

Distance-based Representative Skyline

Yufei Tao¹ Ling Ding¹ Xuemin Lin² Jian Pei³

¹Chinese University of Hong Kong ²University of New South Wales and NICTA ³Simon Fraser University

Abstract—Given an integer k , a *representative skyline* contains the k skyline points that best describe the tradeoffs among different dimensions offered by the full skyline. Although this topic has been previously studied, the existing solution may sometimes produce k points that appear in an arbitrarily tiny cluster, and therefore, fail to be representative. Motivated by this, we propose a new definition of representative skyline that minimizes the distance between a non-representative skyline point and its nearest representative. We also study algorithms for computing distance-based representative skylines. In 2D space, there is a dynamic programming algorithm that guarantees the optimal solution. For dimensionality at least 3, we prove that the problem is NP-hard, and give a 2-approximate polynomial time algorithm. Using a multidimensional access method, our algorithm can directly report the representative skyline, without retrieving the full skyline. We show that our representative skyline not only better captures the contour of the entire skyline than the previous method, but also can be computed much faster.

I. INTRODUCTION

Given a set \mathcal{D} of multidimensional points, the *skyline* [1] consists of the points that are not dominated by any other point. Specifically, a point p *dominates* another p' if the coordinate of p is smaller than or equal to that of p' on all dimensions, and strictly smaller on at least one dimension. Figure 1 shows a classical example with a set \mathcal{D} of 13 points, each capturing two properties of a hotel: its distance to the beach (the horizontal coordinate), and price (the vertical coordinate). The skyline has 8 points p_1, p_2, \dots, p_8 .

Skyline retrieval has received considerable attention from the database community, resulting in a large number of interesting results as surveyed in Section VI. These research efforts reflect the crucial importance of skylines in practice. In particular, it is well-known [2] that there exists an inherent connection between skylines and top-1 queries. Specifically, given a preference function $f(p)$ which calculates a *score* for each point p , a *top-1* query returns the data point with the lowest score. As long as function $f(\cdot)$ is *monotone*¹, the top-1 result is definitely in the skyline. Conversely, every skyline point is guaranteed to be the top-1 result for at least one preference function $f(\cdot)$.

The skyline operator is particularly useful in scenarios of multi-criteria optimization where it is difficult, or even impossible, to formulate a good preference function. For example, consider a tourist that wants to choose from the hotels in Figure 1 a good one offering a nice tradeoff between price and distance. S/he may not be sure about the relatively weighting

¹Namely, $f(p)$ grows as long as the coordinate of p along any dimension increases.

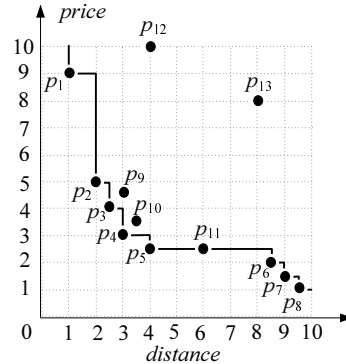


Fig. 1. A skyline example

of the two dimensions, or in general, whether the quality of a hotel should be assessed through a linear, quadratic, or other types of preference functions. In this case, it would be reasonable to return the skyline, so that the tourist can directly compare the tradeoffs offered by different skyline points. For example, as far as tradeoffs are concerned, the skyline points in Figure 1 can be divided into three subsets S_1, S_2, S_3 :

- $S_1 = \{p_1\}$, which includes a hotel that is very close to the beach, but rather expensive;
- $S_2 = \{p_2, p_3, p_4, p_5\}$, where the hotels are farther away from the beach, but cheaper;
- $S_3 = \{p_6, p_7, p_8\}$, where the hotels are the cheapest, but far from the beach.

In this paper, we study the problem of retrieving *representative skyline*, which includes a small number k of skyline points that best describe the possible tradeoffs in the full skyline. For example, given $k = 3$, our solution will report p_1, p_4 , and p_7 , each of which comes from a distinct subset illustrated above, representing a different tradeoff.

Representative skylines are especially helpful in web-based recommendation systems such as the one in our previous hotel example. Skyline computation can be a rather costly process, particularly in high dimensional spaces. This necessitates a long waiting period before the entire skyline is delivered to the user, which may potentially incur negative user experience. A better approach is to return a few early skyline points representing the *contour* of the final skyline, and then, progressively refine the contour by reporting more skyline points. In this way, a user can understand the possible tradeoffs s/he may eventually get, well before the query finishes. Moreover, given such valuable information, the user may also notify the web server to stop fetching more skyline points offering uninteresting tradeoffs, thus significantly reducing the processing time.

Furthermore, representative skylines also enable a *drill-down* manner to explore the full skyline. In practice, the number of skyline points can be large, and increases rapidly with the dimensionality [3]. In fact, even a two-dimensional dataset may have a sizable skyline, when the data distribution is “anti-correlated” [1]. Presenting a user with a large result set creates serious confusion and may even complicate the process of selecting the best object. A representative skyline, on the other hand, gives a user a concise, high-level, summary of the entire skyline. The user can then identify a few representatives, and ask the server to provide the other skyline points that are similar to those representatives.

To the best of our knowledge, so far there is only a single piece of work [4] on representative skylines. The work of [4], however, adopts a definition of representativeness that sometimes returns skyline points that are barely representative. For instance, as detailed in the next section, with $k = 3$ the definition in [4] advocates a representative set of $\{p_3, p_4, p_5\}$ in the example of Figure 1. Clearly, this is a poor choice because all the points in the set come from S_2 , and do not indicate the tradeoffs provided by S_1 and S_3 .

Motivated by this, we propose a new definition of representative skyline. Our definition builds on the intuition that a good representative skyline should be such that, for every non-representative skyline point, there is a nearby representative. Therefore, we aim at minimizing the maximum distance between a non-representative skyline point and its closest representative. We also study algorithms for finding distance-based representative skylines. In 2D space, there is a dynamic programming algorithm that optimally settles the problem in polynomial time. For dimensionality at least 3, we show that the problem is NP-hard, and give a 2-approximate polynomial algorithm. Utilizing a multidimensional access method, our algorithm can quickly identify the k representatives without extracting the entire skyline. Furthermore, the algorithm is progressive, and does not require the user to specify the value of k . Instead, it continuously returns representatives that are guaranteed to be a 2-approximate solution *at any moment*, until either manually terminated or eventually producing the full skyline. We provide both theoretical and empirical evidence that, compared to the definition in [4], our representative skyline not only better captures the contour of the full skyline, but also can be computed considerably faster.

The rest of the paper is organized as follows. Section II clarifies our formulation of representative skyline, and analyzes its advantages over that of [4]. Section III presents an algorithm for finding the optimal representative skyline in 2D space. Section IV tackles retrieval of representative skylines in dimensionality at least 3. Section V evaluates the proposed techniques with extensive experiments. Section VI surveys the previous literature on skyline. Finally, Section VII concludes the paper with a summary of our results.

II. REPRESENTATIVE SKYLINES AND BASIC PROPERTIES

Let \mathcal{D} be a set of d -dimensional points. Without loss of generality, we consider that each dimension of the data space

has been normalized to a unit range $[0, 1]$ with length 1. In other words, the maximum distance between any two points is \sqrt{d} , where d is the dimensionality. We use \mathcal{S} to denote the full skyline of \mathcal{D} .

In the sequel, we first review the only existing definition of representative skyline proposed in [4], and elaborate its defects. Then, we propose our definition, and explain its superiority over the proposition in [4]. Finally, we identify the underlying optimization problem.

Defects of the existing formulation. Given an integer k , Lin et al. [4] define the representative skyline as the set \mathcal{K} of k skyline points of \mathcal{D} that maximizes the number of non-skyline points dominated by at least one point in \mathcal{K} . We refer to this definition as *max-dominance representative skyline*.

For example, consider Figure 1 and $k = 3$. The max-dominance representative skyline is $\mathcal{K} = \{p_3, p_4, p_5\}$. To understand why, first note that every non-skyline point is dominated by at least one point in \mathcal{K} . Furthermore, exclusion of any point from \mathcal{K} will leave at least one non-skyline point un-dominated. Specifically, omitting p_3 from \mathcal{K} renders p_9 un-dominated, and omitting p_4 (p_5) renders p_{10} (p_{11}) un-dominated. As discussed in Section I, $\mathcal{K} = \{p_3, p_4, p_5\}$ is not sufficiently representative, because all the points in \mathcal{K} are from the same cluster.

To enable a direct comparison with our definition of representative skyline to be explained shortly, we introduce the concept of *representation error* of \mathcal{K} , denoted as $Er(\mathcal{K}, \mathcal{S})$. Intuitively, a representative skyline \mathcal{K} is good if, for every non-representative skyline point $p \in \mathcal{S} - \mathcal{K}$, there is a representative in \mathcal{K} close to p . Hence, $Er(\mathcal{K}, \mathcal{S})$ quantifies the representation quality as the maximum distance between a non-representative skyline point in $\mathcal{S} - \mathcal{K}$ and its nearest representative in \mathcal{K} , under Euclidean distance, or formally:

$$Er(\mathcal{K}, \mathcal{S}) = \max_{p \in \mathcal{S} - \mathcal{K}} \{ \min_{p' \in \mathcal{K}} \|p, p'\| \}. \quad (1)$$

The above error metric makes more sense than the apparent alternative of taking the sum, instead of max. We will come back to this issue later in this section.

