On Representing Skylines by Distance*

Yufei Tao
Chinese University of Hong Kong
Hong Kong
taoyf@cse.cuhk.edu.hk

Jian Li
Tsinghua University
China
lijian83@mail.tsinghua.edu.cn

Ling Ding
University of Pennsylvania
USA
lingding@cis.upenn.edu

Xuemin Lin
University of New South Wales
Australia
lxue@cse.unsw.edu.au

Jian Pei
Simon Fraser University
Canada
jpei@cs.sfu.ca

Abstract

Given an integer k, a *representative skyline* contains the k skyline points that best describe the trade-offs among different dimensions offered by the full skyline. This paper proposes a *distance-based* formulation, which aims at minimizing the distance between a non-representative skyline point and its nearest representative. In 2D space, there is a dynamic-programming algorithm for computing an optimal representative skyline, whereas for dimensionality at least 3, we prove that the problem is NP-hard, and give a 2-approximate polynomial-time solution. The effectiveness and efficiency of our techniques have been confirmed by extensive experimentation.

*A short version of this paper has appeared in ICDE'09.

1 Introduction

Given a set $\mathcal{D}$ of multidimensional points, the *skyline* [2] 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, ..., p_8.

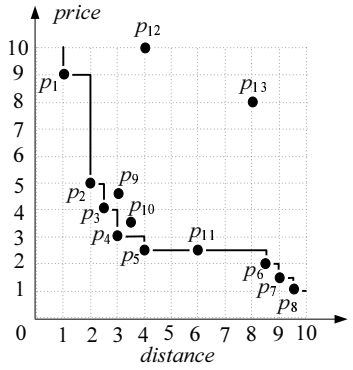


Figure 1: A skyline example

Skyline retrieval has received considerable attention from the database community, resulting in a large number of interesting results as surveyed in Section 6. These research efforts reflect the crucial importance of skylines in practice. In particular, it is well-known [21] 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 relative weighting 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 computing a *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

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

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 [4]. In fact, even a two-dimensional dataset may have a sizable skyline, when the data distribution is "anti-correlated" [2]. Presenting a user with a large result set elicits 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 [18] on representative skylines. The work of [18], 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 [18] advocates a representative set of $\{p_3, p_4, p_5\}$ in the example of Figure 1. This would be 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, given the (full) skyline, finds an optimal solution to the problem in $O(mk)$ time, where m is the number of points in the skyline. For dimensionality at least 3, we show that the problem is NP-hard, and give a 2-approximate polynomial-time 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 [18], our representative skyline not only better captures the contour of the full skyline, but also can be computed considerably faster.

A short version of this work has appeared in [28]. The current paper extends that preliminary report with two new contributions:

- The first one is a new algorithm for finding an optimal solution in 2d space which, as mentioned earlier, uses $O(mk)$ time (Section 3.3). This is a dramatic improvement over the best algorithm in [28], which demands $O(m^2k)$ time.

- The second new contribution is the establishment of a crucial property (Lemma 4) that proves the effectiveness of distance-based representative skylines in any fixed-dimensionality d. This property was proved only in 2d space in [28]; since then, proving the property for $d \geq 3$ has been an urgent open issue because without settling it the effectiveness of our approach has remained *unjustified* for those dimensionalities! This work formally closes the issue. As we demonstrate in the proof of Lemma 4, it turns out that the $d \geq 3$ case requires an argument completely different from the one in [28]. The discovery of that argument also serves as a technical breakthrough since [28].

The rest of the paper is organized as follows. Section 2 clarifies our formulation of representative

skyline, analyzes its properties, and points out its advantages over [18]. Section 3 presents an algorithm for finding an optimal representative skyline in 2D space. Section 4 tackles dimensionalities at least 3. Section 5 evaluates the proposed techniques with extensive experiments. Section 6 surveys the previous literature on skyline. Finally, Section 7 concludes the paper with a summary of our results.

2 Representative skylines and properties

Let $\mathcal{D}$ be a set of d-dimensional points. Without loss of generality, we often consider that each coordinate of every point in $\mathcal{D}$ has been normalized into a unit range $[0, 1]$. Refer to the box $[0, 1]^d$ as the *data space*. Denote by $\mathcal{S}$ the full skyline of $\mathcal{D}$. In the sequel, we first review the only existing definition of representative skyline proposed in [18], and elaborate its defects. Then, we propose our definition, and explain its superiority over the proposition in [18]. Finally, we identify the underlying optimization problem.

2.1 Defects of the existing formulation

Given an integer k, Lin et al. [18] define a 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 1, $\mathcal{K} = \{p_3, p_4, p_5\}$ 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 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 arbitrarily 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, ..., 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, ..., 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 ($1 \leq i \leq k$), place another point p_{k+i} that is exclusively dominated

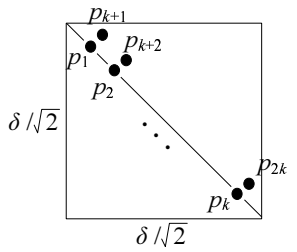


Figure 2: A gadget in the proof of Lemma 1

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, ..., p_{2k+1} constitute $\mathcal{D}$.

The max-dominance representative skyline $\mathcal{K}$ of $\mathcal{D}$ includes $\{p_1, p_2, ..., p_k\}$, 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. □

Recall that $\sqrt{2}$ is the maximum possible distance of two data 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 $O(n \log m + m^2 k)$ time [18] to find an optimal solution, where n and m are the sizes of $\mathcal{D}$ 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 [18] finds a solution with provably good quality guarantees in $O(nm)$ time. Apparently, this is prohibitive for large datasets. Acknowledging the vast cost complexity, the authors of [18] also provide another heuristic algorithm that is faster but may return a solution with poor quality.

For fairness, we should point out that the defects of max-dominance representative skylines are valid only if the objective is to summarize the spatial distribution of the skyline points (i.e., the possible tradeoffs along the skyline), which is the focus of this paper as explained in Section 1. In fact, the max-dominance definition was designed to achieve different purposes, which have been nicely explained in [18].

2.2 Our formulation

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

Definition 2. *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.*

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 an optimal solution of the k-*center problem* [13] on the full skyline $\mathcal{S}$. As a result, the distance-based representative skyline shares several properties of the k-center problem. 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

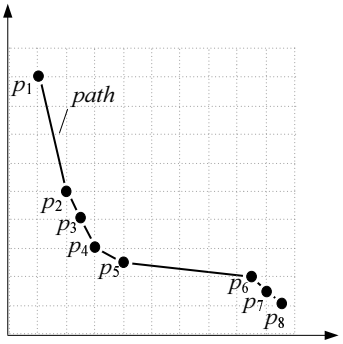


Figure 3: A path in the proof of Lemma 3

representatives to a cluster simply because it has a large density. 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 outweight the benefit of trying to reduce such distances of points in a faraway sparse cluster.

representative will go into this cluster to refine its precision. Indeed, the distance-based representative skyline with $k = 4$ is $\mathcal{K} = \{p_1, p_3, p_4, p_7\}$.

Our formulation of representative skyline enjoys an attractive theoretical guarantee:

Lemma 3. *For any $k \geq 2$ and any 2D dataset $\mathcal{D}$ with cardinality at least k, the representation error $Er(\mathcal{K})$ of a 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, ..., 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. 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. □

Comparing Lemmas 1 and 3, 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).

