or learn user preferences, they should be exploited for higher throughput in terms of user clicks. Similar results are seen on the real data where the threshold distributions are specified by the data sets. We also measure the distribution of values of each attribute and find that they indeed behave close to Gaussians. Here we summarize our results.

Our Contributions.

- We define a representative skyline objective based on maximizing the probability of a random user clicking on one of the displayed results. In addition to this definition, the novelty lies in (i) modeling users based on threshold preferences, and (ii) defining a *random* user using preference distributions. Our objective has all the desirable properties of representative skylines, such as scale invariance and stability. We also show how to generalize beyond threshold preference distributions.
- We prove that optimizing this objective is NP-hard even for three dimensions. However, for a finite set of specified users, a simple greedy algorithm obtains a $(1 - 1/e)$-approximation. For the more interesting case of unknown users modeled as preference distributions, we show how a greedy algorithm can be simulated to achieve the same $(1-1/e)$-approximation. This algorithm being computationally expensive, we present an efficient sampling-based algorithm that guarantees a $(1 - 1/e - \epsilon)$-approximation ratio for any $0 < \epsilon < 1$.
- We perform experiments on real and synthetic data on the efficient sampling-based algorithm. Our sampling algorithm outperforms all previous representative skyline algorithms. For the real data, we further verify that the distributions measured from the attribute values are unimodal and bell-shaped (providing credence to our assumption of Gaussians).

Overview. We introduce notation and the problem definition in Section II. The theoretical results for the easier finite user base is presented in Section III. The theoretical results for the more interesting case of continuous user distributions are then presented in Section IV. Our experimental results are detailed in Section V. We mention related work in Section VI. Finally we mention a few directions for future research in Section VII.

II. Problem Formulation

A. Definitions and Axioms

We begin by defining skylines precisely. Given a set D of d-dimensional points, a point $x \in D$ is said to dominate a point $y \in D$, if $x(i)$ (the i-th coordinate of x) is no smaller than $y(i)$ for every $1 \leq i \leq d$; further $x(i) > y(i)$ for at least some i. A point is said to be in the skyline if it is not dominated by any other point in the database. The set of all points that are not dominated by any other point is said to be the skyline set. In this sense, the hotel example is somewhat misleading as a smaller price is better than a larger price. Such "semantics" problems, however, can easily be fixed by inverting the values. Therefore, throughout this paper we assume that users prefer larger values on any attribute.

Various different definitions have been considered in the literature, each having its own merits. However, all of them are ad hoc in the sense that they all define an objective that is *intuitive* but not necessarily *principled*. In this paper, we would like to define an objective that captures the goal of database representation, which we believe is to induce a click, by whichever user that views the representative set. So we would like to define an objective that *maximizes* the probably that a user (or a randomly sampled user - in the absence of additional information) would click on at least one of the displayed results. This event would occur only when (at least) one of the tuples displayed satisfies the users requirements or threshold for *quality*. In order to define such an objective, we first introduce some notation followed by necessary axioms.

Let $q_u(s)$ denote the quality of a tuple s for a user u. We want this to be the probability with which the user would click on the corresponding tuple. Let $q_U(S)$ denote the quality of a set of tuples S for a set of users U. The problem we are considering is representing the best possible set of k tuples, among the skyline tuples to maximize average quality. We formalize the notion of quality with the following axioms (we call these "axioms" as these are constraints we want satisfied):

Axiom 0. $q_u(s) \in [0, 1]$, and $q_u(s)$ is the probability of user u clicking on s when s is shown.

Axiom 1. $q_u(\{s_1, s_2\}) = \max\{q_u(s_1), q_u(s_2)\}$ - The intuition is that a user clicks on only one hotel, the one that look best to the user, after looking at all results displayed. Note that, by this axiom, the quality value of a set is not affected by the order in which the elements in the set are displayed.

Axiom 1 states that the probability the user clicks on one of the items is equal to the probability that the user clicks on the better of the two items. An alternate definition is $q_u(\{s_1, s_2\}) = 1-(1-q_u(s_1))(1-q_u(s_2))$, i.e., the probability that the user clicks on one of them. The intuition for the former is that the user first finds the best hotel/result, and then decides to either click on it, or not. The intuition for the latter is that the user looks at the hotels/results one by one, and decides whether or not to click on each of them independently. However, since we will only consider the case when $q_u(s)$ is either 0 or 1 later on, these two definitions are equivalent.

Axiom 2. $q_U(S) = \frac{1}{|U|} \sum_{u \in U} q_u(S)$ - Average quality of the displayed set S, averaged over all users.

Towards a problem definition. Suppose the set of users U is finite and their quality vectors are fully specified. Then we have the following problem.

Problem definition for finite case. Given D, and a set of users U, find $S \subseteq D$, $|S| = k$, such that $\max_S q_U(S)$.

Notice that if $q_u(s) \in \{0, 1\}$, then $q_U(S)$ is exactly the fraction of all $|U|$ users that would click on at least one of S.

Now consider the case when there is no knowledge of a finite set of users. In such a situation, one could assume, or perhaps have knowledge of, a distribution over quality vectors $q_u()$. For example, one could assume that the quality vectors $q_u()$ are drawn from random $[0, 1]$ vectors (the dimension of the vectors being $|D|$). Suppose that such a distribution over quality vectors is specified. Then one can define the problem as maximizing $E(q_U(S))$ where U is the user distribution (i.e. distribution of quality vectors) and $E()$ is the expectation taken over this distribution.

To fully specify the problem formulation, we need to understand how users interpret qualities of displayed results. In particular, given a database entry s with attribute values, how does a user u determine $q_u(s)$. We now specify this aspect of our formulation before returning to formalize the problem definition for user distributions.

B. Modeling Users

Consider a hotel website that displays ten results. How does a user looking at a set of result hotels decide to click on a hotel or not? Does the user try to evaluate the exact quality of each displayed result? Does the user look for one hotel that meets his requirements, and then click on it for more statistics? We believe that a user decides to click on a hotel link if it meets the user's hard constraints. For example, a user may be willing to pay up to $150 a night and want something that is rated at least 4 stars. Assuming these constraints are met, the user is likely to click on the hotel and look for additional details (such as proximity to downtown, breakfast options etc.). Similarly, a user looking to buy a car may be interested in all cars that have a fuel efficiency of at least 20 miles per gallon, and the engine of which delivers at least 6 horse powers. In other words, users set mental thresholds and if these minimum thresholds are not met, the user does not even bother to explore the item. We model users based on such thresholds.

Fixed Thresholds. Any user u_i is modeled as a set of thresholds along each dimension. If each threshold is met for a tuple s, then the user is satisfied enough to click on s. Therefore, u_i is represented by a set of thresholds $(u_i^1, u_i^2, \ldots, u_i^d)$. Further, a user is either satisfied with a tuple enough to click on it, or not. Therefore, $q_u(s)$ is either 0 or 1, for every u and every $s = \{s^1, s^2, \ldots, s^d\}$. $q_u(s) = 1$ if $u^j \leq s^j$ for each $j \in \{1, 2, \ldots, d\}$, and $q_u(s) = 0$ otherwise. Intuitively, for a user, results below the threshold are not explored further and other results are good enough to elicit a click. An example of such a discrete choice model in literature is [10].