In the sequel, when the second parameter of function $Er(\cdot, \cdot)$ is the full skyline \mathcal{S} of \mathcal{D} , we often abbreviate $Er(\mathcal{K}, \mathcal{S})$ as $Er(\mathcal{K})$. For example, in Figure 1, when $\mathcal{K} = \{p_3, p_4, p_5\}$, $Er(\mathcal{K}) = \|p_5, p_8\|$, i.e., p_8 is the point that is worst represented by \mathcal{K} .

We present a lemma showing that the representation error of max-dominance representative skyline can be arbitrarily bad.

LEMMA 1: For any k , there is a 2D dataset \mathcal{D} such that the $Er(\mathcal{K})$ of the max-dominance representative skyline \mathcal{K} is at least $\sqrt{2} - \delta$ for any small $\delta > 0$.

PROOF: Given a δ , we will construct such a dataset \mathcal{D} with cardinality $2k + 1$. First, create a *gadget* with $2k$ points p_1, p_2, \dots, p_{2k} as follows. Take a square with side length $\delta/\sqrt{2}$. Along the square’s anti-diagonal, evenly put down k points p_1, p_2, \dots, p_k as shown in Figure 2, making sure that p_1 and p_k are not at the two ends of the anti-diagonal. For each p_i ($1 \leq i \leq k$),

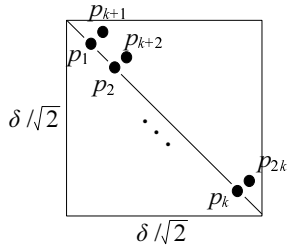


Fig. 2. A gadget in the proof of Lemma 1

place another point p_{k+i} that is exclusively dominated by p_i , as in Figure 2. When this is done, place the gadget into the data space, aligning the upper-left corner of its square with that of the data space. Finally, put another data point p_{2k+1} at the lower-right corner of the data space. Points $p_1, p_2, \dots, p_{2k+1}$ constitute \mathcal{D} .

The max-dominance representative skyline \mathcal{K} of \mathcal{D} includes $\{p_1, p_2, \dots, p_k\}$, and the full skyline \mathcal{S} of \mathcal{D} is $\mathcal{K} \cup \{p_{2k+1}\}$. Hence, $Er(\mathcal{K})$ equals the Euclidean distance between p_{2k+1} and p_k , which is at least $\sqrt{2}-\delta$, noticing that the anti-diagonals of the data space and the gadget square have length $\sqrt{2}$ and δ respectively. \square

Recall that $\sqrt{2}$ is the maximum possible distance of two points in 2D space. Hence, Lemma 1 suggests that the representation error $Er(\mathcal{K})$ can arbitrarily approach this worst value, no matter how large k is. Straightforwardly, a similar result holds in any dimensionality d , replacing $\sqrt{2}$ with \sqrt{d} .

Another drawback of max-dominance representative skyline is that it is costly to compute. In 2D space, even if the skyline \mathcal{S} is available, it still takes $O(|\mathcal{D}| \log m + m^2 \cdot k)$ time [4] to find an optimal solution, where $|\mathcal{D}|$ and m are the sizes of the dataset \mathcal{D} and \mathcal{S} , respectively. In other words, the cost is as expensive as scanning \mathcal{D} several times. The overhead is even greater in higher dimensional spaces. First, when the dimensionality is at least 3, the problem is NP-hard so only approximate solutions are possible in practice. The best algorithm in [4] finds a solution with provably good quality guarantees in $O(|\mathcal{D}| \cdot m)$ time. Apparently, this is prohibitive for large datasets. Acknowledging the vast cost complexity, the authors of [4] also provide another heuristic algorithm that is faster but may return a solution with poor quality.

Our formulation. We introduce the concept of *distance-based representative skyline* as follows.

DEFINITION 1: Let \mathcal{D} be a multidimensional dataset and \mathcal{S} its skyline. Given an integer k , the *distance-based representative skyline* of \mathcal{D} is a set \mathcal{K} of k skyline points in \mathcal{S} that minimizes $Er(\mathcal{K}, \mathcal{S})$ as calculated by Equation 1. \square

In other words, the distance-based representative skyline consists of k skyline points that achieve the lowest representation error. For example, in Figure 1, with $k = 3$ the distance-based representative skyline is $\mathcal{K} = \{p_1, p_4, p_7\}$, whose $Er(\mathcal{K})$ equals $\|p_4, p_2\|$.

The distance-based representative skyline is essentially the optimal solution of the *k-center problem* [5] on the full skyline

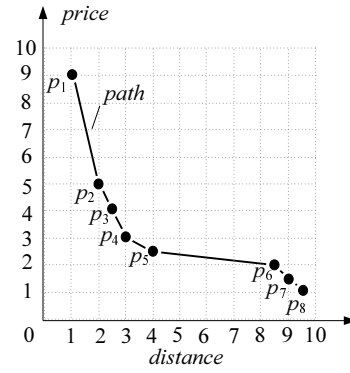


Fig. 3. A path in the proof of Lemma 2

\mathcal{S} . As a result, the distance-based representative skyline also shares the properties of k -center results. One, particularly, is that the result is not sensitive to the densities of clusters. This is very important for capturing the *contour* of the skyline. Specifically, we do not want to allocate many representatives to a cluster simply because it has a large density (as shown in the experiments, the max-dominance representative skyline suffers from this drawback). Instead, we would like to distribute the representatives evenly along the skyline, regardless of the densities of the underlying clusters. This also answers the question we posed earlier why the error metric of Equation 1 is better than its sum-counterpart $\sum_{p \in \mathcal{S} - \mathcal{K}} \{\min_{p' \in \mathcal{K}} \|p, p'\|\}$. The latter tends to give more representatives to a dense cluster, because doing so may reduce the distances of a huge number of points to their nearest representatives, which may outweigh the benefit of trying to reduce such distances of points in a faraway sparse cluster.

Our formulation of representative skyline enjoys an attractive theoretical guarantee, as shown in the following lemma in correspondence to Lemma 1.

LEMMA 2: For any $k \geq 2$ and any 2D dataset \mathcal{D} with cardinality at least k , the representation error $Er(\mathcal{K})$ of the distance-based representative skyline \mathcal{K} is strictly smaller than $2/k$.

PROOF: Denote the full skyline of \mathcal{D} as \mathcal{S} , and the number of points in \mathcal{S} as m . Let us sort the points in \mathcal{S} in ascending order of their x-coordinates, and let the sorted order be p_1, p_2, \dots, p_m . Use a segment to connect each pair of consecutive skyline points in the sorted list. This way, we get a *path*, consisting of $m - 1$ segments. Define the *length* of the path as the total length of all those segments. The length of the path is strictly lower than 2, i.e., the total length of the two axes of the data space. Figure 3 shows the path formed by the 8 skyline points in the example of Figure 1.

To prove the lemma, we will construct a set \mathcal{K}' of k skyline points such that $Er(\mathcal{K}')$ is already smaller than $2/k$. As the distance-based representative skyline \mathcal{K} minimizes the representation error, its $Er(\mathcal{K})$ can be at most $Er(\mathcal{K}')$, an hence, must be lower than $2/k$ as well.

Specifically, we create \mathcal{K}' as follows. Initially, \mathcal{K}' contains the first skyline point p_1 of the sorted list, i.e., the beginning

of the path. Then, we walk along the path, and keep a *meter* measuring the distance traveled. As long as the meter is smaller than $2/k$, we ignore all the skyline points seen on the path. Once the meter reaches or exceeds $2/k$, we will add to \mathcal{K}' the next skyline point encountered, and reset the meter to 0. Next, the above process is repeated until we have finished the entire path. We call the points already in \mathcal{K}' at this moment the *picked representatives*. Since the length of the entire path is less than 2, there are at most k picked representatives. In case \mathcal{K}' has less than k picked representatives, we arbitrarily include more skyline points of \mathcal{S} into \mathcal{K}' to increase the size of \mathcal{K}' to k .

To prove $Er(\mathcal{K}') < 2/k$, we show a stronger statement: for every skyline point $p \in \mathcal{S}$, there is a picked representative whose distance to p is smaller than $2/k$. This is trivial if p is a picked representative itself. Otherwise, let p' be the picked representative right before we came to p in walking along the path. $\|p', p\|$ is at most the length of the segments on the path from p' to p , which is smaller than $2/k$ by the way we decide picked representatives. \square

Comparing Lemmas 1 and 2, it is clear that distance-based representative skyline gives a much stronger worst-case guarantee on the representation quality. Furthermore, its guarantee meets our intuitive expectation that the representation precision ought to *monotonically grow* with k . This is a property that max-dominance representative skyline fails to fulfill, as its representation error can be arbitrarily bad regardless of how large k is (Lemma 1). We note that a result analogous to Lemma 2 also exists in higher dimensionality d . Its derivation, however, needs to follow a different rationale that, in any dimensionality, it is not possible to place $k + 1$ disjoint balls with the same radius r , as r grows large enough. We omit the details here.