A result analogous to Lemma 3 can also be established in higher dimensionalities, but through a drastically different approach:

Lemma 4. *For any $k \geq 2$ and any dataset $\mathcal{D}$ (with $|\mathcal{D}| \geq k$) of a fixed dimensionality d, the representation error $Er(\mathcal{K})$ of a distance-based representative skyline $\mathcal{K}$ is $O((\frac{1}{k})^{1/(d-1)})$.*

Proof. For simplicity, we assume that $\mathcal{D}$ is in *general position*, such that all pairs of points have distinct distances. This assumption can be easily removed by extending our proof with details obstructing the central ideas. Further, we consider that $\mathcal{S}$ has more than k points; otherwise, the lemma is trivially true. We will find a set $\mathcal{K}'$ of k skyline points such that $Er(\mathcal{K}') = O((\frac{1}{k})^{1/d})$. This will establish the lemma because $Er(\mathcal{K}')$ bounds $Er(\mathcal{K})$ from above.

Given a point p, define its *representative distance* *rep-dist*$(p, \mathcal{K}')$ as the distance between p and its closest representative, or formally:

$$\textit{rep-dist}(p, \mathcal{K}') = \min_{p' \in \mathcal{K}'} \|p, p'\|. \tag{2}$$

We construct $\mathcal{K}'$ by repeating the following $k-1$ times, starting from a $\mathcal{K}'$ containing the point with the smalllest coordinate in $\mathcal{S}$:

> add to $\mathcal{K}'$ the point in $\mathcal{S} - \mathcal{K}'$ with the largest representative distance, i.e., the point determining the current $Er(\mathcal{K}')$.

Note that as the content of $\mathcal{K}'$ expands, the representative distance of a point may decrease, because its nearest representative may change to the one most recently added. Set $r = Er(\mathcal{K}')$. We will show that the $\mathcal{K}'$ has the desired property that $r = O((\frac{1}{k})^{1/d})$.

Given two d-dimensional circles, we say that they are *non-eclipsing* if neither circle covers the center of the other. In other words, either the two circles are disjoint, or their intersection does not contain the center of any circle. Two circles are said to be *eclipsing* otherwise.

Call the points in $\mathcal{K}'$ *representatives*, and denote them as $c_1, ..., c_k$, in the order they are added to $\mathcal{K}'$. Focusing on any particular c_i ($i \leq k$), we say that a point $p \in \mathcal{S}$ is *represented* by c_i, if c_i is the closest to p among all the points in $\mathcal{K}'$. Associate c_i with the circle C_i that centers at c_i and has radius r. Below we show that $C_1, ..., C_k$ are mutually non-eclipsing with an inductive argument on k. Let us start with two facts:

$F1$ Our strategy for generating $\mathcal{K}'$ guarantees that if a point is a representative for $k = j$, it is also a representative for $k = j+1$. In other words, as k increases, $\mathcal{K}'$ only expands; no existing representative will be left out.

$F2$ The circles associated with the representatives shrink continuously as k increases, noticing that $r = Er(\mathcal{K}')$ decreases whenever $\mathcal{K}'$ takes in an extra point.

We are ready to elaborate on our inductive proof. As the basic step, we show that C_1 and C_2 are non-eclipsing for $k = 2$. Suppose for contradiction that they are not, in which case two possibilities could happen. First, C_1 would pass a point $p \notin \mathcal{K}'$ represented by c_1 such that $\|p, c_1\| > \|c_1, c_2\|$. This cannot happen due to the choice of c_2. Second, C_2 would pass a point $p \notin \mathcal{K}'$ (represented by c_2) satisfying $\|p, c_2\| > \|c_1, c_2\|$. However, the fact that p is represented by c_2 indicates that $\|p, c_1\| > \|p, c_2\| > \|c_1, c_2\|$, which, once again, violates how c_2 was decided.

Assuming that $C_1, ..., C_k$ are mutually non-eclipsing for $k = j$, we prove that this is also true for $k = j + 1$. Since, by $F2$, $C_1, ..., C_j$ have become smaller (compared to what they were when $k = j$), our inductive assumption implies that these j circles must still be mutually non-eclipsing. Hence, it suffices to show that each of them is also non-eclipsing with C_{j+1}.

Suppose for contradiction that C_i (for some $i \in [1, j]$) is eclipsing with C_{j+1}. To explain that this cannot happen, let p be the point defining r currently, that is, *rep-dist*$(p, \mathcal{K}') = r$. Define c_α as the nearest representative for p in $\mathcal{K}' \setminus \{c_{j+1}\}$, namely, the representative that represented p when $k = j$. Similarly, denote by c_β the nearest representative for c_{j+1} in $\mathcal{K}' \setminus \{c_{j+1}\}$. It thus holds that

$$\|p, c_\alpha\| \geq r > \|c_i, c_{j+1}\| \geq \|c_\beta, c_{j+1}\|.$$

This contradicts the selection of c_{j+1}.

Equipped with the above result, we proceed to prove Lemma 4. We will first illustrate the rationale by establishing a weaker error bound of $O((\frac{1}{k})^{1/d})$. Given a circle C, let us define its *miniature* to be the circle having the same center as C but a radius half that of C. Observe that if two circles C, C' are non-eclipsing, their miniatures *must* be disjoint. Each of $C_1, ..., C_k$ has radius at most $\sqrt{d}$. Therefore, the miniatures of $C_1, ..., C_k$ are k disjoint circles in a d-dimensional square of side length $1 + \sqrt{d}$. The sqaure has a volume of $(1 + \sqrt{d})^d$, forcing each of the k miniature circles to have a volume of $O(1/k)$. Therefore, every miniature has a radius $O((\frac{1}{k})^{1/d})$, implying the same for each C_i.

To tighten the bound to $O((\frac{1}{k})^{1/(d-1)})$, let us first point out a geometric fact. Given a d-dimensional point p, define $dom(p)$ as the region in the d-dimensional universe that is dominated by p. Note that $dom(p)$ extends beyond the data space. Given the skyline $\mathcal{S}$, consider the union of the $dom(p)$ of all the skyline points $p \in \mathcal{S}$, and denote the union as $dom(\mathcal{S})$. Let B be the boundary of $dom(\mathcal{S})$. Notice that every facet of B is part of a $d - 1$ dimensional axis-parallel plane. The geometric fact we need is:

$F3$ Let p be a point in $\mathcal{S}$. The intersection of B and any *solid circle* C, which centers at p and has radius r, has a $d - 1$ dimensional volume of $\Omega(r^{d-1})$. Note that a solid circle includes all points on the boundary and in the interior.

To verify this fact, let us focus on the inscribed (d-dimensional) solid square s of C, where "solid" has the same meaning as in solid circle. Apparently, B has a side length of $r/\sqrt{d}$. We will show that the $d - 1$ dimensional volumn of $s \cap B$ is $\Omega(r^{d-1})$, which is sufficient for proving $F3$ because $s \cap B$ obviously has a smaller volume than $C \cap B$. As mentioned earlier, every facet of B is axis-parallel. Let us project both s and B onto the first $d - 1$ dimensions of the universe. The ($d - 1$ dimensional) volume of $s \cap B$ is at least that of the intersection of the two projections. It is easy to see that the latter intersection has volume at least $(\frac{r}{2\sqrt{d}})^{d-1} = \Omega(r^{d-1})$ (in fact, the volume is the smallest if p is the only point in $\mathcal{S}$), which establishes $F3$.

Finally, we are ready to prove the error bound $O((\frac{1}{k})^{1/(d-1)})$. Consider the miniatures of $C_1, ..., C_k$ defined earlier. As explained before, these miniatures are mutually disjoint, and hence, their intersections with B must also be mutually disjoint. In other words, by $F3$, the union of all those intersections has a volume $\Omega(k(r/2)^{d-1})$. On the other hand, that union must be inside a d-dimensional square with side length $1 + \sqrt{d}$. In other words, $k(r/2)^{d-1} = O(1)$, indicating that $r = O((\frac{1}{k})^{1/(d-1)})$. □

The above lemma proves that, in general d-dimensional space, our formulation of representative skylines still enjoys the property that the representation error monotonically diminishes as the space budget k increases, a property that max-dominance representative skylines do not have (as explained in Section 2.1).