While already interesting, the threshold assumption can in fact be generalized. Specifically, one can model situations where each user operates on a set of thresholds: If the highest threshold for u is met by s, then the user is certainly interested, i.e., $q_u(s) = 1$. If the lowest threshold is not met, then $q_u(s) = 0$. If one of the intermediate thresholds is met, then $q_u(s) \in (0, 1)$, depending on which threshold is met. Notice that this can be reduced to a setting where the number of users is increased (some repeated, to obtain the weights) and each user has just one threshold. We omit the precise reduction for brevity. Therefore, without loss of generality, for the rest of the paper we assume that each user has one fixed threshold leading to $0-1$ preferences (that is, q value is 1 if threshold is met and 0 otherwise). We begin with an interesting observation that characterizes skylines in terms of threshold functions. We omit the proof since it is straightforward.

Observation. Threshold functions captures skylines precisely. That is, for every skyline point x, there is a threshold function f_x such that f_x is satisfied (or 1) for x and unsatisfied (or 0) for every other skyline point.

Threshold Distributions. Since one is not usually aware of an explicit set of n users, we study the case when only a distribution over the users' thresholds is known. Let the distribution of thresholds be given by the probability density function $f()$, and the cumulative density function $F()$. So $F(t_1, t_2, \ldots, t_d)$ denotes the probability that any user u has thresholds $u^j \leq t_j$ for $1 \leq j \leq d$. Further, $F(x, x, \ldots, x) \to 1$ as $x \to \infty$. We assume that all users are independent and identically distributed according to $F()$. Notice that given a set of points, computing F over thresholds satisfied by these points may be difficult. We assume that F can take as input d intervals such as $[t_i^l, t_i^u]$ for $1 \leq i \leq d$, and easily compute $F([t_1^l, t_1^u], [t_2^l, t_2^u] \ldots, [t_d^l, t_d^u])$, the probability that a random user has thresholds in these intervals. This is the only assumption we make. Our results hold regardless of whether the distributions along different dimensions are dependent or independent.

We now define $q_F(S)$ as the probability that $q_u(S) = 1$ when u (or u's thresholds) are sampled from the distribution $f()$. Notice that $q_F(S)$ is like the expected fraction of users that will click on one of S when the users' thresholds are from the distribution specified by $f()$. In particular, $q_F(S) = E(q_f(S))$ where this expectation is taken over a sample from the distribution with probability density function f.

C. Problem Definition - Maximizing Clicks

Problem Definition for Threshold Distributions. Given D, k and the cumulative density function for threshold distributions denoted as F, find S to maximize $q_F(S) = E_f(q_f(S))$, where f is a sample from this distribution, and $|S| = k$. Recall, $q_f(S) = 1$ if (at least) one point in S satisfies f.

Notice that $q_F(S)$ is like $q_U(S)$ in the limit where U tends to ∞ and each user is I.I.D. An example of F could be given by independent Normal distributions along each dimension. For example, in the case of hotels, maybe only 10 percent of the user base is willing to pay more than 250 dollars a night and only 10 percent of them can pay only 50 dollars a night. A large fraction of the people have a threshold for the hotel price between 100 and 150 dollars. Similarly for star ratings, one might know that 80% of all users have a threshold on hotel rating between 3-stars and 5-stars, and 10% each below and above 3-stars and 5-stars respectively. These thresholds may be correlated (as probably is the case for hotels' star ratings and prices) or may be independent (perhaps for cars' sporty looks and miles per gallon).

Properties. Two desirable properties that have been investigated in the past for representative skyline definitions are *stability*, and *scale invariance*. The notion of stability asks that adding non-skyline points should not change the objective or the optimal solution. This is important as otherwise hotels could manipulate their probability of being in the representative skyline set by just adding non-skyline (fake) hotels to the database. Therefore it is desirable that a representative skyline objective definition be oblivious to such perturbations.

Another important consideration for a good definition is that it be scale invariant. This means that rescaling the attribute values along any dimension should not alter the representative skyline set (as long as the rescaling factors are positive). Notice that the skyline set itself is never altered by such rescaling. For example, if one rescales the star values of hotels such that the stars are now in the range of one star to ten stars, instead of one star to five stars, this should not alter the solution. In our case, this of course means also rescaling the thresholds. It is fairly straightforward to see that our definitions of $q_F(S)$ and $q_U(S)$ are both scale invariant and stable.

Additional Notation. The number of dimensions/attributes in the database is denoted by d. The total number of points in the database is denoted by M and the number of skyline points is given by m. n denotes interchangeably the number of thresholds, or the number of users. For threshold distributions, we later on use μ and σ to denote the mean and the standard deviation, and F to denote the cumulative density function over d attributes.

III. n EXPLICIT THRESHOLD FUNCTIONS

In this section we consider the simple case when there are exactly n users U. Each user in U is represented by u_i (for $1 \leq i \leq n$), where u_i is a threshold vector. Recall that a skyline point p satisfies the threshold vector u_i if p is at least as much as u_i in each of the d dimensions. Recall that $q_U(p)$ is the fraction of all thresholds in U that are satisfied by p. The goal is to find a subset of size k of all the skylines points in D so that the number of thresholds satisfied is maximized. Throughout this section, we assume that the input itself contains only the skyline points (since we focus on the problem of isolating representative sets from the skylines, rather than of computing skylines from the entire database). This section states two main results. First that maximizing $q_U(S)$ for $S \subseteq D$, $|S| = k$ is NP-hard even for $d = 3$, and second that there is a simple greedy algorithm that approximates the optimal $q_U(S)$ to within a constant factor.

Theorem 1 ([6]): Finding $S \subseteq D$, $|S| = k$ to maximize $q_U(S)$ is NP-hard even for $d = 3$ dimensions.

NP-hardness for large d follows easily by reduction from the Max Coverage problem; $d = 3$ requires a bit more work. For brevity we omit the proof here; [6] prove this in a slightly different context. We now present SIMPLE-GREEDY to attain a constant approximation for $q_U(S)$. A similar algorithm was suggested in [6] in a different context.

Algorithm 1 SIMPLE-GREEDY(D, k, ϵ)

Input: A set D of tuples, a desired output size k and a set of input thresholds $U = \{u_1, u_2, \ldots, u_n\}$

Output: A set $\mathsf{ALG} \subseteq D$ of k tuples such that $q_U(\mathsf{ALG}) \geq (1 - 1/e) q_U(\mathsf{OPT})$ (for all $\mathsf{OPT} \subseteq D$, $|\mathsf{OPT}| = k$).

1: Let $\mathsf{ALG} = \emptyset$.
2: **for** $i = 1$ to k **do**
3: Let p be a tuple in D such that $q_U(p)$ is maximum.
4: Add p to ALG.
5: Delete every threshold u_i in U such that u_i is satisfied by p.
6: **end for**

Theorem 2: Algorithm SIMPLE-GREEDY achieves a $(1 - 1/e)$-approximation to the optimal.

This can be proved via a reduction to the Max-Coverage problem. However, we present a different proof here for completeness and as we need these definitions in the next section. The proof uses work of Nemhauser, Wolsey and Fisher [11], [12] who show that such a greedy algorithm guarantees a $(1 - 1/e)$-approximation for any monotonically non-decreasing sub-modular maximization objective.

Monontonically non-decreasing function g**.** Function g is said to be monotonically non-decreasing if for any sets S and T such that $S \subseteq T$, $g(S) \leq g(T)$.

Submodular function g**.** Function g is said to be submodular if for any sets S and T and any element p such that $S \subseteq T$ and $p \notin T$, $g(S \cup \{p\}) - g(S) \geq g(T \cup \{p\}) - g(T)$.

In words, non-decreasing means that the function value only improves when a larger superset is considered; submodularity essentially refers to decreasing marginal utility; i.e. adding one more point to a smaller set helps more (increases the function value more) than it would on adding to a larger (super)set.