Problem. From now on, we will use the term *representative skyline* to refer to any subset \mathcal{K} of the full skyline \mathcal{S} . If \mathcal{K} has k points, we say that it is *size- k* . The problem we study in this paper can be defined as:

PROBLEM 1: Given an integer k , find an optimal size- k representative skyline \mathcal{K} that has the smallest representation error $Er(\mathcal{K}, \mathcal{S})$ given in Equation 1 among all representative skylines of size k . \square

Note that an optimal representative skyline is a distance-based representative skyline in Definition 1. Sometimes it is computationally intractable to find the optimal solution. In this case, we instead aim at computing a representative skyline whose representation error is as low as possible.

III. THE TWO-DIMENSIONAL CASE

In this section, we give an algorithm for solving Problem 1 optimally in 2D space. We consider that the skyline \mathcal{S} of dataset \mathcal{D} has already been computed using an existing algorithm. Let m be the size of \mathcal{S} . Denote the skyline points in \mathcal{S} as p_1, p_2, \dots, p_m , sorted in ascending order of their x-coordinates.

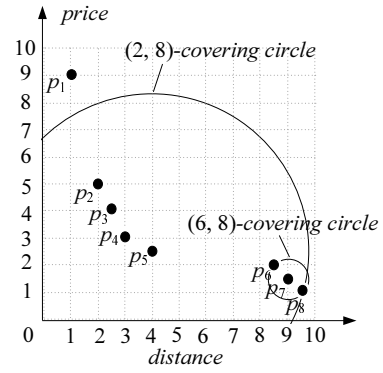


Fig. 4. Covering circles

We adopt the notation \mathcal{S}_i to represent $\{p_1, p_2, \dots, p_i\}$ where $i \leq m$. Specially, define $\mathcal{S}_0 = \emptyset$. Introduce a function $opt(i, t)$ to be the optimal size- t representative skyline of \mathcal{S}_i , where $t \leq i$. Hence, the optimal size- k representative skyline of \mathcal{S} is essentially $opt(m, k)$. Let function $optEr(i, t)$ be the representation error of $opt(i, t)$ with respect to \mathcal{S}_i , or formally, $optEr(i, t) = Er(opt(i, t), \mathcal{S}_i)$, where $Er(\cdot, \cdot)$ is given in Equation 1.

For any $1 \leq i \leq j \leq m$, we use $radius(i, j)$ to denote the radius of the smallest circle that (i) covers points p_i, p_{i+1}, \dots, p_j , and (ii) centers at one of these $j - i + 1$ points. Call the circle the (i, j) -covering circle, and denote its center as $center(i, j)$. For example, consider the skyline in Figure 3. Figure 4 shows the $(2, 8)$ - and $(6, 8)$ -covering circles, whose centers are p_5 and p_7 , respectively. Hence, $center(2, 8) = p_5$ and $center(6, 8) = p_7$.

There exists a recursive equation about $optEr(i, t)$ when $t \geq 2$:

$$optEr(i, t) = \min_{j=t}^{i-1} \{ \max\{optEr(j-1, t-1), radius(j, i)\} \} \quad (2)$$

The above equation is based on the following rationale. Assume, without loss of generality, that the optimal size- t representative skyline of \mathcal{S}_i is $\{p_{j_1}, p_{j_2}, \dots, p_{j_t}\}$ with $1 \leq j_1 < j_2 < \dots < j_t \leq i$, i.e., p_{j_1}, \dots, p_{j_t} are in ascending order of their x-coordinates. Let p_j be the first point (in ascending order of x-coordinates) in \mathcal{S}_i that has p_{j_t} as its nearest representative. Then, $\{p_{j_1}, \dots, p_{j_{t-1}}\}$ must be the optimal size- $(t-1)$ representative skyline of \mathcal{S}_{j-1} , and p_{j_t} must be $center(j, i)$.

Let v be the value of j where Equation 2 reaches its minimum; we have:

$$opt(i, t) = opt(v-1, t-1) \cup \{center(v, i)\} \quad (3)$$

Equations 2 and 3 point to a dynamic programming algorithm $2D-opt$ in Figure 5 for computing $opt(k, m)$, i.e., the size- k representative skyline of \mathcal{D} .

Time complexity. As explained shortly, Line 1 of $2D-opt$ can be implemented in $O(m^2)$ time, where m is the size of the full skyline \mathcal{S} of \mathcal{D} . Line 2 obviously requires $O(m)$ time. Lines 3-6 perform $k-2$ iterations. Each iteration evaluates Equations 2 and 3 m times respectively. Regardless of i and

Algorithm 2D-opt (\mathcal{S}, k)Input: the skyline \mathcal{S} of dataset \mathcal{D} and an integer k Output: the representative skyline of \mathcal{D}

1. for each pair of (i, j) such that $1 \leq i \leq j \leq m$, derive $radius(i, j)$ and $center(i, j)$.
 2. set $opt(i, 1) = \{center(1, i)\}$ and $optEr(i, 1) = radius(1, i)$ for each $1 \leq i \leq m$
 3. for $t = 2$ to $k - 1$
 4. for $i = t$ to m
 5. compute $optEr(i, t)$ by Equation 2
 6. compute $opt(i, t)$ by Equation 3
 7. compute $optEr(k, m)$ and $opt(k, m)$ by Equations 2 and 3
 8. return $opt(k, m)$
-

Fig. 5. An optimal algorithm for computing 2D representative skylines

t , every evaluation of Equation 2 can be completed in $O(m)$ time, and that of Equation 3 in $O(k)$ time. Hence, Lines 3-6 altogether incur $O(m^2(k-2))$ cost. Finally, Line 7 requires $O(m)$ time. Therefore, the overall complexity of 2D-opt is $O(m^2(k-2) + m)$. Note that this is much lower than the complexity of $O(|\mathcal{D}| \cdot \log m + m^2 \cdot k)$ (mentioned in Section III) of computing the optimal 2D max-dominance skyline.

Computing covering circles. Next, we give an $O(m^2)$ -time algorithm to find all the covering circles, i.e., $radius(i, j)$ and $center(i, j)$ for all $1 \leq i \leq j \leq m$. Note that one cannot hope to do any better because there are $\Omega(m^2)$ circles to decide.

First, it is easy to see that

$$radius(i, j) = \min_{u=i}^j \{\max\{\|p_i, p_u\|, \|p_u, p_j\|\}\}. \quad (4)$$

Let $p_u.radius(i, j) = \max\{\|p_i, p_u\|, \|p_u, p_j\|\}$. Equation 4 can be re-written as:

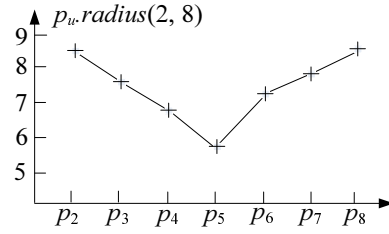
$$radius(i, j) = \min_{u=i}^j p_u.radius(i, j), \quad (5)$$

Thus, $center(i, j)$ equals the p_u where the above equation reaches its minimum.

As u moves from i to j , the value of $p_u.radius(i, j)$ initially decreases and then increases, exhibiting a V-shape. For the example of Figure 4, we plot $p_u.radius(2, 8)$ as u grows from 2 to 8 in Figure 6. Since $p_u.radius(2, 8)$ is the lowest at $u = 5$, the $(2, 8)$ -covering circle centers at p_5 , as shown in Figure 4.

The V-shape property offers an easy way, called *simple scan*, of finding $radius(i, j)$ and $center(i, j)$ as follows. We only need to inspect p_i, p_{i+1}, \dots, p_j in this order, and stop once $p_u.radius(i, j)$ starts to increase, where u is the point being inspected. At this moment, we have just passed the minimum of Equation 5. Hence, we know $center(i, j) = p_{u-1}$ and $radius(i, j) = p_{u-1}.radius(i, j)$. A simple scan needs $O(j-i)$ time to decide a $radius(i, j)$. This, however, results in totally $O(m^3)$ time in determining all the covering circles, which makes the time complexity of our algorithm 2D-opt $O(m^3)$ as well.

We bring down the time to $O(m^2)$ with a method called *collective pass*, which obtains the (i, i) -, $(i, i+1)$ -, ..., (i, m) -covering circles collectively in *one* scan from p_i to p_m in

Fig. 6. Plot of $p_u.radius(2, 8)$ for $2 \leq u \leq 8$

$O(m-i)$ time. Since a collective pass is needed for every $1 \leq i \leq m$, overall we spend $O(m^2)$ time.

The collective pass is based on a crucial observation in the following lemma. We omit the proof due to the space constraint.

LEMMA 3: For any $i \leq j_1 < j_2$, it holds that $center(i, j_1) \leq center(i, j_2)$.