2.3 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:

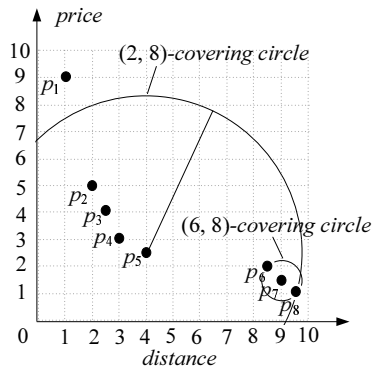


Figure 4: Covering circles

Problem 5. *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.*

Sometimes it is computationally intractable to find an optimal solution. In this case, we instead aim at computing a representative skyline whose representation error is as low as possible.

3 The two-dimensional case

In this section, we discuss how to solve Problem 5 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, ..., p_m, sorted in ascending order of their x-coordinates. We adopt the notation $\mathcal{S}_i$ to represent $\{p_1, p_2, ..., p_i\}$ where $i \leq m$, with $\mathcal{S}_0 = \emptyset$.

3.1 The first algorithm

Introduce a function $opt(i, t)$ to be an optimal size-t representative skyline of $\mathcal{S}_i$, where $t \leq i$. Hence, the goal of Problem 5 is to compute $opt(m, k)$. 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

- covers points p_i, p_{i+1}, ..., p_j, and
- centers at one of these $j - i + 1$ points.

We refer to the above circle as the (i, j)-*covering circle*, and denote its center as $center(i, j)$. For example, for 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.

Lemma 6. *For $t \geq 2$,*

$$optEr(i, t) \quad = \quad \min_{j=t}^{i} \{\max\{optEr(j-1, t-1), radius(j, i)\}\} \tag{3}$$

Proof. We will first establish a useful property named *continuity of covering*:

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
4. for $i = t$ to m
5. compute $optEr(i, t)$ by Equation 3
6. compute $opt(i, t)$ by Equation 4
7. return $opt(k, m)$

Figure 5: An optimal algorithm for computing 2D representative skylines

> Let x, y be two fixed integers in $[1, m]$ such that $x < y$. Define u as the smallest integer in $(x, y]$ such that $\|p_x, p_u\| > \|p_u, p_y\|$. Then, it holds that (i) $\|p_x, p_v\| > \|p_v, p_y\|$ for any $v > u$, and (ii) $\|p_x, p_w\| < \|p_w, p_y\|$ for any $w < u$.

Alternatively, u can be understood as the first point that comes closer to p_y than to p_x, as we walk along $\mathcal{S}$ from p_x to p_y. We will prove only part (i), because a similar argument works for part (ii). For $v < y$, observe that, on each dimension, the coordinate difference between p_v and p_y (respectively, p_x) is smaller (larger) than that of p_u and p_y (p_x). Hence:

$$\|p_x, p_v\| > \|p_x, p_u\| > \|p_u, p_y\| > \|p_v, p_y\|.$$

For $v > y$, on each dimension, the coordinate difference between p_v and p_y is always lower than that of p_v and p_x, indicating immediately $\|p_x, p_v\| > \|p_v, p_y\|$.

Now let us get back to Equation 3. Suppose that an optimal size-t representative skyline of $\mathcal{S}_i$ is $\{p_{j_1}, p_{j_2}, ..., p_{j_t}\}$ with $1 \leq j_1 < j_2 < ... < j_t \leq i$. 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. By continuity of covering, p_{j_t} must be the nearest representative for all p_v satisfying $j \leq v \leq i$, and *cannot* be the nearest representative for any p_w with $w < j$. It follows that, $\{p_{j_1}, ..., p_{j_{t-1}}\}$ must be an optimal size-$(t-1)$ representative skyline of $\mathcal{S}_{j-1}$, and p_{j_t} must be an optimal size-1 representative skyline of $\{p_j, p_{j+1}, ..., p_i\}$, namely, $p_{j_t} = center(j, i)$. □

Let v be the value of j where Equation 3 reaches its minimum; we have:

$$opt(i, t) = opt(v - 1, t - 1) \cup \{center(v, i)\} \tag{4}$$

Equations 3 and 4 point to a dynamic programming algorithm *2d-opt* in Figure 5 for computing $opt(k, m)$, i.e., the size-k representative skyline of $\mathcal{D}$.

As explained in the next subsection, Line 1 of *2d-opt* can be implemented in $O(m^2)$ time. Line 2 obviously requires $O(m)$ time. Lines 3-6 perform $k - 1$ *iterations*. Each iteration evaluates Equations 3 and 4 at most m times respectively. Regardless of i and t, every evaluation of Equation 3 can be completed in $O(m)$ time, and that of Equation 4 in $O(k)$ time. Hence, Lines 3-6 altogether incur $O(m^2k)$ cost. Therefore, the overall complexity of *2d-opt* is $O(m^2k)$. Note that this is much lower than the complexity $O(n \log m + m^2k)$, as mentioned in Section 3, of computing an optimal 2D max-dominance skyline.

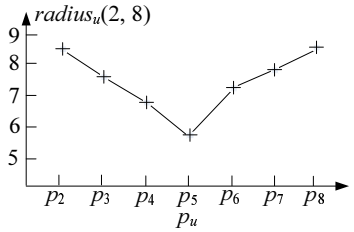


Figure 6: Plot of $radius_u(2, 8)$ for $2 \leq u \leq 8$

3.2 Covering circle computation

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 $1 \leq i \leq j \leq m$. First, it is easy to see that

$$radius(i, j) = \min_{u=i}^{j} \{\max\{\|p_i, p_u\|, \|p_u, p_j\|\}\}. \tag{5}$$

Given i, j, u satisfying $i \leq u \leq j$, we define

$$radius_u(i, j) = \max\{\|p_i, p_u\|, \|p_u, p_j\|\}.$$

Equation 5 can be re-written as:

$$radius(i, j) = \min_{u=i}^{j} radius_u(i, j), \tag{6}$$

Thus, if v is the value of u at which the above equation is minimized, $center(i, j)$ equals p_v.

Our earlier proof for the continuity of covering (see Lemma 6) indicates that, as u moves from i to j, the value of $\|p_i, p_u\|$ continuously increases while that of $\|p_u, p_j\|$ continuously decreases. As a result, the value of $radius_u(i, j)$ initially decreases and then increases, exhibiting a V-shape. For the example of Figure 4, we plot $radius_u(2, 8)$ as u grows from 2 to 8 in Figure 6. Since $radius_u(2, 8)$ 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}, ..., p_j in this order, and stop once $radius_u(i, j)$ starts to increase, where u is the point being inspected. At this moment, we have just passed the minimum of Equation 6. Hence, we know $center(i, j) = p_{u-1}$ and $radius(i, j) = radius_{u-1}(i, j)$. 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 the time down 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 $O(m - i)$ 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.

Lemma 7. *For any $i \leq j_1 < j_2$, it holds that $center(i, j_1) \leq center(i, j_2)$.*

Proof. Let $u = center(i, j_1)$. We aim at showing that, for any $v \in [i, u)$, $radius_u(i, j_2) < radius_v(i, j_2)$. Hence, the center of the (i, j_2)-covering circle cannot be p_v; thus, the lemma is correct. Figure 7 illustrates the relative positions of p_i, p_v, p_u, p_{j_1} and p_{j_2}.