Proof: [of Theorem 2]

The proof follows now using [11], [12] and making the simple observations that our objective function $g = q_U(.)$ is monotonically non-decreasing, and submodular.

The non-decreasing property of $q_U(S)$ follows immediately as adding more points to S can only increase the number of thresholds satisfied in U.

To observe that $q_U()$ is sub-modular, consider two sets of points S and T such that $S \subseteq T$ and point $p \notin T$. Now, $q_U(S \cup \{p\}) - q_U(S)$ is $\frac{1}{n}$ times the number of thresholds that are satisfied by p but not satisfied by S. Similarly, $q_U(T \cup \{p\}) - q_U(T)$ is $\frac{1}{n}$ times the number of thresholds that are satisfied by p but not satisfied by T. Since T is a superset of S, all thresholds satisfied by S are also satisfied by T. It follows that $q_U(S \cup \{p\}) - q_U(S) \geq q_U(T \cup \{p\}) - q_U(T)$. ■

Algorithm 2 GREEDY-ON-DISTRIBUTION(D, k, F)

Input: A set of d-dimensional (skyline) points $D = \{p^1, p^2, ..., p^m\}$, an integer k (the desired output size), and the distribution of threshold functions specified by the CDF F.

Output: A subset of D of size k, denoted by S.

1: Set $S = \{\}$
2: **for** i=1 to k **do**
3: p^* = $\arg\max_{p \in D}$DENSITY-INCREMENT-COVERED(p, S, F)
4: $S = S \cup p^*$.
5: **end for**
6: **return** S

IV. DISTRIBUTION ON THRESHOLD FUNCTIONS

We now consider the more interesting case where there is no explicit set of fixed n users. Rather, users are unknown and modeled as distributions over threshold preference functions.

A. Greedy Algorithm directly on Threshold Distribution

Recall the setting that we have a distribution on threshold functions. This means that given $x_1, x_2, \ldots, x_d$, we know the fraction of all users whose thresholds are less than this. Specifically, we have easy access to the cumulative density function (CDF). Further, we assume that the cumulative density function is easy to compute given ranges along each dimension: Given ranges $[a_1, b_1], [a_2, b_2], \ldots [a_d, b_d]$, we can compute the fraction of thresholds (or fractions of users with thresholds) such that the thresholds $(t_1, t_2, \ldots, t_d)$ satisfy $a_i \leq t_i \leq b_i$ for all $1 \leq i \leq d$. This is a reasonable assumption. For example for thresholds independent along each dimension: Even if only the probability density function along each attribute is known, the density would just be the product of d independent definite integrals.

Notice, this does not assume that given any region bounded by points, it is immediate how to compute the density under this region. This is in fact the difficult aspect of performing a greedy algorithm. How do we know the fraction of thresholds that are already covered by a chosen set of points? It is not easy to compute the CDF of the threshold distribution of the region dominated by these points.

We present GREEDY-ON-DISTRIBUTION, a greedy approach on such threshold distributions. We introduce some notation for convenience. We use $F(I_1, I_2, \ldots, I_d)$ where I_i are intervals (such as $[a_i, b_i]$) and F is the cumulative density of the thresholds that satisfy these intervals (analogous to fraction of thresholds if they were finite).

The main intuition of Algorithm GREEDY-ON-DISTRIBUTION is as follows. In the first step we pick the skyline point that satisfies the maximum density of thresholds. In the second step, we pick the skyline point

Algorithm 3 DENSITY-INCREMENT-COVERED(l, S, F)

1: **return** DENSITY-COVERED($|S|+1$, $S \cup \{p\}$, F, 0, $\{\}$) - DENSITY-COVERED($|S|$, S, F, 0, $\{\}$)

Algorithm 4 DENSITY-COVERED(k', R, F, $d'(0 \leq d' \leq d)$, $\mathbf{I} = \{I_1, I_2, \ldots, I_{d'}\}$)

Input: A set R of k' d-dimensional (skyline) points $R = \{p^1, p^2, ..., p^{k'}\}$ (here $p^j = (p_1^j, p_2^j, ..., p_d^j)$), the distribution of threshold functions specified by the CDF F, and a set of d' intervals $\mathbf{I} = \{I_1, I_2, \ldots, I_{d'}\}$, where each I_i is an interval of the form $(-\infty, b]$ or $[a, b]$.

Output: Density F in the region that contains all points $t = (t_1, t_2, \ldots, t_d)$ such that for all $j \leq d'$, t_j is in the interval I_j, and for $j > d'$ t_j is such that, we have the points $(0, 0, ..., 0, t_{d'+1}, t_{d'+2}, \ldots, t_d)$ dominated by $(\infty, \infty, ..., \infty, p^j_{d'+1}, ..., p^j_d)$ for at least one j.

1: **if** $d' = d$ **then**
2: **return** INTERVAL-DENSITY-COVERED(**I**, F)
3: **end if**
4: Lexicographically sort the k' points in increasing order, and compute the skyline set of $(\infty, \infty, ..., \infty, p^j_{d'+1}, ..., p^j_d)$.
5: Let the points in increasing lexicographical order be $p^1, p^2, \ldots, p^{k'}$, without loss of generality.
6: $sum = 0$.
7: $d' = d' + 1$.
8: Append interval $I_{d'} = (-\infty, p^1_{d'}]$ to **I**, and denote this set of intervals by **J**.
9: $sum = sum +$ DENSITY-COVERED(k', R, F, d', **J**)
10: **for** $s = 2$ to $s = k'$ **do**
11: Append interval $I_{d'} = [p^{s-1}, p^s]$ to **I**, and denote this set of intervals by **J**.
12: $sum = sum +$DENSITY-COVERED(k', R, F, d', **J**)
13: **end for**
14: **return** sum

that satisfies the maximum density of thresholds, among thresholds that have not already been satisfied by the first point. So on at each step, the point picked satisfies the largest density of thresholds in the range(s) of thresholds that have not already been covered.

The difficulty arises in computing this quantity for any point. Specifically, it reduces to being able to compute the density of thresholds covered by a set of points - one can then take the difference of two sets of points to compute the *marginal* density covered by a point in consideration. This reduction is straightforward, as specified in Algorithm DENSITY-INCREMENT-COVERED. The Algorithm DENSITY-COVERED is a recursive algorithm carefully crafted to compute the density covered by a set of points. For the sake of recursion, the algorithm actually computes the density covered by a set of points R as well as a set of intervals **I**. If **I** is empty, this algorithm returns the density of thresholds satisfied by points in R (by recursive calls to itself). Otherwise, **I** may contain up to d intervals, one corresponding to each dimension. If in fact **I** does contain d intervals, then DENSITY-COVERED ignores R and returns the density of thresholds that satisfy these d intervals along the corresponding dimensions. Notice that this is easy as in Algorithm INTERVAL-DENSITY-COVERED and does not require recursive calls.

Algorithm 5 INTERVAL-DENSITY-COVERED(**I**, F)

Input: A set of d intervals, $I_1, I_2, \ldots, I_d$, and the distribution of threshold functions specified by the CDF F.

Output: $F(I_1, I_2, \ldots, I_d)$.

1: **return** $F(I_1, I_2, \ldots, I_d)$
{F}or Example, in case of independent, this is product of d different CDFs over these d intervals.