Let us say that we perform a simple scan to obtain the (i, j_1) -covering circle, and find that it centers at p_u for some $u \in [i, j_1]$, i.e., $center(i, j_1) = p_u$. The result of Lemma 3 implies that we do not need to look back at the points already scanned to locate the center of the $(i, j_1 + 1)$ -covering circle. In other words, $center(i, j_1 + 1)$ must be either p_u or some point that we have not passed yet. Hence, we can pretend that we have been doing a simple scan for searching the $center(i, j_1 + 1)$ -covering circle so far, and continue the scan from p_u . Following this idea, a collective pass starts by running a simple scan for the (i, i) -covering circle, later switches to the $(i, i+1)$ -covering circle, and so on, until eventually the (i, m) -covering circle. The time is $O(m-i)$, same as a simple scan from p_i to p_m .

IV. THE HIGHER-DIMENSIONAL CASE

We proceed to study Problem 1 in dimensionality $d \geq 3$. Section IV-A shows that no polynomial time exists for finding the optimal solution, and presents a method for obtaining a 2-approximate solution. Sections IV-B and IV-C discuss how to improve the efficiency of the method by using a multidimensional access method.

A. NP-hardness and 2-approximation

LEMMA 4: For any dimensionality $d \geq 3$, Problem 1 is NP-hard.

PROOF: We first establish the NP-hardness at $d = 3$, before extending the result to any higher d . We reduce the 2D k -center problem, which is NP-hard [5], to Problem 1. Specifically, given a set S of 2D points p_1, p_2, \dots, p_n , the k -center problem aims at finding a subset S_k of S with k points that minimizes

$$\max_{p \in S} \{ \min_{p' \in S_k} \|p, p'\| \}. \quad (6)$$

We convert S to a 3D dataset \mathcal{D} as follows. Given a point $p_i \in S$ ($1 \leq i \leq n$) with coordinates $p_i[x]$ and $p_i[y]$, we create a point p'_i in \mathcal{D} whose coordinates $p'_i[x]$, $p'_i[y]$ and $p'_i[z]$ are: $p'_i[x] = \sqrt{2}p_i[x]$, $p'_i[y] = \sqrt{2}p_i[y] - p_i[x]$, and $p'_i[z] = -\sqrt{2}p_i[y] - p_i[x]$.

The resulting \mathcal{D} has two properties. First, it preserves the mutual distances in S , namely, for any $1 \leq i \leq j \leq n$, $\|p_i, p_j\| = \|p'_i, p'_j\|/2$. Second, no two points p'_i and p'_j can dominate each other, namely, if $p'_i[x] \leq p'_j[x]$ and $p'_i[y] \leq p'_j[y]$, then it must hold that $p'_i[z] \geq p'_j[z]$. Hence, if we could find the size- k representative skyline \mathcal{K} of \mathcal{D} in polynomial time, the points in S corresponding to those in \mathcal{K} would constitute the optimal solution S_k to the k -center problem.

Based on the 3D result, it is easy to show that Problem 1 is also NP-hard for any $d > 3$. Specifically, given any 3D dataset \mathcal{D} , we convert it to a d -dimensional dataset \mathcal{D}' by assigning $(d-3)$ 0's as the missing coordinates to each point in \mathcal{D} . The size- k representative skyline of \mathcal{D}' is also the size- k representative skyline of \mathcal{D} . Thus, we have found a polynomial time reduction from the 3D problem to the d -dimensional problem. \square

Fortunately, it is not hard to find a 2-approximate solution \mathcal{K} . Namely, if the optimal representative skyline is \mathcal{K}^* , the representation error of \mathcal{K} is at most twice as large as that of \mathcal{K}^* , i.e., $Er(\mathcal{K}, S) \leq 2 \cdot Er(\mathcal{K}^*, S)$, where $Er(\cdot, \cdot)$ is given in Equation 1.

Such a \mathcal{K} can be found by a standard greedy algorithm [6] for the k -center problem. Specifically, first we retrieve the skyline \mathcal{S} of \mathcal{D} using any existing skyline algorithm, and initiate a \mathcal{K} containing an arbitrary point in \mathcal{S} . Given a point p , define its *representative distance* $rep-dist(p, \mathcal{K})$ as the distance between p and its closest representative, or formally:

$$rep-dist(p, \mathcal{K}) = \min_{p' \in \mathcal{K}} \|p, p'\|. \quad (7)$$

Then, we repeat the following $k-1$ times to create the final \mathcal{K} : add to \mathcal{K} the point in $\mathcal{S} - \mathcal{K}$ with the largest representative distance. Note that as the content of \mathcal{K} expands, the representative distance of a point may vary as well, because its nearest representative may change to the one most recently added. We refer to this solution as *naive-greedy*. It guarantees a 2-approximate solution, as is established directly by the analysis of [6].

As an example, consider the dataset in Figure 1, where $\mathcal{S} = \{p_1, p_2, \dots, p_8\}$. Assume $k = 3$ and that p_4 is the first point inserted to \mathcal{K} . Among the other skyline points, p_8 is farthest from p_4 , and thus, is the second point added to \mathcal{K} . Now, p_1 has the greatest representative distance in $\mathcal{S} - \mathcal{K}$, and hence, enters \mathcal{K} . Thus, the final result is $\mathcal{K} = \{p_4, p_8, p_1\}$.

Naive-greedy has several drawbacks. First, it incurs large I/O overhead because it requires retrieving the entire skyline \mathcal{S} . Since we aim at returning only $k \ll |\mathcal{S}|$ points, ideally we should be able to do so by accessing only a fraction of \mathcal{S} , thus saving considerable cost. Second, it lacks progressiveness, because no result can be output until the full skyline has been computed. In the next section, we will present an alternative algorithm called *I-greedy* which overcomes both drawbacks of *naive-greedy*.

B. I-greedy

I-greedy assumes a multidimensional index on the dataset \mathcal{D} . Although it can be integrated with many access methods

such as quad-trees, k-d trees, etc., next we use the R-tree [7] as an example due to its popularity and availability in practical DBMS.

I-greedy can be regarded as an efficient implementation of the *naive-greedy* algorithm explained in the previous subsection. Specifically, it returns the same set \mathcal{K} of representatives as *naive-greedy*. Therefore, *I-greedy* also has the same approximation ratio as *naive-greedy*.

Recall that, after the first representative, *naive-greedy* repetitively adds to \mathcal{K} the point in $\mathcal{S} - \mathcal{K}$ with the maximum representative distance given by Equation 7. Finding this point is analogous to *farthest neighbor search*, using Equation 7 as the distance function. However, remember that not every point in dataset \mathcal{D} can be considered as a candidate result. Instead, we consider only $\mathcal{S} - \mathcal{K}$, i.e., the set of skyline points still outside \mathcal{K} .

The *best-first* algorithm [8] is a well-known efficient algorithm for farthest neighbor search². To apply *best-first*, we must define the notion of *max-rep-dist*. Specifically, given an MBR R in the R-tree, its max-rep-dist, $max-rep-dist(R, \mathcal{K})$, is a value which upper bounds the representative distance $rep-dist(p, \mathcal{K})$ of any potential skyline point p in the subtree of R . We will discuss the computation of $max-rep-dist(R, \mathcal{K})$ in Section IV-C. Let us refer to both $max-rep-dist(R, \mathcal{K})$ and $rep-dist(p, \mathcal{K})$ as the *key* of R and p , respectively. *Best-first* visits the intermediate and leaf entries of the whole R-tree in descending order of their keys. Hence, the first leaf entry visited is guaranteed to be the point in \mathcal{D} with the largest representative distance.

Let p be the first data point returned by *best-first*. We cannot report p as a representative, unless we are sure that it is a skyline point. Whether p is a skyline point can be resolved using an *empty test*. Such a test checks if there is any data point inside the *anti-dominant region* of p , which is the rectangle having p and the origin of the data space as two opposite corners. If the test returns “empty”, p is a skyline point; otherwise, it is not. In any case, we continue the execution of *best-first* to retrieve the point with the next largest max-rep-dist, and repeat the above process, until enough representatives have been reported.

Best-first may still entail expensive I/O cost, as it performs numerous empty tests, each of which may need to visit many nodes whose MBRs intersect the anti-dominant region of a point. A better algorithm should therefore avoid empty tests as much as possible. *I-greedy* achieves this goal with two main ideas. First, it maintains a *conservative skyline* based on the intermediate and leaf entries already encountered. Second, it adopts an access order different from *best-first*, which *totally* eliminates empty tests.

Conservative skyline. Let S be a mixed set of α points and β rectangles. The conservative skyline of S is the skyline of a set S' with $\alpha + \beta \cdot d$ points, where d is the dimensionality of the data space. The set S' is generated as follows. First, it includes

²Precisely speaking, *best-first* is originally designed for nearest neighbor search [8]. However, its adaptation to farthest neighbor search is trivial.