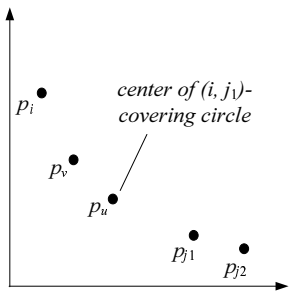


Figure 7: Illustration for the proof of Lemma 7

We distinguish two cases. First, if $\|p_u, p_i\| \leq \|p_u, p_{j_2}\|$, then

$$\begin{aligned} radius_u(i, j_2) &= \max\{\|p_u, p_i\|, \|p_u, p_{j_2}\|\} \\ &= \|p_u, p_{j_2}\| < \|p_v, p_{j_2}\| \\ &\leq \max\{\|p_v, p_i\|, \|p_v, p_{j_2}\|\} \\ &= radius_v(v, j_2). \end{aligned}$$

Next, we focus on the case where $\|p_u, p_i\| > \|p_u, p_{j_2}\|$. Hence, $radius_u(i, j_2) = \|p_u, p_i\|$. Since u is the center of the (i, j_1)-covering circle, we know

$$radius_u(i, j_1) \leq radius_v(i, j_1)$$

meaning that

$$\max\{\|p_i, p_u\|, \|p_u, p_{j_1}\|\} \leq \max\{\|p_i, p_v\|, \|p_v, p_{j_1}\|\}.$$

As $\|p_u, p_i\| > \|p_u, p_{j_2}\|$ which in turn is larger than $\|p_u, p_{j_1}\|$, the above inequality becomes:

$$\|p_i, p_u\| \leq \max\{\|p_i, p_v\|, \|p_v, p_{j_1}\|\}.$$

Observe that $\|p_i, p_u\| > \|p_i, p_v\|$. Hence the above implies $\|p_i, p_u\| \leq \|p_v, p_{j_1}\|$. Therefore:

$$\begin{aligned} radius_u(i, j_2) &= \|p_u, p_i\| \leq |p_v, p_{j_1}\| < \|p_v, p_{j_2}\| \\ &\leq \max\{\|p_v, p_i\|, \|p_v, p_{j_2}\|\} \\ &= radius_v(v, j_2). \end{aligned}$$

Thus we complete the proof. □

Suppose that we have performed 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 7 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 in search of $center(i, j_1 + 1)$ 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.

3.3 An improved algorithm

In this subsection, we will reduce the time complexity of our algorithm dramatically to $O(mk)$, i.e., improving *2d-opt* in Figure 5 by a factor of m. The performance bottleneck of *2d-opt* is that each iteration needs to evaluate Equation 3 too many – specifically $O(m)$ – times. Each evaluation incurs $O(m)$ cost, thus forcing an iteration to spend $O(m^2)$ time overall. Next, we show that it is possible to perform all the evaluations in an iteration collectively (in a way similar to a collective pass for covering circle computation) such that an iteration takes merely $O(m)$ time!

Given everything explained so far (particularly, the continuity of covering in the proof of Lemma 6), we can interpret the problem of finding an optimal representative skyline $\mathcal{K}$ in a slightly different manner. Let $p_1, ..., p_m$ be the points in the full skyline $\mathcal{S}$, sorted in ascending order of their x-coordinates. The goal is to partition the list into k disjoint sublists, each containing *consecutive* points in $\mathcal{S}$, such that all the points of a sublist are represented by a common representative, which belongs to the sublist as well. These representatives constitute $\mathcal{K}$. We will refer to each of those sublists as a *cluster*. Obviously, there is a total order among the clusters, and among the points in each cluster. As a quick example, the $\mathcal{K}$ of $k = 3$ for Figure 1 has three clusters: $\{p_1\}$, $\{p_2, ..., p_5\}$, and $\{p_6, ..., p_8\}$.

We need some extra notations. For any $j \leq i$, define

$$optEr_j(i,t) = \max\{optEr(j-1, t-1), radius(j,i)\} \tag{7}$$

with which Equation 3 can be simplified to:

$$optEr(i,t) = \min_{j=t}^{i} optEr_j(i,t). \tag{8}$$

Define $optPos(i,t)$ as the value of j that minimizes the above equation. This value has an intuitive meaning: it tells us where is the best position to start the last cluster, in an optimal size-t representative skyline of $\mathcal{S}_i = \{p_1, ..., p_i\}$. Sometimes, there may be multiple j that can minimize $optEr(i,t)$, in which case we define $optPos(i,t)$ to be the largest of them:

$$optPos(i,t) = \max\{j \mid optEr_j(i,t) = optEr(i,t)\}. \tag{9}$$

The above formulation essentially instructs us to look at the size-t representative skyline whose last cluster has the *smallest* cardinality. Now we give a crucial lemma:

Lemma 8. *For any $i \geq t$, $optPos(i+1,t) \geq optPos(i,t)$.*

Proof. Assume for contradiction that $optPos(i+1,t) < optPos(i,t)$ for some i and t. Let $\mathcal{K}_{i+1}$ be the optimal representative skyline of $\mathcal{S}_{i+1}$ that $optPos(i+1,t)$ corresponds to.

It is easy to see that $optEr(i,t)$ is a non-descending function of i for any fixed t, and $radius(j,i)$ is non-ascending in j for any fixed i. From $optEr(i+1,t) \geq optEr(i,t)$ we have:

$$\begin{aligned}
& optEr(i+1,t) \\
\geq\ & \max\{optEr(i,t), radius(optPos(i+1,t), i+1)\} && (10)\\
=\ & \max\Big\{ \max\big\{optEr(optPos(i,t)-1, t-1), radius(optPos(i,t), i)\big\}, \\
& radius(optPos(i+1,t), i+1)\Big\} && (11)
\end{aligned}$$

where the last equality used Equation 3. Now, dropping the term $radius(optPos(i,t), i)$ from the above and applying our assumption $optPos(i+1,t) < optPos(i,t)$, we have

$$(11) \quad \geq \quad \max\big\{optEr(optPos(i,t)-1, t-1), radius(optPos(i,t), i+1)\big\} \tag{12}$$

Algorithm *fast-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. compute $radius(1, i)$ and $center(1, i)$ for all $i \in [1, m]$
2. set $optEr(i, 1) = radius(1, i)$ for each $1 \leq i \leq m$
3. for $t = 2$ to k
4. $\quad j = t$
5. $\quad$ for $i = t$ to m
6. $\quad\quad$ while $j < m$ and $optEr_j(i, t) \geq optEr_{j+1}(i, t)$
7. $\quad\quad\quad j = j + 1$
8. $\quad\quad optEr(i, t) = optEr_j(i, t)$
9. $\quad\quad opt(i, t) = opt(j - 1, t - 1) \cup center(j, t)$
10. return $opt(k, m)$

Figure 8: A faster 2D optimal algorithm

The right hand side of the above inequality is, by definition, the error of a representative skyline, say $\mathcal{K}'_{i+1}$, of $\mathcal{S}_{i+1}$, where the last cluster starts with $optPos(i, t)$. Clearly, $\mathcal{K}'_{i+1}$ cannot have an error lower than $optEr(i + 1, t)$. Hence, (12) $\geq optEr(i + 1, t)$, implying that all the '$\geq$' in Inequalities 10-12 should be replaced by '='!