Algorithm DENSITY-COVERED starts with $|R|$ points and zero intervals in **I**, and slowly translates this to contain sets of d intervals that reflect ranges determined by points in R. The region covered by R may however correspond to several different sets of intervals (since it may not be just a d-dimensional cuboid). Therefore, the algorithm is designed to slowly break this region into simple cuboids, and then compute the density in each of these. The points in R are sorted lexicographically. This allows us to know the lowest (or highest) along say the first dimension. This, then allows us to compute the cuboid that stretches to this lowest value along the first dimension (by essentially projecting other points to this lowest value). The algorithm queries itself for such projected points (and this interval). Once this density is computed (recursively), this quantity is set aside to be added to the final solution. To compute the remaining density, the points are then projected on to the second lowest value along the first dimension, and the process is repeated. The formal algorithm specifies exactly how this is done. We now proceed to prove the approximation for GREEDY-ON-DISTRIBUTION.

Theorem 3: For S given by GREEDY-ON-DISTRIBUTION, $q_F(S) \geq (1-1/e) q_F(\mathsf{OPT})$. ($\mathsf{OPT} = \arg\max_{T,|T|=k} q_F(T)$)

Proof: The difficulty lies in performing the greedy algorithm carefully; since F is difficult to compute over arbitrary regions, the greedy step had to be done by computing F by systematically and recursively reducing the covered regions to intervals. There are three main claims to proving the approximation ratio. Each is simple. The first states that we are indeed performing the greedy algorithm and the next two state that $q_F(S)$ is a monotonically non-decreasing, and submodular objective function.

GREEDY-ON-DISTRIBUTION performs the greedy choice at every step: The algorithm invokes DENSITY-INCREMENT-COVERED for each of the skyline points on each of the k rounds. Consider any of the k rounds. DENSITY-INCREMENT-COVERED computes the incremental benefit of adding a point to the solution set, where the benefit is measured as the density of the thresholds satisfied (that were not already satisfied so far). The maximal point is then picked. Further,

DENSITY-INCREMENT-COVERED computes the incremental benefits exactly by invoking DENSITY-COVERED which recursively computes the area covered by a set of points. This is done by breaking the region covered into sets of intervals and then the density F is easily computed by querying INTERVAL-DENSITY-COVERED. Therefore, the choice made at each of the k steps is the *best possible* for that specific step (i.e. optimal choice assuming this was the last point to be picked). Therefore GREEDY-ON-DISTRIBUTION achieves the greedy choice at every step.

The objective $q_F(S)$ is monotonically non-decreasing: Notice that $q_F(S)$ is the density sum of F over the region satisfied (or covered) by points in S. Adding a point to S only increases the overall region satisfied by points; therefore $q_F(S) \leq q_F(T)$ if $S \subseteq T$.

The objective $q_F(S)$ is submodular: Observing that $q_F(S)$ is again straightforward. Notice that $q_F(S)$ is exactly the density under the region satisfied by S in the d-dimensional space; so we only need to prove that the volume/region satisfied by a set of points S is submodular. Consider sets of points $S \subseteq T$; the region satisfied by T is a superset of the region satisfied by S. Now look at a point p and the region satisfied by it. If p is added to T, the region satisfied by $T \cup \{p\}$ increases by the volume that got additionally satisfied by p (but was not satisfied by any point in T). Similarly the volume in the region satisfied by $S \cup \{p\}$ minus region satisfied by S contains exactly all points in the space that are dominated by p but not dominated by any point in S. Since $S \subseteq T$, it follows that $q_F(S \cup \{p\}) - q_F(S) \geq q_F(T \cup \{p\}) - q_F(T)$.

The proof now follows using [11], [12] which gives a $(1 - 1/e)$-approximation for the greedy algorithm for an objective that is non-decreasing and sub-modular. ■

1) Time Complexity: The algorithm GREEDY-ON-DISTRIBUTION turns out to be very expensive due to the recursive calls. While performing the i-th greedy choice (for $i \leq k$), we need to compare $O(m)$ skyline points. For each of them, we need to compute the marginal benefit of adding it to the set, as done by DENSITY-INCREMENT-COVERED. Therefore, DENSITY-INCREMENT-COVERED is invoked mk times. This in turn calls DENSITY-COVERED which makes recursive calls to itself up to depth d before terminating with INTERVAL-DENSITY-COVERED. The key trick now is that in INTERVAL-DENSITY-COVERED, all the points are already sorted in increasing lexicographic order (and so this does not have to be done in each call). However, DENSITY-COVERED still needs to call itself $k' - 1$ number of times where k' was the number of points - this can be as large as m; one call for projecting the points down to the t-th highest value along the corresponding dimension (for $t \leq k'$). This therefore yields a time complexity of roughly $(km) * (m-1)^d$ which is $O(km^d)$. The good thing is that d is in the exponent and not k - so this is still manageable for small d (while k typically can be around 10 or 20). However, even for $d \geq 4$ (which is often the case), this becomes hugely expensive. In the following subsection, we consider a more efficient approach.

B. Greedy Algorithm by Sampling from Threshold Distribution

We have show that given explicit n users, there is a very efficient simple algorithm. Further, if users are not specified, and we can only assume a distribution over users, then the greedy algorithm becomes nontrivial to simulate and computationally very expensive. This suggests a natural and more efficient approach - sample users (threshold preferences) from the distribution, and pretend that the samples represent a set of explicit users. Then perform the simple greedy algorithm. With sufficient samples, this seems like a reasonable algorithm that should do well in practice. Here we actually formalize this algorithm and prove a concrete bound.

More generally, on can consider the case when the distribution F is unknown but some samples of threshold functions drawn from F are available. Let $F' = \{f_1, f_2, \ldots, f_n\}$ be these samples. We show that one can obtain an approximate to $q_F()$ that is arbitrarily close to $(1 - 1/e)$, depending on n.

To state the result formally, we need some more notation. Let OPT be the optimal solution. Recall that for any set S, $q_F(S)$ denotes the quality of S on distribution F, i.e., the probability that a threshold function drawn from the distribution with CDF F will be satisfied by a tuple in S. Also recall that for any set U of threshold functions, $q_U(S)$ denotes the quality of S on U, i.e., the number of functions in U that are satisfied by some tuples in S. We are particularly interested in $q_{F'}$. Also recall that m is the number of skyline tuples in D and k is the number of desired output tuples.

Our result is that for any $0 < \epsilon < 1$, one can get a $(1-\epsilon)(1-1/e)$ approximation guarantee with high probability using the number of sampled threshold functions proportional to k, $\ln m$, $1/\epsilon$, and $1/q_F(\mathsf{OPT})$, as in the following theorem.

Theorem 4: For any $0 < \epsilon < 1$, there is an algorithm that uses $n = \left\lceil \frac{3(k+1)\ln m}{\epsilon(1-1/e)q_F(\mathsf{OPT})} \right\rceil + 1$ sample points and, with probability at least $1 - 1/m$, outputs a set of tuples of size k, denoted by ALG, such that $q_F(\mathsf{ALG}) > (1-\epsilon)(1-1/e)q_F(\mathsf{OPT})$.

We now present algorithm SAMPLING-GREEDY and prove this theorem. The algorithm runs as follows: First, sample a set of n threshold functions, denoted by $F' = \{f_1, f_2, \ldots, f_n\}$. Next, run the greedy algorithm on F', i.e., in each step, pick a tuple that satisfies the most number of unsatisfied thresholds in F'.

We now analyze SAMPLING-GREEDY. The key to prove Theorem 4 is the following lemma which says that any set that does not approximate OPT well on distribution F is unlikely to approximate OPT well on set F' of samples.

