# Diversity in Skylines

Yufei Tao
Department of Computer Science and Engineering
Chinese University of Hong Kong
Sha Tin, New Territories, Hong Kong
taoyf@cse.cuhk.edu.hk

**Abstract**

*Given an integer $k$, a* diverse skyline *contains the $k$ skyline points that best describe the tradeoffs (among different dimensions) offered by the full skyline. This paper gives an overview of the latest results on this topic. Specifically, we first describe the state-of-the-art formulation of diverse skylines. Then, we explain several algorithms for finding a diverse skyline, where the objective is to save cost by avoiding the computation of the entire skyline. In particular, we will discuss a polynomial-time algorithm in 2D space that returns the exact result, the NP-hardness of the problem in dimensionality at least 3, and an approximate solution with good quality guarantees.*

## 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 $p_1$, $p_2$, ..., $p_8$.

Skyline retrieval has received considerable attention from the database community, resulting in a large number of interesting results as surveyed in Section 5. These research efforts reflect the crucial importance of skylines in practice. In particular, it is well-known [18] 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 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


**Bulletin of the IEEE Computer Society Technical Committee on Data Engineering**

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

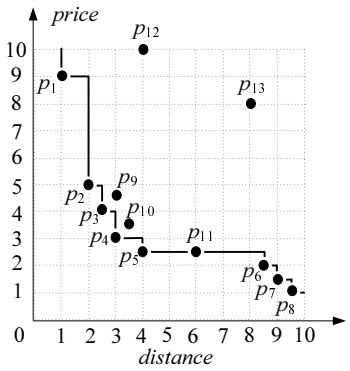


Figure 1: A skyline example

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 article, we discuss the problem of retrieving *diverse skylines*, which includes a small number $k$ of skyline points that best describe the *representative* tradeoffs in the full skyline. For example, given $k = 3$, the representatives would be $p_1$, $p_4$, and $p_7$, each of which comes from a distinct subset illustrated above, providing a unique tradeoff. Diverse skylines are especially helpful in web-based recommendation systems such as the one in our previous hotel example. Skyline computation can be rather costly, 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. The importance of diverse skylines is further discussed in [16, 24].

Skyline diversity can be formulated in several ways [16, 24]. The state-of-the-art definition [24] is designed based on the intuition that, for every non-representative skyline point, there should be a nearby representative. In this paper, we will focus on that definition, and the corresponding computation algorithms. In 2D space, the problem can be settled in polynomial time by a dynamic programming algorithm. For dimensionality at least 3, the problem is NP-hard, but fortunately we show that there is 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.

The rest of the paper is organized as follows. Section 2 clarifies the formulation of diverse skylines. Section 3 presents an algorithm for finding the optimal diverse skyline in 2D space. Section 4 tackles the problem of retrieving diverse skylines in dimensionality at least 3. Section 5 surveys the previous literature on skyline. Finally, Section 6 concludes the paper with a summary.

## 2 Formulation of Diverse Skylines

Let $\mathcal{D}$ be a set of $d$-dimensional points. We use $\mathcal{S}$ to denote the full skyline of $\mathcal{D}$. The quality of a diverse skyline can be evaluated with the concept of *representation error* of $\mathcal{K}$, denoted as $Er(\mathcal{K}, \mathcal{S})$. Intuitively, a Diverse 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:

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

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 $\mathcal{K}$. The following definition is the state-of-the-art formulation of diverse skylines [24]:

**Definition 1:** Let $\mathcal{D}$ be a multidimensional dataset and $\mathcal{S}$ its skyline. Given an integer $k$, the *diverse 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. Each point in $\mathcal{S}$ is a called a *representative*. □

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

The diverse skyline is essentially the optimal solution of the $k$-*center problem* [10] on the full skyline $\mathcal{S}$. As a result, the diverse 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. Instead, we would like to distribute the representatives evenly along the skyline, regardless of the densities of the underlying clusters. This is why 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.

From now on, we will use the term *sub-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-$k$ sub-skyline $\mathcal{K}$ that has the smallest representation error $Er(\mathcal{K}, \mathcal{S})$ given in Equation 1. □

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

## 3 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 $p_1$, $p_2$, ..., $p_m$, sorted in ascending order of their x-coordinates.