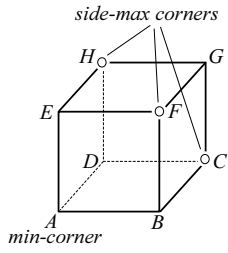


Fig. 7. The side-max corners

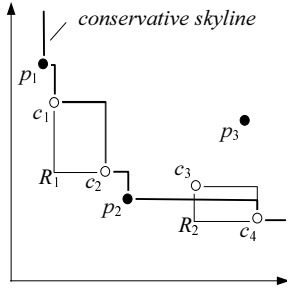


Fig. 8. Conservative skyline

all the α points of S . Second, for every rectangle $R \in S$, S' contains the d side-max corners of R . Specifically, note that R has $2d$ boundaries, each of which is a $(d-1)$ -dimensional rectangle. Among them, only d boundaries contain the min-corner of R , which is the corner of R closest to the origin. On each of those d boundaries, the corner opposite to the min-corner is a side-max corner. Figure 7 shows a 3D MBR whose min-corner is A . A is in 3 boundaries of R , i.e., rectangles $ADHE$, $ABCD$, $ABFE$. The side-max corners are H , C , and F , which are opposite to A in $ADHE$, $ABCD$, $ABFE$ respectively.

To illustrate conservative skyline, let S be the set of points p_1, p_2, p_3 and rectangles R_1, R_2 in Figure 8. The corresponding S' has 7 points $p_1, p_2, p_3, c_1, c_2, c_3, c_4$, noticing that c_1, c_2 are the side-max corners of R_1 , and c_3, c_4 are the side-max corners of R_2 . Hence, the skyline of S' is $\{p_1, c_1, c_2, p_2, c_3, c_4\}$, which is thus the conservative skyline of S .

To understand the usefulness of the conservative skyline, imagine R_1 and R_2 in Figure 8 as two MBRs in the R-tree. Clearly, the real skyline of p_1, p_2, p_3 and any points within R_1 and R_2 must be below the conservative skyline. Hence, if a point p is dominated by any point in the conservative skyline, p cannot appear in the real skyline. In that case, we do not need to issue an empty test for p .

Access order. Let \mathcal{L} be the set of intermediate and leaf entries that have been encountered and are waiting to be processed. We call \mathcal{L} the *access list*. \mathcal{L} is stored in memory. Recall that *best-first* chooses to process the entry in \mathcal{L} with the largest max-rep-dist. As explained earlier, this choice results in empty tests.

I-greedy adopts a different strategy. Let E be the entry in \mathcal{L} with the largest max-rep-dist. *I-greedy* checks whether there is any other intermediate or leaf entry in \mathcal{L} whose min-corner dominates the min-corner of E (as a special case, the min-corner of a point is just itself). If yes, among those entries I -

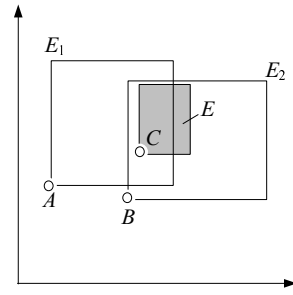


Fig. 9. Illustration of the access order of *I-greedy*

greedy processes the one E' whose min-corner has the smallest L_1 distance to the origin. If no, *I-greedy* processes E .

For example, assume that E is the shaded rectangle in Figure 9, and E_1 and E_2 are the only two entries in \mathcal{L} whose min-corners dominate the min-corner of E . As the min-corner of E_1 has a shorter L_1 distance to the origin, it is the entry to be processed by *I-greedy*.

To see why the access order makes sense, observe that it provides little gain in visiting E first in Figure 9. This is because even if we find any data point p in E , we must access E_1 or E_2 anyway to decide whether p is in the skyline. Hence, a better option is to open E_1 or E_2 first, which may allow us to obtain a tighter conservative skyline to prune E . Between E_1 and E_2 , E_1 is preferred, because it is more likely to include points or MBRs closer to the origin, which may result in a tighter conservative skyline.

Algorithm. We are ready to explain the details of *I-greedy*. It takes as an input an initial set \mathcal{K} containing an arbitrary skyline point p_{1st} . This point will be used as the first representative. For example, p_{1st} can be the point in \mathcal{D} with the smallest x-coordinate, which can be found efficiently in $O(\log_B |\mathcal{D}|)$ I/Os where B is the page size and $|\mathcal{D}|$ the cardinality of \mathcal{D} . *I-greedy* does not require a user to specify the number k of representatives to be returned. Instead, it continuously outputs representatives ensuring that, if so far it has produced t representatives, then they definitely make a 2-approximate solution, i.e., their representation error at most twice larger than that of the optimal size- t representative skyline.

At any moment, *I-greedy* maintains three structures in memory:

- the set \mathcal{K} of representatives found so far.
- an access list \mathcal{L} that contains all the intermediate and leaf entries that have been encountered but not processed or pruned yet.
- a conservative skyline \mathcal{S}_{con} of the set $\mathcal{L} \cup \mathcal{K}$.

Figure 10 presents the pseudocode of *I-greedy*. At the beginning, \mathcal{L} contains only the root entries of the R-tree. Next, *I-greedy* executes in iterations. In each iteration, it first identifies the entry E of \mathcal{L} with the largest max-rep-dist. Then, it checks whether (the min-corner of) E is dominated by any point in the conservative skyline \mathcal{S}_{con} . If yes, E is pruned, and the current iteration finishes.

Consider that E is not pruned, so the iteration continues. Following our earlier discussion on access order, *I-greedy*

Algorithm *I-greedy* (\mathcal{K})Input: a set \mathcal{K} with an arbitrary skyline point p_{1st}

Output: until stopped, continuously produce representatives

1. initiate \mathcal{L} to contain the root entries of the R-tree
 2. while \mathcal{L} is not empty
 3. E = the entry in \mathcal{L} with the largest max-rep-dist
 4. if E is not dominated by any point in \mathcal{S}_{con}
 5. E' = the entry with the minimum L_1 distance to the origin among all entries in \mathcal{L} whose min-corners dominate the min-corner of E
 6. if (E' exists)
 7. /* E' must be an intermediate entry */
 8. access the child node N of E'
 9. for each entry E_c in N
 10. if ($E_c \neq p_{1st}$) and (E_c is not dominated by any point in \mathcal{S}_{con})
 11. insert E_c in \mathcal{L}
 12. else /* E' does not exist */
 13. if E is a point p
 14. add p to \mathcal{K} and output p
 15. else
 16. access the child node N of E
 17. for each entry E_c in N
 18. if ($E_c \neq p_{1st}$) and (E_c is not dominated by any point in \mathcal{S}_{con})
 19. insert E_c in \mathcal{L}
-

Fig. 10. The *I-greedy* algorithm

looks for the entry E' with the smallest L_1 distance to the origin among all entries in \mathcal{L} whose min-corners dominate E . If E' exists, it must be an intermediate entry; otherwise, E' would be in the conservative skyline \mathcal{S}_{con} , and would have pruned E already. In this case, we visit the child node of E' , and insert its entries into \mathcal{L} that are not dominated by any point in the conservative skyline \mathcal{S}_{con} .

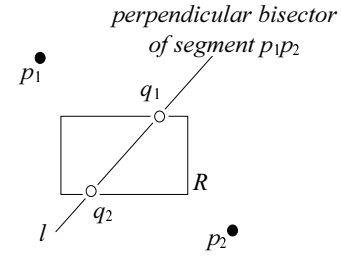
If E' does not exist, *I-greedy* processes E . If E is a point, it is inserted to \mathcal{K} and output as the next representative skyline point. Otherwise (E is an intermediate entry), we access its child node, and insert its entries in \mathcal{L} , if they are not dominated by any point in \mathcal{S}_{con} .

Recall that the *naive-greedy* algorithm in Section IV-A first extracts the entire skyline \mathcal{S} . Given an R-tree, the best way to do so is to apply the I/O optimal algorithm *BBS* [2]. Next, we show that *I-greedy* requires at most the same I/O overhead as *BBS*. In other words, *I-greedy* never entails higher I/O cost than *naive-greedy*.

LEMMA 5: When allowed to run continuously, *I-greedy* retrieves the whole skyline \mathcal{S} with the optimal I/O cost, i.e., same as *BBS*.

PROOF: It is obvious that *I-greedy* eventually computes the whole skyline, because it only prunes nodes whose min-corners are dominated by a point in the conservative skyline \mathcal{S}_{con} , and hence, cannot contain skyline points. Next, we will prove its I/O optimality.

As shown in [2], any R-tree-based skyline algorithm must access all nodes (whose min-corners are) not dominated by any skyline point. Assume that *I-greedy* is not I/O optimal,

Fig. 11. Finding the lowest value of $\text{max-rep-dist}(R, \mathcal{K})$

and accesses a node N dominated by a skyline point p . This access must happen at either Line 7 or 15 in Figure 10. In either case, when N is accessed, p or one of its ancestors must be in \mathcal{L} . Otherwise, p already appears in the representative set \mathcal{K} , and hence, would have pruned N .