Lemma 5: For any set T of size k such that $q_F(T) \leq (1-\epsilon)(1-1/e)q_F(\mathsf{OPT})$, $Pr[q_{F'}(T) \geq (1-1/e)q_{F'}(\mathsf{OPT})] \leq 1/m^{k+1}$ where the probability is over all choices of F'.

Proof: For $i = 1, 2, ..., n$, let X_i be a 0/1 random variable which is 1 if and only if f_i is satisfied by a tuple in T. Similarly, let X_i^* be a 0/1 random variable which is 1 if and

Algorithm 6 SAMPLING-GREEDY(D, k, ϵ)

Input: A set D of tuples, a desired output size k and a desired approximation accuracy $0 < \epsilon < 1$.

Output: A set $\mathsf{ALG} \subseteq D$ of k tuples such that $q_F(\mathsf{ALG}) \geq (1-\epsilon)(1-1/e)q_F(\mathsf{OPT})$ with probability at least $1 - 1/m$.

1: Sample n points from F. Let $F' = \{f_1, f_2, \ldots, f_n\}$ be the set of such points.
2: Let $\mathsf{ALG} = \emptyset$.
3: **for** $i = 1$ to k **do**
4: Let p be a tuple in D such that $q_{F'}(p)$ is maximum.
5: Add p to ALG.
6: Delete every function f_i in F' such that f_i is satisfied by p.
7: **end for**

only if f_i is satisfied by a tuple in OPT. expectation.

$$\begin{aligned} & E[\sum_{i=1}^{n}((1-1/e)X_i^* - X_i)] \\ &= \sum_{i=1}^{n}((1-1/e)E[X_i^*] - E[X_i]) \\ &= \sum_{i=1}^{n}((1-1/e)q_F(\mathsf{OPT}) - q_F(T)) \\ &\geq \epsilon n(1-1/e)q_F(\mathsf{OPT}) \end{aligned}$$

where the last inequality comes from the fact that $q_F(T) \leq (1-\epsilon)(1-1/e)q_F(\mathsf{OPT})$. Therefore,

$$\begin{aligned} & Pr[q_{F'}(T) \geq (1-1/e)q_{F'}(\mathsf{OPT})] \\ &= Pr[\sum_{i=1}^{n}((1-1/e)X_i^* - X_i) \leq 0] \\ &\leq Pr[\sum_{i=1}^{n}((1-1/e)X_i^* - X_i) \\ &\leq (1-\delta)\epsilon n(1-1/e)q_F(\mathsf{OPT})] \\ &\leq e^{-\epsilon n(1-1/e)q_F(\mathsf{OPT})\delta^2/3}, \end{aligned} \quad (1)$$

for any $0 < \delta < 1$, where the last inequality is by Chernoff bound (see, e.g., Mitzenmacher-Upfal [13, Theorem 4.5]). We chose δ to be close enough to one so that $n \geq \frac{3(k+1)\ln m}{\epsilon(1-1/e)q_F(\mathsf{OPT})\delta^2}$. (Such δ exists since $n > \frac{3(k+1)\ln m}{\epsilon(1-1/e)q_F(\mathsf{OPT})}$.) Plugging in the value of n to (1), we have $Pr[q_{F'}(T) \geq (1-1/e)q_{F'}(\mathsf{OPT})] \leq 1/m^{k+1}$ as desired. ■

Proof: [Proof of Theorem 4] By union bound over all sets of size k, it follows that with probability at least $1-1/n$, every set T of size k such that $q_F(T) \leq (1-\epsilon)(1-1/e)q_F(\mathsf{OPT})$ has $q_{F'}(T) < (1-1/e)q_{F'}(\mathsf{OPT})$. Let ALG be the output of the algorithm. Since $q_{F'}(\mathsf{ALG}) \geq (1-1/e)q_{F'}(\mathsf{OPT})$, ALG cannot be one of those sets T and thus $q_F(\mathsf{ALG}) > (1-\epsilon)(1-1/e)q_F(\mathsf{OPT})$ as desired. ■

Notice that we have shown $q_F(\mathsf{ALG}) > (1-\epsilon)(1-1/e)q_F(\mathsf{OPT})$. For a different ϵ, rearranging gives $q_F(\mathsf{ALG}) > (1-1/e-\epsilon)q_F(\mathsf{OPT})$. We now analyze the computational cost of SAMPLING-GREEDY.

1) Time Complexity: We now sketch an $O(|F'||D|) = O(mn) \approx O(\frac{m}{\epsilon})$ time implementation of Algorithm IV-B. We construct an $m \times n$ array A where $A[i][j]$ equals 1 if tuple $p_i \in D$ satisfies function $f_j \in F'$, and 0 otherwise. This array can be constructed in $O(|F'||D|)$ time. We also keep an array N of size m where $N[i]$ is the number of functions currently in F' that tuple p_i in D satisfies.

As the algorithm proceeds, we pick some tuple p_i and we have to delete from F' all functions satisfied by p_i. This is done by "scanning" all elements in row $A[i][*]$ and for any j such that $A[i][j] = 1$, we "nullify" elements in column $A[*][j]$ to 0 and update the numbers in N accordingly. Observe that each cell $A[i][j]$ is "scanned" only once (only when p_i is picked). Also, each cell $A[i][j]$ is "nullified" only once (since nullification of column $A[*][j]$ is done when $A[i][j] = 1$ for some i and after that every element in column $A[*][j]$ will be 0). Therefore, the total running time is $O(|F'||D|)$. We also note that the implementation could be slightly more efficient (but asymptotically the same) by using link lists.

Since this approach of SAMPLING-GREEDY is much faster, and its approximation tends to that of the exact greedy algorithm GREEDY-ON-DISTRIBUTION (as the sampling is increased), we use this for experiments.

V. EXPERIMENTAL EVALUATION

In this section we show that the SAMPLING-GREEDY (in short Greedy) algorithm consistently covers a large fraction of thresholds, outperforming other representative skyline definitions due to its knowledge of the distribution. All our implementation was done in C and run on a 1.7GHz Intel Xeon machine running Linux 2.6.9.

In all cases, we generated anti-correlated points using the data set generator presented in [2]. This generates points in the range $[0, 1]$ along each dimension. Anti-correlated data is most interesting to us because it leads to a large skyline set that needs a good representation. We averaged the performance of all the experiments by repeating each experiment 10 times with independent sets of points (and training thresholds for our Greedy algorithm). Unless otherwise specified, we used default values of one hundred thousand input points ($M = 100000$), four dimensions ($d = 4$), and $k = 10$ representative points. For the experiments on synthetic data, thresholds were generated according to the Normal distribution independently along each dimension, with parameters μ as mean and σ as standard deviation. For default values, μ and σ are 0.5 and 0.25 respectively so that the majority of the points are in the same range as the anti-correlated data points.[1] We evaluated the performance of each algorithm (measured as the fraction of random points drawn from the distribution that were covered)

[1] A subtle point here is that some points may have negative coordinates (since we sample from Gaussians) and the thresholds are then automatically satisfied. Similarly, some thresholds could be greater than 1 and then they can never be satisfied. However, this comparison is uniform across all algorithms and we vary μ and σ as well in our experiments.

n	2D	4D	6D
100	0.748574	0.473685	0.249502
1000	0.766047	0.538480	0.313911
10000	0.769406	0.544203	0.326666
100000	0.769732	0.544632	0.328671

TABLE I

COVERAGE OF OUR GREEDY ALGORITHM USING DIFFERENT TRAINING SET SIZES ($M = 100000$, $k = 10$, AND $\mu = 0.5$, $\sigma = 0.25$ IN ALL DIMENSIONS)

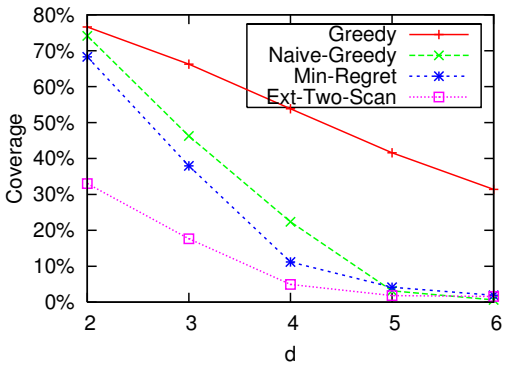


Fig. 1. Coverage percentage with varying dimensions ($M = 100000$, $n = 1000$, $k = 10$, and $\mu = 0.5$, $\sigma = 0.25$ in all dimensions)

by generating one million thresholds. For the experiments on the real data, the data set presented values along different attribuets. We used the values themselves as thresholds, and generated points separately for the evaluation.