Now we see that $\mathcal{K}'_{i+1}$ turns out to be an *optimal* representative skyline of $\mathcal{S}_{i+1}$. Therefore, our assumption that $optPos(i + 1, t) < optPos(i, t)$ violates the maximality in the definition of $optPos(i + 1, t)$. □

The above lemma can be employed to accelerate *2d-opt* in a way similar to Lemma 7. Recall that each iteration of the algorithm needs to calculate $optEr(t, t)$, $optEr(t + 1, t)$, ..., $optEr(m, t)$ for some $t \leq k$. Before, the calculation of each of these terms requires evaluating Equation 3 *afresh*, but now, Lemma 8 enables us to take the same *never-look-back* approach as in a collective pass (Section 3.2). Specifically, assume that we have found a $j = optPos(i, t)$ that minimizes Equation 3 for $optEr(i, t)$. To find the best j for $optEr(i + 1, t)$, it suffices to search forward, i.e., considering only those $j \geq optPos(i, t)$. The effect is that, during all the evaluations of Equation 3 in an iteration, each value of j is examined at most once, as opposed to $O(m)$ times in *2d-opt*.

An issue remains before we can elaborate the details of a new algorithm – how do we know whether $j = optPos(i, t)$? In the context of Section 3.1, we settled a similar issue by resorting to a V-shape property (see Figure 6). Interestingly, a similar property also exists here:

Lemma 9. *For any fixed i, t, $optEr_j(i, t)$ is:*

- *a non-ascending function of j when $j \leq optPos(i, t)$,*
- *a non-descending function of j when $j \geq optPos(i, t)$.*

Proof. Obvious from Equation 7 because, as j grows, $optEr(j - 1, t - 1)$ is non-descending whereas $radius(j, i)$ is non-ascending. □

Equipped with this lemma, we can now evaluate Equation 8 (or equivalently, Equation 3) easily by comparing $optEr_j(i, t)$ with $optEr_{j+1}(i, t)$ while increasing j, and catch the best $j = optPos(i, t)$ as soon as we see $optEr_j(i, t) < optEr_{j+1}(i, t)$. Putting all the pieces together, we obtain a new algorithm *fast-2d-opt* in Figure 8 for solving Problem 5 optimally in 2D space.

A final remark concerns the computation of covering circles. Let us focus on their radii, $radius(i, j)$, because their centers can be obtained just as side-products. In Section 3.1, we dealt with this by simply producing $radius(i, j)$ for all possible i, j. Since that alone already takes $O(m^2)$ time – beating our objective

$O(mk)$ – we thus aim at computing *only* the necessary radii. All the $radius(1, i)$ required by Lines 1 and 2 of Figure 8 can be prepared by a single collective pass in $O(m)$ time. It remains to discuss those radii implicitly demanded by Line 6, recalling that $optEr_j(i, t)$ is defined as in Equation 7.

Let us call Lines 4-9 of *fast-2d-opt* an *iteration*. The crucial observation is that, in each iteration, the values of i and j are both *monotonically increasing*, which dictates that $center(i, j)$ must be monotonically increasing as well! This puts us in a situation analogous to what we encountered in Section 3.2, allowing us to deploy similar ideas to produce all the $radius(i, j)$ needed (by Line 6) in an iteration using only $O(m)$ time. Specifically, imagine that we have obtained $radius(i, j)$ and $p_u = center(i, j)$ (where $i \leq u \leq j$) through a simple scan (a procedure explained in Section 3.2). Let $radius(i', j')$ be the next radius required by the current iteration ($i' \geq i$ and $j' \geq j$). We pretend that the simple scan for computing that radius has come to $p_{\max\{u,i'\}}$, and continue the scan from there.

It is thus clear that all the covering circles needed by the k iterations of Figure 8 can be computed in $O(mk)$ time. The rest of the cost of the algorithm is easy to analyze, and is clearly bounded above by $O(mk)$.

4 The higher-dimensional case

We proceed to study Problem 5 in dimensionality $d \geq 3$. Section 4.1 shows that no polynomial-time algorithm exists for finding an optimal solution, and presents a method for obtaining a 2-approximate solution. Sections 4.2 and 4.3 discuss how to improve the efficiency of the method by using a multidimensional access method.

4.1 NP-hardness and 2-approximation

Lemma 10. *For any dimensionality $d \geq 3$, Problem 5 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 [13], to Problem 5. Specifically, given a set S of 2D points p_1, p_2, ..., p_n, the k-center problem aims at finding a subset S_k of S with k points that minimizes

$$\max_{p \in S} \{ \min_{p' \in S_k} \|p, p'\| \}. \tag{13}$$

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 an optimal solution S_k to the k-center problem.

Based on the 3D result, it is easy to show that Problem 5 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 D. Thus, we have found a polynomial time reduction from the 3D problem to the d-dimensional problem. □

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

where $Er(\cdot,\cdot)$ is given in Equation 1. Such a $\mathcal{K}$ can be found by a greedy algorithm similar to a procedure described in the proof of Lemma 4. 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 $\mathcal{S}$. Then, we add to $\mathcal{K}$ the point in $\mathcal{S}-\mathcal{K}$ with the largest representative distance (Equation 2), and repeat this until $|\mathcal{K}| = k$. We refer to this solution as *naive-greedy*. It guarantees a 2-approximate solution as can be established directly by the analysis of [11].

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*.

4.2 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 [1] 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 2. Finding this point is analogous to *farthest neighbor search*, using Equation 2 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 [14] is a well-known efficient algorithm for farthest neighbor search[2]. 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 4.3. 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

[2]Precisely speaking, *best-first* is originally designed for nearest neighbor search [14]. However, its adaptation to farthest neighbor search is trivial.

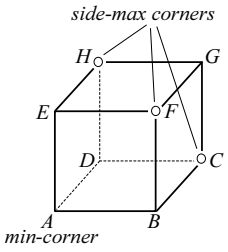


Figure 9: The side-max corners

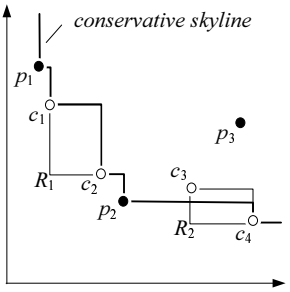


Figure 10: Conservative skyline

S' is generated as follows. First, it includes 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 9 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 10. 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 $\{p_1, c_1, c_2, p_2, c_4\}$, which is thus the conservative skyline of S.

To understand the usefulness of the conservative skyline, imagine R_1 and R_2 in Figure 10 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-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 11, 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 11. 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 $O(\log_B n)$ I/Os where B is the page size and n 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

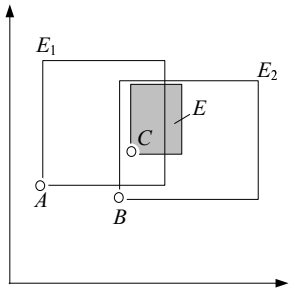


Figure 11: Illustration of the access order of *I-greedy*

has produced t representatives, then they definitely make a 2-approximate solution, i.e., their representation error at most twice larger than that of an 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 12 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* 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 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 S_{con}.

Recall that the *naive-greedy* algorithm in Section 4.1 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* [21]. 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 11. *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 S_{con}, and hence, cannot contain skyline points. Next, we will prove its I/O optimality.

As shown in [21], 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, and accesses a node N dominated by a skyline point p. This access must happen at either Line 7 or 15 in Figure 12. 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.