We adopt the notation $\mathcal{S}_i$ to represent $\{p_1, p_2, ..., p_i\}$ where $i \leq m$. Specially, define $\mathcal{S}_0 = \emptyset$. Introduce a function $opt(i, t)$ to be the optimal size-$t$ diverse skyline of $\mathcal{S}_i$, where $t \leq i$. Hence, the optimal size-$k$ diverse

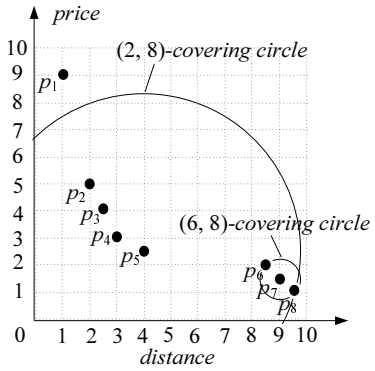


Figure 2: Covering circles

---

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

---

Figure 3: An optimal algorithm for computing 2D diverse skylines

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 $p_i$, $p_{i+1}$, ..., $p_j$, and (ii) centers at one of these $j - i + 1$ points. Call the circle the $(i, j)$-*covering circle*, and denote its center as $center(i, j)$. Figure 2 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 $t \geq 2$:

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

The above equation is based on the following rationale. Assume, without loss of generality, that the optimal size-$t$ diverse skyline of $\mathcal{S}_i$ is $\{p_{j_1}, p_{j_2}, ..., p_{j_t}\}$ with $1 \leq j_1 < j_2 < ... < j_t \leq i$, i.e., $p_{j_1}$,..., $p_{j_t}$ 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, $\{p_{j_1}, ..., p_{j_{t-1}}\}$ must be the optimal size-$(t - 1)$ diverse 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:

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

Equations 2 and 3 point to a dynamic programming algorithm *2D-opt* in Figure 3 for computing $opt(k, m)$, i.e., the size-$k$ diverse 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 $t$, 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 3) 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

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

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

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

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. 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 $p_u.radius(i,j)$ 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.

The time can be brought down to $O(m^2)$ with a method called *collective pass*. The main idea 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 details can be found in [24].

# 4 The Higher-dimensional Case

We proceed to study Problem 1 in dimensionality $d \geq 3$. As proved in [24], the problem is NP-hard for $d \geq 3$. Section 4.1 describes an algorithm for finding 2-approximate solution. Then, Section 4.2 discusses how to improve the efficiency of the approximate algorithm.

## 4.1 2-approximation

A 2-approximate solution $\mathcal{K}$ is a sub-skyline with $k$ points whose representation error is at most twice that of $\mathcal{K}^*$. Namely, if the optimal diverse skyline is $\mathcal{K}^*$, then $Er(\mathcal{K}, \mathcal{S}) \leq 2 \cdot Er(\mathcal{K}^*, \mathcal{S})$, where $Er(\cdot,\cdot)$ is given in Equation 1.

Such a $\mathcal{K}$ can be found by a standard greedy algorithm [9] 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:

$$\textit{rep-dist}(p, \mathcal{K}) = \min_{p' \in \mathcal{K}} \|p, p'\|. \tag{6}$$

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 [9].

As an example, consider the dataset in Figure 1, where $\mathcal{S} = \{p_1, p_2, ..., p_8\}$. 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 outline an alternative algorithm called *I-greedy* which overcomes both drawbacks of *naive-greedy*.

### 4.2 I-greedy

*I-greedy* assumes a multidimensional index (such as an R-tree [1]) on the dataset $\mathcal{D}$. It 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 6. Finding this point is analogous to *farthest neighbor search*, using Equation 6 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 [11] 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 $R$. *max-rep-dist*$(R, \mathcal{K})$ can be easily computed, as shown in [24]. 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. The details can be found in [24].

## 5 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* [23], *NN* [14], *BBS* [18], *Lattice* [17],

[2]Precisely speaking, *best-first* is originally designed for nearest neighbor search [11]. However, its adaptation to farthest neighbor search is trivial.

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* [25], *skycube* [20], and so on. Besides traditional centralized DBMS, skyline search has also been studied in distributed systems [7, 26], streams [21], p2p networks [27], partially-ordered domains [3], etc. The concept of skyline has numerous useful variations. One example is the *diverse skyline* studied in this paper, and this notion is originally introduced in [16]. Other examples include $k$*-dominant skyline* [4], *spatial skyline* [22], *probabilistic skyline* [19], *reverse skyline* [8, 15], *privacy skyline* [6], *approximately dominating representatives* [13], 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 [11] 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 diverse skylines, 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 $\varepsilon$ [12]. The greedy algorithm described in Section 4.1 provides a 2-approximate solution for any distance metric satisfying the triangle inequality [9].