We compared the following algorithms:

1) Our Greedy algorithm based on sampling (Algorithm IV-B) presented in Section IV.
2) The Naive-Greedy algorithm proposed in [14] for computing a distance-based representative skyline.
3) The Min-Regret algorithm [9] for approximating the skyline with minimal regret with linear utility functions.
4) The Ext-Two-Scan algorithm proposed in [8].

A. Experiments with synthetic data

We begin by presenting our experiments on synthetic data. Table I shows the performance of our Greedy algorithm as the size of the training set (n), i.e. number of thresholds sampled, is increased. We see from the table that beyond $n = 1000$ samples there is not much further improvement in performance (at most $1 - 2\%$) and so we fix this value for the remaining experiments. Note that it is good that such a relatively small value of n suffices since this affects the running time of the Greedy algorithm and, in the case where the training points are inferred from real data, not many are needed to make our algorithm perform well.

In Figure 1 we compare the coverage of all the algorithms with default values and the number of dimensions (d) varied. As would be expected, the coverage of all the algorithms declines with additional dimensions. Intuitively, this is because

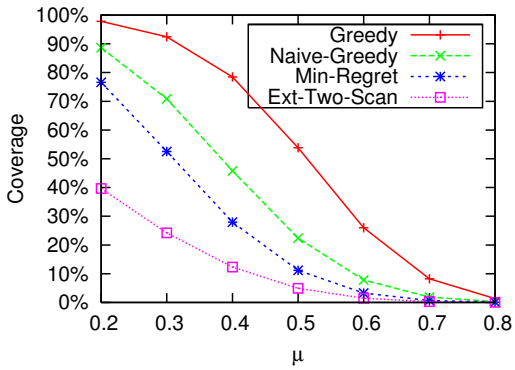


Fig. 4. Coverage percentage with varying μ ($M = 100000$, $n = 1000$, $d = 4$, $k = 10$, and $\sigma = 0.25$ in all dimensions)

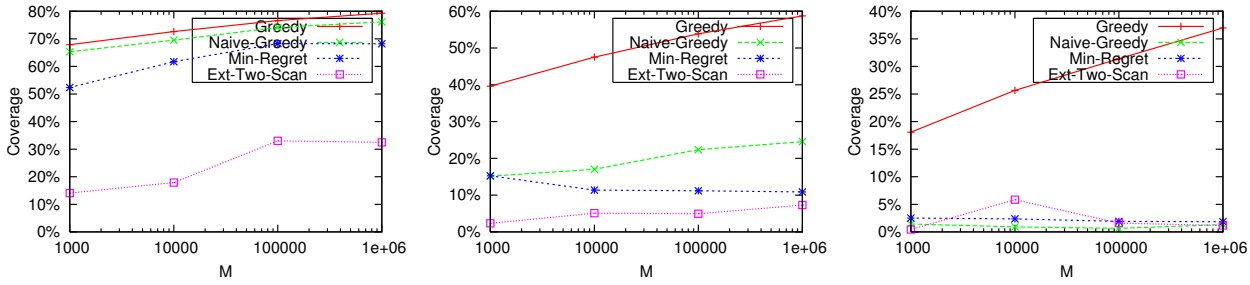
more dimensions need more attributes to be satisfied by the representative points. The coverage of our Greedy algorithm decreases gracefully in a near-linear fashion, in contrast with the other algorithms that drop off much more rapidly.

Figures 2[a-c] shows how the algorithms compare when the number of points (M) is varied for the case of $d = 2, 4$, and 6 dimensions, respectively. We observe that all the algorithms except Ext-Two-Scan perform well for the case of $d = 2$. For higher dimensions, all the other algorithms deteriorate in quality with respect to our Greedy algorithm, with a stark difference observed for the case of $d = 6$. Note that the coverage of our algorithm increases with M because having an increased number of choices increases the quality of the final solution. We vary the size of the reprentation (k) in Figure 3, again for the cases of 2, 4, and 6 dimensions. The coverage of all the algorithms are monotonically non-decreasing with k, as is to be expected. Once again we see a widening gap between the performance of our algorithm and the others as the number of dimensions increase.

Next, we varied the mean of the distribution used in the experiment, as seen in Figure 4. The purpose of this experiment is to show that, as the thresholds get higher, it is increasingly hard to cover most of the points. Similarly, we varied the standard deviation of the distribution in Figures 5[a-c]. Note that in these figures the axis for σ (x-axis) is in the decreasing direction. What these figures show is that, as the standard deviation is decreased (i.e., the target thresholds are more tightly clustered), our algorithm performs increasingly better versus the others. This is, once again, due to the advantage of distribution knowledge that our algorithm possesses.

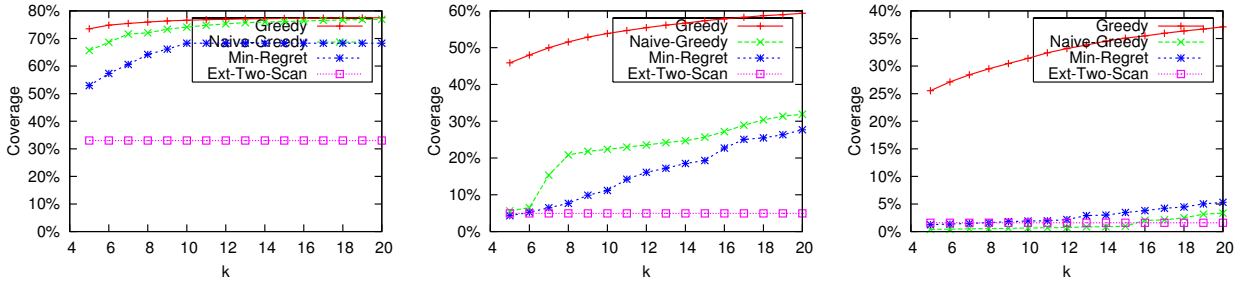
B. Experiments with real data

We experimented on two real data sets of real thresholds, called House and NBA. The House data consists of $127,931$ points with six dimensions collected from statistics of expenditures of families in the US on various utilities. The NBA data has $17,264$ five-dimensional points collected from the statistics of NBA players. For the database set of points, we used anti-correlated points as before, except we re-scaled them to be in the interval $[H - 2\sigma, H]$ in each interval, where H