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) /* E' must be an intermediate entry */
7. access the child node N of E'
8. for each entry E_c in N
9. if ($E_c \neq p_{1st}$) and (E_c is not dominated by any point in S_{con})
10. insert E_c in $\mathcal{L}$
11. else /* E' does not exist */
12. if E is a point p
13. add p to $\mathcal{K}$ and output p
14. else
15. access the child node N of E
16. for each entry E_c in N
17. if ($E_c \neq p_{1st}$) and (E_c is not dominated by any point in S_{con})
18. insert E_c in $\mathcal{L}$

Figure 12: The *I-greedy* algorithm

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. □

4.3 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 *max-rep-dist*$(R, \mathcal{K})$, which must be at least the largest representative distance *rep-dist*$(p, \mathcal{K})$ of any point p in R.

To gain some insight about the smallest possible *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 13. The figure also illustrates the perpendicular bisector l of the segment connecting p_1 and p_2. For points p (i) above l, *rep-dist*$(p, \mathcal{K})$ equals $\|p, p_1\|$, (ii) below l, *rep-dist*$(p, \mathcal{K}) = \|p, p_2\|$, and (iii) on l, *rep-dist*$(p, \mathcal{K}) = \|p, p_1\| = \|p, p_2\|$. It is easy to see that, for points p in R, *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 13, as *rep-dist*$(q_1, \mathcal{K}) <$ *rep-dist*$(q_2, \mathcal{K})$, we know that *max-rep-dist*$(R, \mathcal{K}) =$ *rep-dist*$(q_2, \mathcal{K})$.

The implication of the above analysis is that, in order to derive the lowest *max-rep-dist*$(R, \mathcal{K})$, we need to resort to the *Voronoi diagram* [9] of the points in $\mathcal{K}$. The Voronoi diagram 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 *max-rep-dist*$(R, \mathcal{K})$ that is low, although may not be

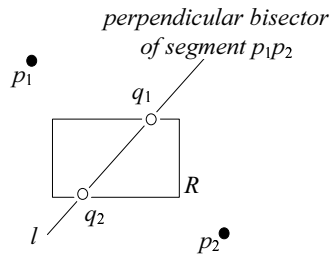


Figure 13: Finding the lowest value of *max-rep-dist*$(R, \mathcal{K})$

the lowest. Specifically, we set

$$\textit{max-rep-dist}(R, \mathcal{K}) = \min_{p \in \mathcal{K}} \{maxdist(p, R)\}. \tag{14}$$

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

Lemma 12. *The* max-rep-dist$(R, \mathcal{K})$ *from Equation 14 is at least as large as the* rep-dist$(p, \mathcal{K})$ *of any point* p *in* R*.*

Proof. Given any representative $p' \in \mathcal{K}$, it always holds that $\|p', p\| \leq maxdist(p', R)$. So $\min_{p' \in \mathcal{K}} \|p', p\| \leq \min_{p' \in \mathcal{K}} maxdist(p', R)$. The left side of the inequality is exactly *rep-dist*$(p, \mathcal{K})$. □

5 Experiments

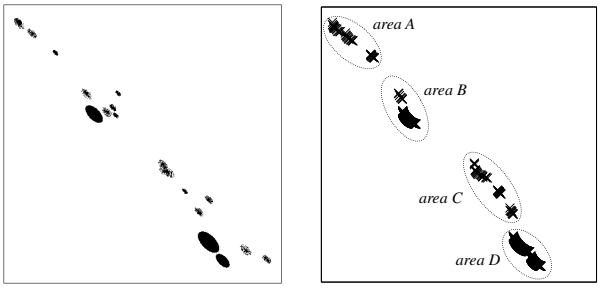
This section has two objectives. First, we will demonstrate that our distance-based representative skyline outperforms the previous method of max-dominance skyline [18] (reviewed in Section 2) in two crucial aspects: our representative skyline (i) better captures the contour of the full skyline, and (ii) is much cheaper to compute. This will establish distance-based representative skyline as an effective 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 14a. 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, as illustrated in Figure 14b.

NBA is a real dataset which is downloadable at *www.databasebasketball.com*, and frequently adopted in the skyline literature [22, 23, 29]. 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.

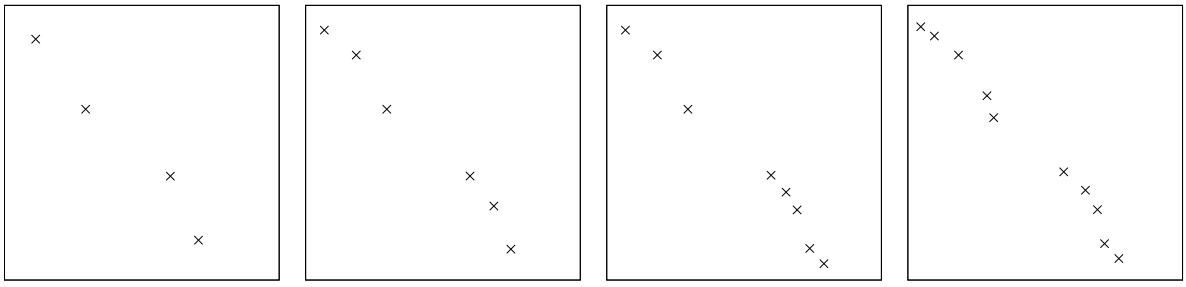
Besides the above datasets, we also created *anti-correlated* datasets with various cardinalities and dimensionalities to test the algorithms' scalability. The anti-correlated distribution has become a benchmark in the skyline research [2, 21, 27], and is very suitable for scalability test because it has a fairly sizable skyline. Our generation of this distribution follows exactly the description in the seminal work of [2].

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



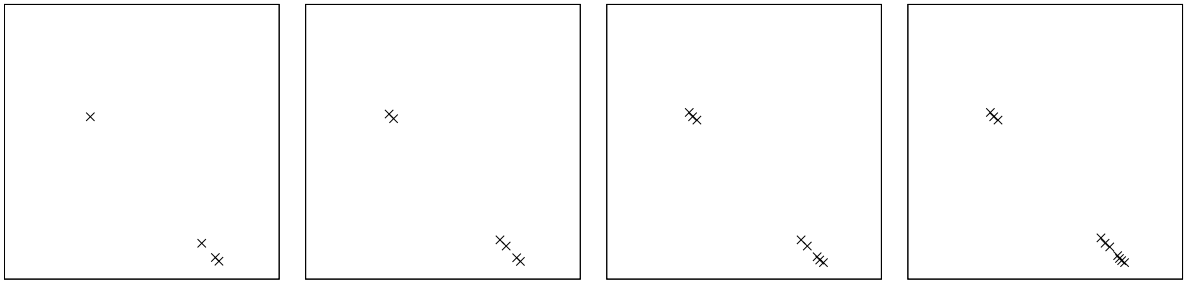
(a) The *Island* dataset (b) The skyline of *Island*

Figure 14: Visualization of the *Island* dataset



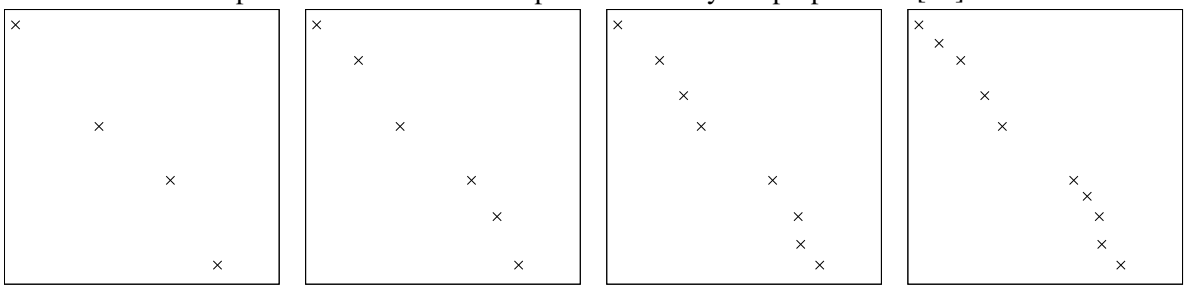
(a) $k = 4$ (b) $k = 6$ (c) $k = 8$ (d) $k = 10$