As the min-corner of any ancestor of p dominates N , we can eliminate the possibility that N is visited at Line 15, because for this to happen E' at Line 5 must not exist, i.e., the min-corner of no entry in \mathcal{L} can dominate N . On the other hand, if N is visited at Line 7, N must have the lowest L_1 distance to the origin, among all entries in \mathcal{L} whose min-corners dominate the E at Line 2. This is impossible because (i) any E dominated by the min-corner of N is also dominated by p or the min-corner of any of its ancestors, and (ii) p or any of its ancestors has a smaller L_1 distance to the origin than N . \square

C. Computing the Maximum Representative Distance

This section will settle the only issue about *I-greedy* that has been left open. Namely, given the set \mathcal{K} of representatives already found, and an MBR R , we want to compute its $\text{max-rep-dist}(R, \mathcal{K})$, which must be at least the largest representative distance $\text{rep-dist}(p, \mathcal{K})$ of any point p in R .

To gain some insight about the smallest possible $\text{max-rep-dist}(R, \mathcal{K})$, let us consider a simple example where \mathcal{K} has only two points p_1 and p_2 , as shown in Figure 11. The figure also illustrates the perpendicular bisector l of the segment connecting p_1 and p_2 . For points p (i) above l , $\text{rep-dist}(p, \mathcal{K})$ equals $\|p, p_1\|$, (ii) below l , $\text{rep-dist}(p, \mathcal{K}) = \|p, p_2\|$, and (iii) on l , $\text{rep-dist}(p, \mathcal{K}) = \|p, p_1\| = \|p, p_2\|$. It is easy to see that, for points p in R , $\text{rep-dist}(p, \mathcal{K})$ is maximized when p is at the intersection q_1 or q_2 between l and the edges of R . In Figure 11, as $\text{rep-dist}(q_1, \mathcal{K}) < \text{rep-dist}(q_2, \mathcal{K})$, we know that $\text{max-rep-dist}(R, \mathcal{K}) = \text{rep-dist}(q_2, \mathcal{K})$.

The implication of the above analysis is that, in order to derive the lowest $\text{max-rep-dist}(R, \mathcal{K})$, we need to resort to the *Voronoi diagram* [9] of the points in \mathcal{K} . The Voronoi diagram [9] consists of a set of $|\mathcal{K}|$ polygonal cells, one for each point $p \in \mathcal{K}$, including all locations in the data space that has p as the closest representative. Unfortunately, Voronoi diagrams in dimensionality at least 3 are costly to compute. Furthermore, even if such a diagram was available, we still need to examine the intersection between perpendicular hyper-planes with the boundaries of R , which is challenging even in 3D space [9].

We circumvent the obstacle by finding a value for $\text{max-rep-dist}(R, \mathcal{K})$ that is low, although may not be the

lowest. Specifically, we set

$$\text{max-rep-dist}(R, \mathcal{K}) = \min_{p \in \mathcal{K}} \{\text{maxdist}(p, R)\}. \quad (8)$$

where $\text{maxdist}(p, R)$ is the maximum distance between a point p and a rectangle R . The next lemma shows that the equation gives a correct value of $\text{max-rep-dist}(R, \mathcal{K})$. Proof is omitted due to the space constraint.

LEMMA 6: The $\text{max-rep-dist}(R, \mathcal{K})$ from Equation 8 is at least as large as the $\text{rep-dist}(p, \mathcal{K})$ of any point p in R .

V. EXPERIMENTS

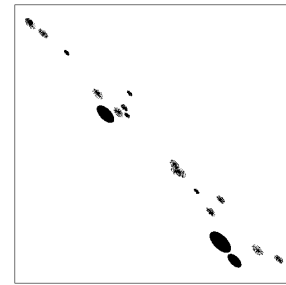
This section has two objectives. First, we will demonstrate that our distance-based representative skyline outperforms the previous method of max-dominance skyline [4] (reviewed in Section II) in two crucial aspects: our representative skyline (i) better captures the contour of the full skyline, and (ii) is much cheaper to compute. This will establish distance-based representative skyline as a more useful and practical skyline summary. Second, we will compare the efficiency of the proposed algorithms, and identify their strengths and shortcomings.

Data. Our experimentation is mainly based on a synthetic dataset *Island* and a real dataset *NBA*. *Island* is two-dimensional, and contains 63383 points whose distribution is shown in Figure 12a. These points form a number of clusters along the anti-diagonal of the data space, which simulates a common paradox where optimizing one dimension compromises the other. *Island* has 467 skyline points, whose distribution is given in Figure 12b.

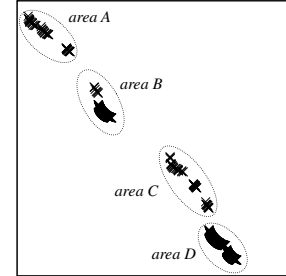
NBA is a real dataset which is downloadable at www.databasebasketball.com, and frequently adopted in the skyline literature [10], [11], [12]. It includes 17265 five-dimensional points, each recording the performance of a player on five attributes: the number of points scored, rebounds, assists, steals, and blocks, all of which are averaged over the total number of minutes played from 1950 to 1994. The skyline of *NBA* has 494 points.

Besides the above datasets, we also created *anti-correlated* datasets with various cardinalities and dimensionalities to test the algorithms' scalability. The anti-correlated distribution has become a benchmark in the skyline research [1], [2], [13], and is very suitable for scalability test because it has a fairly sizable skyline. Our generation of this distribution follows exactly the description in the seminal work of [1].

Representation quality. The first set of experiments utilizes dataset *Island* to assess how well a representative skyline reflects the contour of the full skyline. We examine three methods: *2d-opt*, *I-greedy*, and *2D-max-dom*. Specifically, *2d-opt* is our dynamic-programming algorithm in Section III that finds the optimal distance-based representative skyline. *I-greedy* is our 2-approximate algorithm in Section IV-B. *2d-max-dom* is the algorithm in [4] that computes the optimal max-dominance representative skyline.



(a) The *Island* dataset



(b) The skyline of *Island*

Fig. 12. Visualization of the *Island* dataset

As shown in Figure 12b, the skyline of *Island* can be divided into 4 *areas A, B, C, and D*. A good representative skyline should have representatives from every area, to provide the user with an adequate overview of the entire skyline. Figures 13a-13d illustrate the optimal distance-based representative skylines with size $k = 4, 6, 8,$ and 10 respectively returned by algorithm *2d-opt*. Clearly, in all cases, the representative skyline nicely indicates the shape of the full skyline. In particular, the representative skyline always includes a point in each of the four areas. Furthermore, the precision of representation improves as k increases.

Figures 13e-13h present the max-dominance representative skylines found by *2d-max-dom*. Unfortunately, the representative skyline never involves any point from areas *A* and *C*. To understand this, notice that as shown in Figure 12, both areas *B* and *D* have a very dense cluster. Therefore, from the perspective of max-dominance representative skyline, it is beneficial to put more representatives in these areas, since they are able to dominate more non-skyline points. Even at $k = 10$, the representative skyline still hardly provides a good summary of the entire skyline. Returning it to a user would create the misconception that no tradeoffs would be possible in areas *A* and *C*.

Finally, Figures 13i-13l depict the distance-based representative skylines produced by *I-greedy*. It is easy to see that although these representative skylines are different from those by *2d-opt*, they also capture the contour of the full skyline. In fact, starting from $k = 6$, the representative skylines by *I-greedy* and *2d-opt* already look very similar. It is worth noting that our other approximate algorithms, *naive-greedy* and *best-first* discussed in Sections IV-A and IV-B respectively, output exactly the same result as *I-greedy*.

Figure 14a shows the ratio between the representation error of *I-greedy* (*2d-max-dom*) and that of the optimal algorithm