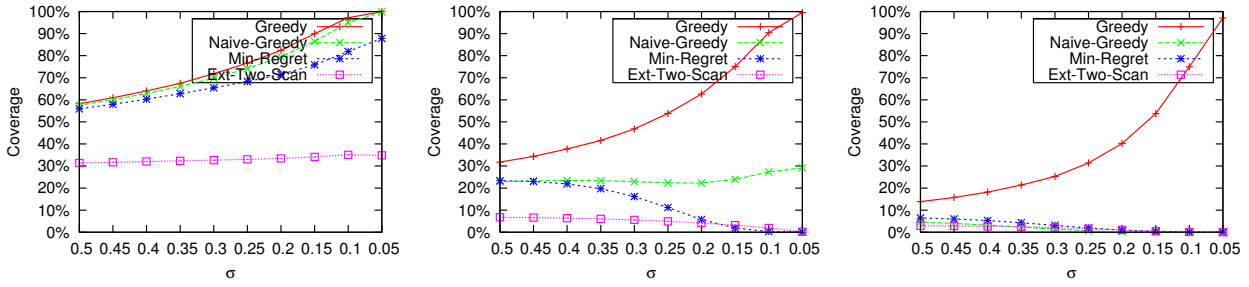
(a) 2D (b) 4D (c) 6D

Fig. 2. Coverage percentage with varying M ($k = 10$, $n = 1000$, and $\mu = 0.5$, $\sigma = 0.25$ in all dimensions)



(a) 2D (b) 4D (c) 6D

Fig. 3. Coverage percentage with varying k ($M = 100000$, $n = 1000$, and $\mu = 0.5$, $\sigma = 0.25$ in all dimensions)



(a) 2D (b) 4D (c) 6D

Fig. 5. Coverage percentage with varying σ ($M = 100000$, $n = 1000$, $k = 10$, and $\mu = 0.5$ in all dimensions)

is the maximum value the thresholds attained and σ is the average of the standard deviation of each attribute. (In the case of House this was $[6391.95, 10000.00]$ and for NBA it was $[9036.59, 10000]$.)

We show the performance of the algorithms in Figure 6 in which we vary the size of the reprentation (k). Our Greedy algorithms outstrips the other algorithms in both cases. In fact, for small values of k (e.g., near $k = 5$), it covers over twice the fraction of thresholds as the next best algorithm.

We also tested the attribute values on each of the attributes to see if they actually follow something like a Normal distribution. We plotted the attribute values for each of the dimensions on both the real data sets, House and NBA. Some dimensions exhibit cleaner Gaussians than others but all of them are indeed unimodal and nearly bell-shaped. We present the plots for the first dimensions of House as well as NBA. These plots could vary depending on scaling etc. so have to be taken with a grain of salt. However, it is nice that they are close to Gaussian in nature. These are shown in Figures 7 and 8.

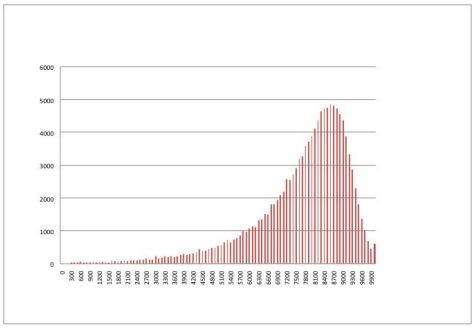
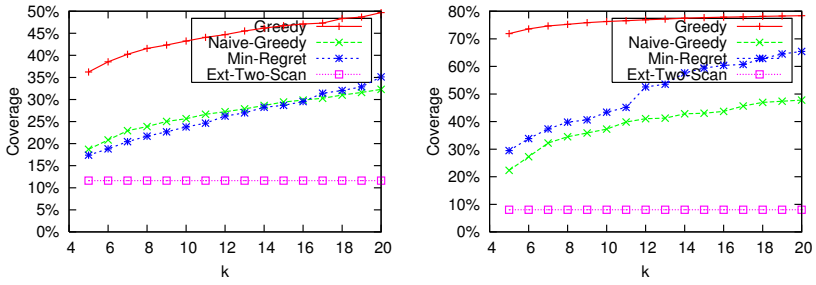


Fig. 7. Plots of attribute values of first dimension for House data



(a) House (b) NBA

Fig. 6. Coverage percentage with varying k for real data sets

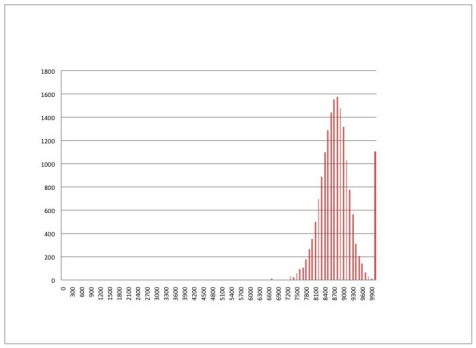


Fig. 8. Plots of attribute values of first dimension for NBA data

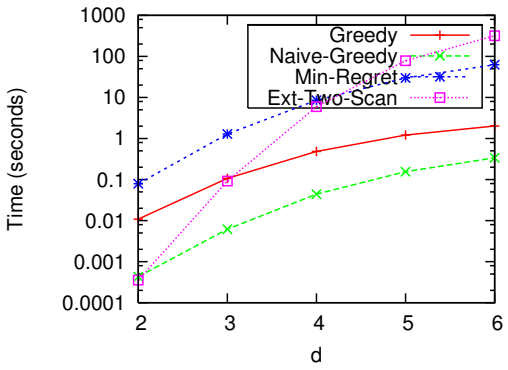


Fig. 9. Running time with varying d ($n = 1000$, $M = 100000$, $k = 10$, and $\mu = 0.5$ in all dimensions)

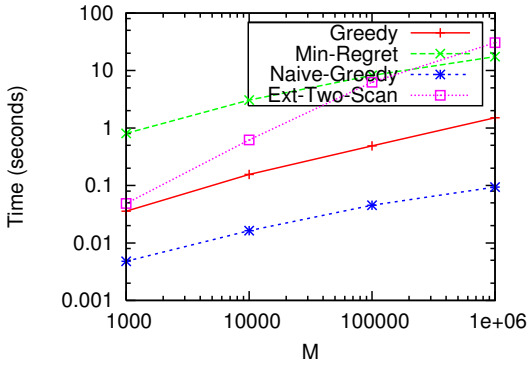


Fig. 10. Running time with varying M ($n = 1000$, $d = 4$, $k = 10$, and $\mu = 0.5$ in all dimensions)

C. Running Times

We compared the running times of the different algorithms based on our C implementations. As stated earlier, these times were averaged over 10 runs. In Figures 9 and 10 we vary the number of dimensions and the number of data points. We see that in both cases the Greedy algorithm completes within 1-2 seconds. Some of the other algorithms took in the order of 10s to 100s of seconds, though we did not try to optimize them.

VI. Related Work

Skyline computation, previously known as Pareto sets, admissible points, and maximal vectors, has been extensively studied in the theory and mathematics community since 1960s (see, e.g., [15]–[17]). The problem, and the name *skyline*, were introduced to the database community by Börzsönyi et al. [2]. Owing to the vast body of work on skylines, we restrict ourselves to mentioning only those most relevant to the context of representative skylines.

Motivated by the possibly large size of skylines, many variants of skylines have been considered. Lin et al. [6] and Tao et al. [5] consider finding k skyline tuples that best represent the contour of the entire skyline, called k representative skyline. [6] propose finding a set of k skyline tuples that dominate the most number of tuples. This approach is scale invariant but not stable. [5] illustrates that the approach of [6] could lead to an undesirable solution. This is essentially because of the fact that it is unstable. They propose an alternative distance-based solution which is to solve the *k-center* problem on the skyline tuples. This approach is stable but not scale invariant since their definition is based on distances. Also, in recent work, we [9] considered another approach to representative skylines, based on linear utility functions. One drawback of this work is that it tries to maximize the *minimum happiness* over all users; this may not be optimal for maximizing click probabilities *averaged* over all users. While we study user-based thresholds, there has been orthogonal interesting work on constrained subspace skyline computation [18] where range constrains on the skyline queries are specified.