Optimal distance-based representative skyline



(e) $k = 4$ (f) $k = 6$ (g) $k = 8$ (h) $k = 10$

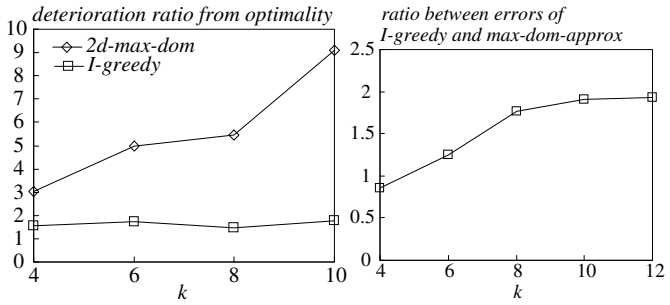
Optimal max-dominance representative skyline proposed in [18]



(i) $k = 4$ (j) $k = 6$ (k) $k = 8$ (l) $k = 10$

The output of *I-greedy*

Figure 15: Representative skylines



(a) *Island* (b) *NBA*

Figure 16: Representation error comparison

2d-max-dom. Specifically, *2d-opt* is the algorithm in Figure 5 that finds an optimal distance-based representative skyline. *I-greedy* is our 2-approximate algorithm in Section 4.2. *2d-max-dom* is the algorithm in [18] that computes an optimal max-dominance representative skyline. It is worth noting that *fast-2d-opt* in Figure 8 returns exactly the same result as *2d-opt*, whereas our other approximate algorithms *naive-greedy* and *best-first* in Sections 4.1 and 4.2 respectively have the same output as *I-greedy*.

As shown in Figure 14b, 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 15a-15d illustrate 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.

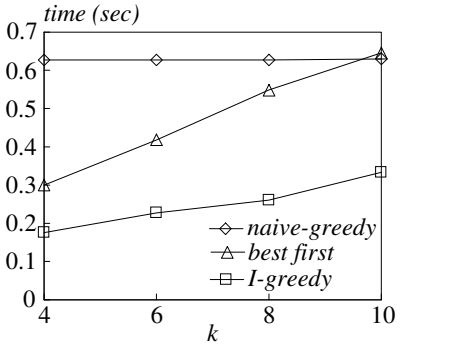
Figures 15e-15h present the max-dominance representative skylines found by *2d-max-dom*. Unfortunately, the representative skyline never involves any point from areas A and C. To understand this, notice that as shown in Figure 14, both areas B and D have a very dense cluster. Therefore, from the perspective 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 creat the misconception that no tradeoffs would be possible in areas A and C.

Finally, Figures 15i-15l 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.

Figure 16a shows the ratio between the representation error of *I-greedy* and that of *2d-opt* in the experiments of Figure 15, together with the ratio of *2d-max-dom* also with respect to *2d-opt*. 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.

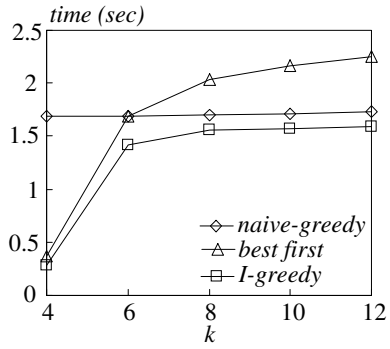
The next experiment inspects the representation error on dataset *NBA*. We again examine *I-greedy* but discard *2d-opt* and *2d-max-dom* because they are restricted to dimensionality 2. Instead, we compare *I-greedy* against *max-dom-approx*, which is an algorithm in [18] that returns a max-dominance skyline with theoretical guarantees in any dimensionality. Figure 16b 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*, *fast-2d-opt*, *naive-greedy*, *best-first*, *I-greedy*, *2d-max-dom*, and *max-dom-approx*. Recall that, as mentioned in



k	4	6	8	10
2d-opt	6.4	6.4	6.5	6.4
fast-2d-opt	0.60	0.61	0.61	0.62
2d-max-dom	$>$ 290 seconds			

(a) *Island*



k	4	6	8	10	12
max-dom-approx	$>$ 120 seconds				

(b) *NBA*

Figure 17: Running time vs. k

Section 4.2, *best-first* extends the traditional *best-first* algorithm for farthest nearest neighbor search with empty tests. Furthermore, algorithms *2d-opt*, *fast-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 17a 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 *fast-2d-opt*, *naive-greedy*, *best-first*, 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 due to its vast time complexity. Its performance is significantly improved by *fast-2d-opt*, confirming the analysis in Section 3.3. 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 (just as with *fast-2d-opt*). 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 17b presents the results of the same experiment on dataset *NBA*. Note that we leave out the 2D algorithms *2d-opt* and *fast-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 17a, except that *best-first* becomes slower than *naive-greedy* after $k = 6$.

To further analyze *fast-2d-opt*, *naive-greedy*, *best-first*, and *I-greedy*, in Table 1a, we provide their detailed I/O and CPU time in the experiments of Figure 17a. Table 1b gives the same information with respect to Figure 17b, but excluding the inapplicable *fast-2d-opt*. 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 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* and *fast-2d-opt* are 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 starting from $k = 8$.

The following experiment investigates the scalability of *fast-2d-opt*, *naive-greedy*, *best-first*, and *I-greedy* with respect to the dataset cardinality. For this purpose, we fix k to 10. Using 2D anti-correlated

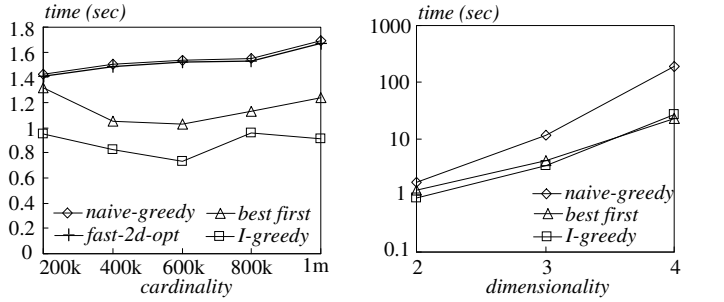
k	4	6	8	10
fast-2d-opt	54 (.06)	54 (.07)	54 (.07)	54 (.08)
naive-greedy	54 (.09)	54 (.09)	54 (.09)	54 (.09)
best-first	24 (.06)	33 (.09)	42 (.13)	50 (.15)
I-greedy	10 (.18)	12 (.23)	14 (.26)	17 (.33)

(a) *Island*

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

(b) *NBA*

Table 1: Breakdowns of running time. Format: I/O (CPU)



(a) Vs. cardinality (2D) (b) Vs. dimensionality (1m card.)

Figure 18: Scalability comparison (anti-correlated)

datasets, Figure 18a plots the overall execution time of the four 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 [21] 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 18b compares *naive-greedy*, *best-first*, and *I-greedy* on 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.

6 Related work

The first work on skylines in the database area is due to Borzsonyi et al. [2]. Since then, many algorithms have been developed for computing skylines efficiently, for example, *Bitmap* [27], *NN* [17], *BBS* [21], *SFS* [7], *LESS* [12], *Lattice* [20], *rand* [26], etc. These algorithms focus on the original data space, while considerable efforts have also been made to retrieve skylines in subspaces, such as *subsky* [29], *skycube* [23], and so on. Besides traditional centralized DBMS, skyline search has also been studied in distributed

systems [8, 30], streams [25], p2p networks [31], partially-ordered domains [3], 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 [18]. Other examples include k*-dominant skyline* [4], *spatial skyline* [24], *probabilistic skyline* [22], *reverse skyline* [10, 19], *privacy skyline* [6], *approximately dominating representatives* [16], retrieving the points with the highest *subspace skyline frequencies* [5], and so on.