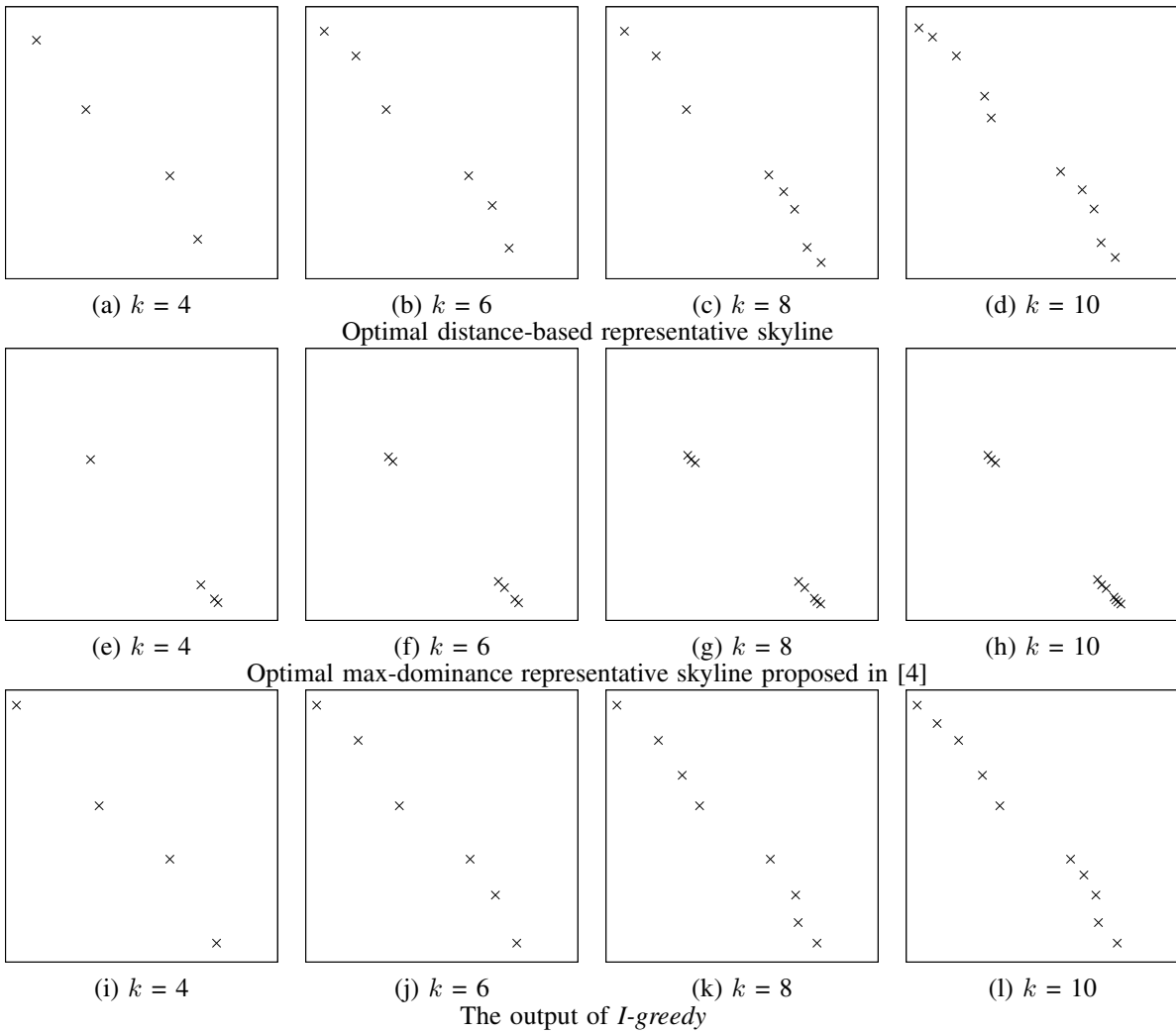


Fig. 13. Representative skylines

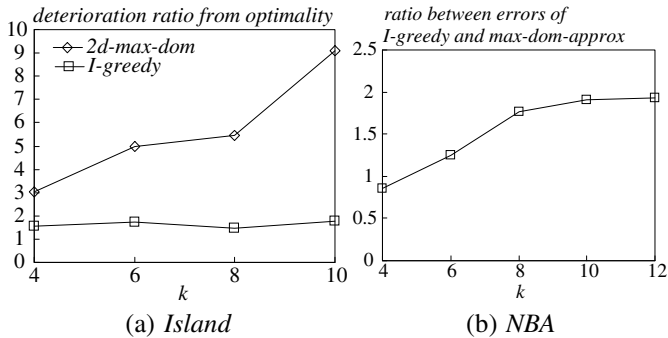


Fig. 14. Representation error comparison

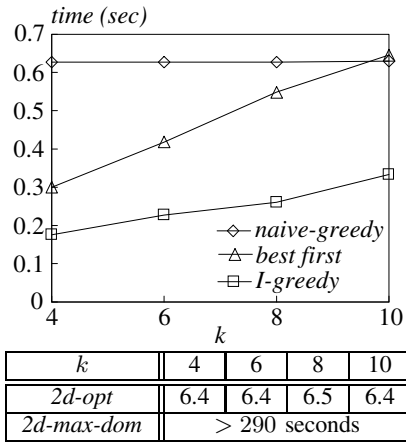
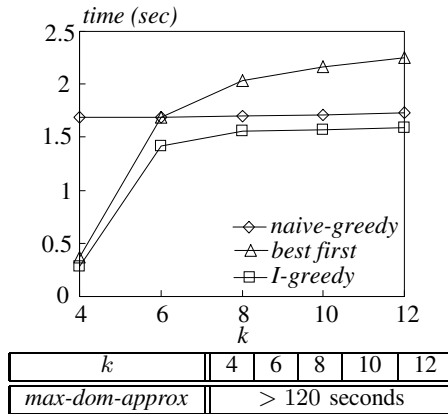
2d-opt in the experiments of Figure 13. Recall that the representation error is given by Equation 1. As expected, the ratio of *I-greedy* never exceeds 2, because *I-greedy* is guaranteed to yield a 2-approximate solution. The ratio of *2d-max-dom*, however, is unbounded and escalates quickly with k .

The next experiment inspects the representation error on dataset *NBA*. We again examine *I-greedy* but discard *2d-opt* and *2d-max-dom* because they are restricted to dimensionality

2. Instead, we compare *I-greedy* against *max-dom-approx*, which is an algorithm in [4] that returns a max-dominance skyline with theoretical guarantees in any dimensionality. Figure 14b plots the ratio between the error of *max-dom-approx* and *I-greedy* as k varies from 4 to 12. It is clear that the relative accuracy of *I-greedy* continuously increases as more representatives are returned.

Efficiency. We now proceed to study the running time of representative skyline algorithms: *2d-opt*, *naive-greedy*, *best-first*, *I-greedy*, *2d-max-dom*, and *max-dom-approx*. Recall that, as mentioned in Section IV-B, *best-first* extends the traditional *best-first* algorithm for farthest nearest neighbor search with empty tests. Furthermore, algorithms *2d-opt* and *2d-max-dom* are applicable to 2D datasets only. We index each dataset using an R-tree with 4k page size, and deploy the tree to run the above algorithms. All the following experiments are performed on a machine with an Intel dual-core 1GHz CPU.

Figure 15a illustrates the execution time of alternative algorithms on dataset *Island* as a function of k . Note that *2d-max-dom* takes nearly 5 minutes. It is almost 50 times slower than *2d-opt*, and over 300 times slower than *naive-greedy*, *best-first*,

(a) *Island*(b) *NBA*Fig. 15. Running time vs. k

and *I-greedy*. This indicates that distance-based representative skyline is indeed much cheaper to compute than the max-dominance version. Among the proposed algorithms, *2d-opt* is most costly because it performs dynamic programming to find the optimal solution, while the other algorithms provide only approximate solutions. As expected, the running time of *naive-greedy* is not affected by k , because it is dominated by the cost of retrieving the full skyline. The overhead of both *best-first* and *I-greedy* grows with k . While the performance of *best-first* deteriorates rapidly, *I-greedy* remains the most efficient algorithm in all cases.

Figure 15b presents the results of the same experiment on dataset *NBA*. Note that we leave out the 2D algorithm *2d-opt* and replace *2d-max-dom* with *max-dom-approx*. Again, it incurs considerably larger cost to calculate max-dominance representative skylines than distance-based ones. The behavior of *naive-greedy*, *best-first*, and *I-greedy* is identical to Figure 15a, except that *best-first* becomes slower than *naive-greedy* after $k = 6$.

To further analyze *naive-greedy*, *best-first*, and *I-greedy*, in Tables Ia and Ib, we provide their detailed I/O and CPU time in the experiments of Figures 15a and 15b respectively. In each cell of the tables, the value outside the bracket is the I/O cost in number of page accesses, and the value inside is the CPU time in seconds. Observe that *I-greedy* requires the

k	4	6	8	10
<i>naive-greedy</i>	54 (.09)	54 (.09)	54 (.09)	54 (.09)
<i>best-first</i>	24 (.06)	33 (.09)	42 (.13)	50 (.15)
<i>I-greedy</i>	10 (.18)	12 (.23)	14 (.26)	17 (.33)

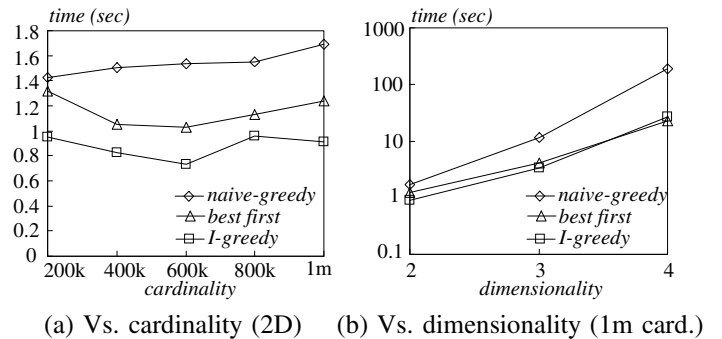
(a) *Island*

k	4	6	8	10	12
<i>naive-greedy</i>	156 (.13)	156 (.13)	156 (.14)	156 (.15)	156 (.18)
<i>best-first</i>	21 (.16)	86 (.83)	94 (1.1)	104 (1.1)	113 (1.1)
<i>I-greedy</i>	12 (.16)	70 (.72)	72 (.84)	73 (.84)	74 (.85)

(b) *NBA*

TABLE I

BREAKDOWNS OF RUNNING TIME. FORMAT: I/O (CPU)



(a) Vs. cardinality (2D)

(b) Vs. dimensionality (1m card.)