Several other approaches have also been suggested; some

control the size of the output without explicitly specifying a bound. Xia et al. [7] propose ϵ-skyline queries where users specify weights on attributes, and use the parameter ϵ to control the size of the output. Mindolin et al. [19] propose *p-skyline* queries which is a framework that allows attributes to have different importance. To avoid asking users for weights explicitly, they offer an alternative approach to discover importance from user feedback. Lee et al. [20] avoid asking users for utility functions by requesting only partial ranking over attributes. Chan et al. [21] introduce the concept of skyline frequency as a way to measure the importance of a point based on the number of subsets of dimensions for which it is in the skyline. A related concept of k-dominance proposed by Chan et al. [8] relaxes the definition of dominance to a point dominating another if it dominates on at least k dimensions. All these approaches are used to reduce the size of the output.

An orthogonal approach to skylines, also used to displayed only k interesting tuples from the database is the *top-k* operator (see [22] for a recent survey). This approach is based on a widely-accepted assumption that utility functions exist. See for example PREFER [23], ONION [24], Ranked Join Indices [25]. The main benefit of representative skylines over such techniques is that it avoids asking users explicitly for inputs; users may be unnecessarily burdened to input weights on all attributes (sometimes weights are not even known). Yiu and Mamoulis [26], and Papadopoulos et. al. [27] propose the *top-k dominating* query to leverage the benefits of top-k as well as skylines; it is scale invariance but not stable. Goncalves and Vidal [28] propose two operators combining top-k and skylines, but require users to specify utility functions. There is a lot more work on both top-k as well as skylines that we do not mention here. Further references on top-k and variants of skylines can be found in [9] and [29] respectively.

VII. Future Directions

An open question raised by this work is to understand users' *joint* preferences along different attributes. For e.g., a user with a high threshold on hotel cost probably has a high threshold on hotel rating as well. The knowledge of such correlations (or the absence of it) can perhaps be exploited for better solutions. Perhaps the most interesting follow-up is to investigate whether user preferences can be learned over time in this context of databases (such as a hotel reservation system). This is a research direction in itself that has received attention in other domains. For instance, search engines are constantly trying to learn about users' preferences for contexts of keywords, or categories of URLs. This direction is reminiscent of multi-armed bandit type problems where a standard solution approach is to *explore* and *exploit*. When a system starts off, it may not have a good idea about users' preferences; over time, it understands the preferences better. As a steady state solution, the system *exploits* the current knowledge of user preferences by serving results based on it, while continuing to *explore* and learn the preferences better. It would be interesting to investigate such approaches here.

References

[1] P. Godfrey, R. Shipley, and J. Gryz, "Algorithms and analyses for maximal vector computation," *VLDB J.*, vol. 16, no. 1, pp. 5–28, 2007, also appeared in VLDB'05.

[2] S. Börzsönyi, D. Kossmann, and K. Stocker, "The skyline operator," in *ICDE*, 2001, pp. 421–430.

[3] J. L. Bentley, H. T. Kung, M. Schkolnick, and C. D. Thompson, "On the average number of maxima in a set of vectors and applications," *J. ACM*, vol. 25, no. 4, pp. 536–543, 1978.

[4] P. Godfrey, "Skyline cardinality for relational processing," in *FoIKS*, 2004, pp. 78–97.

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

[6] X. Lin, Y. Yuan, Q. Zhang, and Y. Zhang, "Selecting stars: The k most representative skyline operator," in *ICDE*, 2007, pp. 86–95.

[7] T. Xia, D. Zhang, and Y. Tao, "On skylining with flexible dominance relation," in *ICDE*, 2008, pp. 1397–1399.

[8] 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.

[9] D. Nanongkai, A. Das Sarma, A. Lall, R. J. Lipton, and J. Xu, "Regret-minimizing representative databases," *PVLDB*, 2010.

[10] T. J. Gilbride and G. M. Alleny, "A choice model with conjunctive, disjunctive, and compensatory screening rules," *INFORMS: Marketing Science*, vol. 23, no. 3, pp. 391–406, 2004.

[11] G. Cornuejols, M. Fisher, and G. Nemhauser, "Location of bank accounts to optimize float: An analytic study of exact and approximate algorithms," *Management Science*, pp. 789–810, 1977.

[12] G. Nemhauser, L. Wolsey, and M. Fisher, "An analysis of approximations for maximizing submodular set functions," *Mathematical Programming*, vol. 14, no. 1, pp. 265–294, 1978.

[13] M. Mitzenmacher and E. Upfal, *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. New York, NY, USA: Cambridge University Press, 2005.

[14] B. Jiang and J. Pei, "Online interval skyline queries on time series," in *ICDE*, 2009.

[15] 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*, vol. 11, no. 2, pp. 249–269, 1966.

[16] H. T. Kung, F. Luccio, and F. P. Preparata, "On finding the maxima of a set of vectors," *J. ACM*, vol. 22, no. 4, pp. 469–476, 1975, also in FOCS'74.

[17] J. Matousek, "Computing dominances in e^n," *Inf. Process. Lett.*, vol. 38, no. 5, pp. 277–278, 1991.

[18] E. Dellis, A. Vlachou, I. Vladimirskiy, B. Seeger, and Y. Theodoridis, "Constrained subspace skyline computation," in *CIKM*, 2006.

[19] D. Mindolin and J. Chomicki, "Discovering relative importance of skyline attributes," *PVLDB*, vol. 2, no. 1, pp. 610–621, 2009.

[20] J. Lee, G. won You, and S. won Hwang, "Personalized top-k skyline queries in high-dimensional space," *Inf. Syst.*, vol. 34, no. 1, pp. 45–61, 2009.

[21] C. Y. Chan, H. V. Jagadish, K.-L. Tan, A. K. H. Tung, and Z. Zhang, "On high dimensional skylines," in *EDBT*, 2006, pp. 478–495.

[22] I. F. Ilyas, G. Beskales, and M. A. Soliman, "A survey of top- query processing techniques in relational database systems," *ACM Comput. Surv.*, vol. 40, no. 4, 2008.

[23] V. Hristidis, N. Koudas, and Y. Papakonstantinou, "Prefer: A system for the efficient execution of multi-parametric ranked queries," in *SIGMOD Conference*, 2001, pp. 259–270.

[24] 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 Conference*, 2000, pp. 391–402.

[25] P. Tsaparas, T. Palpanas, Y. Kotidis, N. Koudas, and D. Srivastava, "Ranked join indices," in *ICDE*, 2003, pp. 277–.

[26] M. L. Yiu and N. Mamoulis, "Multi-dimensional top- dominating queries," *VLDB J.*, vol. 18, no. 3, pp. 695–718, 2009, also VLDB'07.

[27] A. N. Papadopoulos, A. Lyritsis, A. Nanopoulos, and Y. Manolopoulos, "Domination mining and querying," in *DaWaK*, 2007, pp. 145–156.

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

[29] A. Das Sarma, A. Lall, D. Nanongkai, and J. Xu, "Randomized multi-pass streaming skyline algorithms," *PVLDB*, vol. 2, no. 1, pp. 85–96, 2009.