The *best-first* algorithm mentioned in Section 4.2 is proposed by Hjaltason and Samet [14] 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 [15]. The greedy algorithm described in Section 4.1 provides a 2-approximate solution for any distance metric satisfying the triangle inequality [11].

7 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 [18] 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 [18]. We also develop several algorithms for computing distance-based representative skylines. In 2D space, our technique finds an optimal solution in time linear to both the number of points in the full skyline and the output size. In higher dimensional spaces where computing an 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 [18].

References

[1] N. Beckmann, H. Kriegel, R. Schneider, and B. Seeger. The R*-tree: An efficient and robust access method for points and rectangles. In *Proceedings of ACM Management of Data (SIGMOD)*, pages 322–331, 1990.

[2] Stephan Börzsönyi, Donald Kossmann, and Konrad Stocker. The skyline operator. In *Proceedings of International Conference on Data Engineering (ICDE)*, pages 421–430, 2001.

[3] Chee Yong Chan, Pin-Kwang Eng, and Kian-Lee Tan. Stratified computation of skylines with partially-ordered domains. In *Proceedings of ACM Management of Data (SIGMOD)*, pages 203–214, 2005.

[4] Chee Yong Chan, H. V. Jagadish, Kian-Lee Tan, Anthony K. H. Tung, and Zhenjie Zhang. Finding k-dominant skylines in high dimensional space. In *Proceedings of ACM Management of Data (SIGMOD)*, pages 503–514, 2006.

[5] Chee Yong Chan, H. V. Jagadish, Kian-Lee Tan, Anthony K. H. Tung, and Zhenjie Zhang. On high dimensional skylines. In *Proceedings of Extending Database Technology (EDBT)*, pages 478–495, 2006.

[6] Bee-Chung Chen, Raghu Ramakrishnan, and Kristen LeFevre. Privacy skyline: Privacy with multidimensional adversarial knowledge. In *Proceedings of Very Large Data Bases (VLDB)*, pages 770–781, 2007.

[7] Jan Chomicki, Parke Godfrey, Jarek Gryz, and Dongming Liang. Skyline with presorting. In *Proceedings of International Conference on Data Engineering (ICDE)*, pages 717–816, 2003.

[8] Bin Cui, Hua Lu, Quanqing Xu, Lijiang Chen, Yafei Dai, and Yongluan Zhou. Parallel distributed processing of constrained skyline queries by filtering. In *Proceedings of International Conference on Data Engineering (ICDE)*, pages 546–555, 2008.

[9] Mark de Berg, Otfried Cheong, Marc van Kreveld, and Mark Overmars. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, 2008.

[10] Evangelos Dellis and Bernhard Seeger. Efficient computation of reverse skyline queries. In *Proceedings of Very Large Data Bases (VLDB)*, pages 291–302, 2007.

[11] Tomas Feder and Daniel Greene. Optimal algorithms for approximate clustering. In *Proceedings of ACM Symposium on Theory of Computing (STOC)*, pages 434–444, 1988.

[12] Parke Godfrey, Ryan Shipley, and Jarek Gryz. Algorithms and analyses for maximal vector computation. *The VLDB Journal*, 16(1):5–28, 2007.

[13] Teofilo F. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theoretical Computer Science*, 38:293–306, 1985.

[14] G. R. Hjaltason and H. Samet. Distance browsing in spatial databases. *ACM Transactions on Database Systems (TODS)*, 24(2):265–318, 1999.

[15] W. L. Hsu and G. L. Nemhauser. Easy and hard bottleneck location problems. *Discrete Applied Mathematics*, 1:209–216, 1979.

[16] Vladlen Koltun and Christos H. Papadimitriou. Approximately dominating representatives. In *Proceedings of International Conference on Database Theory (ICDT)*, pages 204–214, 2005.

[17] Donald Kossmann, Frank Ramsak, and Steffen Rost. Shooting stars in the sky: An online algorithm for skyline queries. In *Proceedings of Very Large Data Bases (VLDB)*, pages 275–286, 2002.

[18] Xuemin Lin, Yidong Yuan, Qing Zhang, and Ying Zhang. Selecting stars: The k most representative skyline operator. In *Proceedings of International Conference on Data Engineering (ICDE)*, pages 86–95, 2007.

[19] Ziyang Liu and Yi Chen. Reasoning and identifying relevant matches for XML keyword search. *Proceedings of the VLDB Endowment (PVLDB)*, 1(1):921–932, 2008.

[20] Michael D. Morse, Jignesh M. Patel, and H. V. Jagadish. Efficient skyline computation over low-cardinality domains. In *Proceedings of Very Large Data Bases (VLDB)*, pages 267–278, 2007.

[21] Dimitris Papadias, Yufei Tao, Greg Fu, and Bernhard Seeger. An optimal and progressive algorithm for skyline queries. In *Proceedings of ACM Management of Data (SIGMOD)*, pages 467–478, 2003.

[22] Jian Pei, Bin Jiang, Xuemin Lin, and Yidong Yuan. Probabilistic skylines on uncertain data. In *Proceedings of Very Large Data Bases (VLDB)*, pages 15–26, 2007.

[23] Jian Pei, Yidong Yuan, Xuemin Lin, Wen Jin, Martin Ester, Qing Liu, Wei Wang 0011, Yufei Tao, Jeffrey Xu Yu, and Qing Zhang. Towards multidimensional subspace skyline analysis. *ACM Transactions on Database Systems (TODS)*, 31(4):1335–1381, 2006.

[24] Peter Sanders and Dominik Schultes. Engineering highway hierarchies. In *Proceedings of European Symposium on Algorithms (ESA)*, pages 804–816, 2006.

[25] Nikos Sarkas, Gautam Das, Nick Koudas, and Anthony K. H. Tung. Categorical skylines for streaming data. In *Proceedings of ACM Management of Data (SIGMOD)*, pages 239–250, 2008.

[26] Atish Das Sarma, Ashwin Lall, Danupon Nanongkai, and Jun Xu. Randomized multi-pass streaming skyline algorithms. *Proceedings of the VLDB Endowment (PVLDB)*, 2(1):85–96, 2009.

[27] Kian-Lee Tan, Pin-Kwang Eng, and Beng Chin Ooi. Efficient progressive skyline computation. In *Proceedings of Very Large Data Bases (VLDB)*, pages 301–310, 2001.

[28] Yufei Tao, Ling Ding, Xuemin Lin, and Jian Pei. Distance-based representative skyline. In *Proceedings of International Conference on Data Engineering (ICDE)*, pages 892–903, 2009.

[29] Yufei Tao, Xiaokui Xiao, and Jian Pei. Subsky: Efficient computation of skylines in subspaces. In *Proceedings of International Conference on Data Engineering (ICDE)*, 2006.

[30] Akrivi Vlachou, Christos Doulkeridis, and Yannis Kotidis. Angle-based space partitioning for efficient parallel skyline computation. In *Proceedings of ACM Management of Data (SIGMOD)*, pages 227–238, 2008.

[31] Shiyuan Wang, Beng Chin Ooi, Anthony K. H. Tung, and Lizhen Xu. Efficient skyline query processing on peer-to-peer networks. In *Proceedings of International Conference on Data Engineering (ICDE)*, pages 1126–1135, 2007.