Fig. 16. Scalability comparison (anti-correlated)

fewest I/Os, but as a tradeoff, consumes higher CPU time. This is expected because while the access order of *I-greedy* reduces I/Os, enforcing it demands additional computation. *Naive-greedy* is exactly the opposite by entailing the most I/Os and the least CPU power. *Best-first* appears to be a compromise for the 2D dataset *Island*. For the 5D *NBA*, however, *best-first* is worse than *I-greedy* in both I/O and CPU from $k = 8$.

The last set of experiments investigates the scalability of *naive-greedy*, *best-first*, and *I-greedy* with respect to the dataset cardinality and dimensionality. For this purpose, we fix k to 10. Using 2D anti-correlated datasets, Figure 16a plots the overall execution time of the three algorithms as the cardinality grows from 200k to 1 million. At these cardinalities, the skyline has 339, 385, 379, 390, and 376 points, respectively. The performance of all three algorithms exhibits no significant changes. This is consistent with the finding of [2] that the cardinality of an anti-correlated dataset does not have heavy influence on the cost of skyline retrieval.

Next, we fix the cardinality to 1 million. Figure 16b compares the three algorithms with anti-correlated datasets of dimensionalities 2 to 4. The corresponding skyline size equals 376, 2719, and 30663 respectively. All algorithms entail higher overhead because skyline search is in general more difficult in higher dimensionality. Note that at dimensionalities 3 and 4, *naive-greedy* is by far the worst method, because it must spend gigantic cost in fetching the entire skyline. *Best-first* and *I-greedy* avoid such cost by obtaining the representatives directly without extracting the full skyline.

VI. RELATED WORK

The first work on skylines in the database area is due to Borzsonyi et al. [1]. Since then, many algorithms have been developed for computing skylines efficiently, for example, *Bitmap* [13], *NN* [14], *BBS* [2], *Lattice* [15], to name just a few. These algorithms focus on the original data space, while considerable efforts have also been made to retrieve skylines in subspaces, such as *subsky* [10], *skycube* [11], and so on. Besides traditional centralized DBMS, skyline search has also been studied in distributed systems [16], [17], streams [18], p2p networks [19], partially-ordered domains [20], etc.

The concept of skyline has numerous useful variations. One example is the *representative skyline* studied in this paper, and this notion is originally introduced in [4]. Other examples include *k-dominant skyline* [3], *spatial skyline* [21], *probabilistic skyline* [12], *reverse skyline* [22], [23], *privacy skyline* [24], *approximately dominating representatives* [25], retrieving the points with the highest *subspace skyline frequencies* [26], and so on.

The *best-first* algorithm mentioned in Section IV-B is proposed by Hjaltason and Samet [8] for solving nearest/farthest neighbor search. It is I/O optimal in the sense that, given the same R-tree, no other algorithm is able to answer the same query by accessing fewer nodes.

The *k-center* problem, which underlines the proposed distance-based representative skyline, is a classical problem that can be defined on any distance metric. Without knowing the metric, it is NP-hard to find a solution with approximation ratio $2 - \varepsilon$ for any positive ε [27]. The greedy algorithm described in Section IV-A provides a 2-approximate solution for any distance metric satisfying the triangle inequality [6].

VII. CONCLUSIONS

The skyline of a dataset may have a large number of points. Returning all of them may make it difficult for a user to understand the possible tradeoffs offered by the skyline. A better approach is to present only a few representative points that reflect the contour of the entire skyline, so that the user may request only the skyline points in a specific part of the contour that looks interesting.

The only existing formulation of representative skyline [4] sometimes cannot capture the skyline contour accurately. Motivated by this, we propose the distance-based representative skyline, and prove that it has much better worst-case guarantees in representation quality than the definition of [4]. We also develop several algorithms for computing distance-based representative skylines. In 2D space, our technique finds the optimal solution in low-degree polynomial time. In higher dimensional spaces where computing the optimal solution is NP-hard, our technique returns a 2-approximate solution efficiently. Our extensive experimentation shows that distance-based representative skylines not only better capture the contour of the full skyline, but also can be computed in a fraction of the time required by the previous approach in [4].

ACKNOWLEDGEMENTS

Yufei Tao and Ling Ding were supported by Grants CUHK 1202/06 and CUHK 4161/07 from HKRGC. Jian Pei was supported in part by an NSERC Discovery Grant and an Discovery Accelerator Supplements Grant. Xuemin Lin was supported by three ARC DPs (DP0666428, DP0881035, and DP0987557) and a Google research award.

REFERENCES

- [1] S. Borzsonyi, D. Kossmann, and K. Stocker, "The skyline operator," in *ICDE*, 2001, pp. 421–430.
- [2] D. Papadias, Y. Tao, G. Fu, and B. Seeger, "An optimal and progressive algorithm for skyline queries," in *SIGMOD*, 2003, pp. 467–478.
- [3] 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, pp. 503–514.
- [4] X. Lin, Y. Yuan, Q. Zhang, and Y. Zhang, "Selecting stars: The *k* most representative skyline operator," in *ICDE*, 2007, pp. 86–95.
- [5] T. F. Gonzalez, "Clustering to minimize the maximum intercluster distance," *Theor. Comput. Sci.*, vol. 38, pp. 293–306, 1985.
- [6] T. Feder and D. Greene, "Optimal algorithms for approximate clustering," in *STOC*, 1988, pp. 434–444.
- [7] N. Beckmann, H. Kriegel, R. Schneider, and B. Seeger, "The R*-tree: An efficient and robust access method for points and rectangles," in *SIGMOD*, 1990, pp. 322–331.
- [8] G. R. Hjaltason and H. Samet, "Distance browsing in spatial databases," *TODS*, vol. 24, no. 2, pp. 265–318, 1999.
- [9] M. de Berg, O. Cheong, M. van Kreveld, and M. Overmars, *Computational Geometry: Algorithms and Applications*. Springer-Verlag, 2008.
- [10] Y. Tao, X. Xiao, and J. Pei, "Subsky: Efficient computation of skylines in subspaces," in *ICDE*, 2006.
- [11] J. Pei, W. Jin, M. Ester, and Y. Tao, "Catching the best views of skyline: A semantic approach based on decisive subspaces," in *VLDB*, 2005, pp. 253–264.
- [12] J. Pei, B. Jiang, X. Lin, and Y. Yuan, "Probabilistic skylines on uncertain data," in *VLDB*, 2007, pp. 15–26.
- [13] K.-L. Tan, P.-K. Eng, and B. C. Ooi, "Efficient progressive skyline computation," in *VLDB*, 2001, pp. 301–310.
- [14] D. Kossmann, F. Ramsak, and S. Rost, "Shooting stars in the sky: An online algorithm for skyline queries," in *VLDB*, 2002, pp. 275–286.
- [15] M. Morse, J. M. Patel, and H. V. Jagadish, "Efficient skyline computation over low-cardinality domains," in *VLDB*, 2007, pp. 267–278.
- [16] B. Cui, H. Lu, Q. Xu, L. Chen, Y. Dai, and Y. Zhou, "Parallel distributed processing of constrained skyline queries by filtering," in *ICDE*, 2008, pp. 546–555.
- [17] A. Vlachou, C. Doukeridis, and Y. Kotidis, "Angle-based space partitioning for efficient parallel skyline computation," in *SIGMOD*, 2008, pp. 227–238.
- [18] N. Sarkas, G. Das, N. Koudas, and A. K. H. Tung, "Categorical skylines for streaming data," in *SIGMOD*, 2008, pp. 239–250.
- [19] S. Wang, B. C. Ooi, A. K. H. Tung, and L. Xu, "Efficient skyline query processing on peer-to-peer networks," in *ICDE*, 2007, pp. 1126–1135.
- [20] C. Y. Chan, P.-K. Eng, and K.-L. Tan, "Stratified computation of skylines with partially-ordered domains," in *SIGMOD*, 2005, pp. 203–214.
- [21] M. Sharifzadeh and C. Shahabi, "The spatial skyline queries," in *VLDB*, 2006, pp. 751–762.
- [22] E. Dellis and B. Seeger, "Efficient computation of reverse skyline queries," in *VLDB*, 2007, pp. 291–302.
- [23] X. Lian and L. Chen, "Monochromatic and bichromatic reverse skyline search over uncertain databases," in *SIGMOD*, 2008, pp. 213–226.
- [24] B.-C. Chen, R. Ramakrishnan, and K. LeFevre, "Privacy skyline: Privacy with multidimensional adversarial knowledge," in *VLDB*, 2007, pp. 770–781.
- [25] V. Koltun and C. H. Papadimitriou, "Approximately dominating representatives," in *ICDT*, 2005, pp. 204–214.
- [26] 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.
- [27] W. L. Hsu and G. L. Nemhauser, "Easy and hard bottleneck location problems," *Disc. Appl. Math.*, vol. 1, pp. 209–216, 1979.