# 6 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. In this article, we described the formulation of such a diverse skyline, and discussed its computation algorithms. In 2D space, the problem can be solved optimally in low-degree polynomial time. In higher dimensional spaces where computing the optimal solution is NP-hard, it is possible to return a 2-approximate solution efficiently.

# References


[1] N. Beckmann, H. Kriegel, R. Schneider, B. Seeger, *The R*-Tree: An Efficient and Robust Access Method for Points and Rectangles*, in Proc. of ACM Management of Data (SIGMOD), 322–331, 1990.

[2] S. Borzsonyi, D. Kossmann, K. Stocker, *The Skyline Operator*, in Proc. of Int'l Conference on Data Engineering (ICDE), 421–430, 2001.

[3] C. Y. Chan, P.-K. Eng, K.-L. Tan, *Stratified Computation of Skylines with Partially-Ordered Domains*, in Proc. of ACM Management of Data (SIGMOD), 203–214, 2005.

[4] C. Y. Chan, H. V. Jagadish, K.-L. Tan, A. K. H. Tung, Z. Zhang, *Finding $k$-dominant skylines in high dimensional space*, in Proc. of ACM Management of Data (SIGMOD), 503–514, 2006.

[5] C. Y. Chan, H. V. Jagadish, K.-L. Tan, A. K. H. Tung, Z. Zhang, *On High Dimensional Skylines*, in Proc. of Extending Database Technology (EDBT), 478–495, 2006.

[6] B.-C. Chen, R. Ramakrishnan, K. LeFevre, *Privacy Skyline: Privacy with Multidimensional Adversarial Knowledge*, in Proc. of Very Large Data Bases (VLDB), 770–781, 2007.

[7] B. Cui, H. Lu, Q. Xu, L. Chen, Y. Dai, Y. Zhou, *Parallel Distributed Processing of Constrained Skyline Queries by Filtering*, in Proc. of Int'l Conference on Data Engineering (ICDE), 546–555, 2008.

[8] E. Dellis, B. Seeger, *Efficient Computation of Reverse Skyline Queries*, in Proc. of Very Large Data Bases (VLDB), 291–302, 2007.

[9] T. Feder, D. Greene, *Optimal algorithms for approximate clustering*, in Proc. of ACM Symposium on Theory of Computing (STOC), 434–444, 1988.

[10] T. F. Gonzalez, *Clustering to Minimize the Maximum Intercluster Distance*, in Theor. Comput. Sci., (38), 293–306, 1985.

[11] G. R. Hjaltason, H. Samet, *Distance Browsing in Spatial Databases.*, in ACM Transactions on Database Systems (TODS), (24):2, 265–318, 1999.

[12] W. L. Hsu, G. L. Nemhauser, *Easy and hard bottleneck location problems*, in Disc. Appl. Math. (1), 209–216, 1979.

[13] V. Koltun, C. H. Papadimitriou, *Approximately Dominating Representatives*, in Proc. of Int'l Conference on Database Theory (ICDT), 204–214, 2005.

[14] D. Kossmann, F. Ramsak, S. Rost, *Shooting Stars in the Sky: An Online Algorithm for Skyline Queries*, in Proc. of Very Large Data Bases (VLDB), 275–286, 2002.

[15] X. Lian, L. Chen, *Monochromatic and bichromatic reverse skyline search over uncertain databases*, in Proc. of ACM Management of Data (SIGMOD), 213–226, 2008.

[16] X. Lin, Y. Yuan, Q. Zhang, Y. Zhang, *Selecting Stars: The $k$ Most Representative Skyline Operator*, in Proc. of Int'l Conference on Data Engineering (ICDE), 86–95, 2007.

[17] M. Morse, J. M. Patel, H. V. Jagadish, *Efficient Skyline Computation over Low-Cardinality Domains*, in Proc. of Very Large Data Bases (VLDB), 267–278, 2007.

[18] D. Papadias, Y. Tao, G. Fu, B. Seeger, *An Optimal and Progressive Algorithm for Skyline Queries*, in Proc. of ACM Management of Data (SIGMOD), 467–478, 2003.

[19] J. Pei, B. Jiang, X. Lin, Y. Yuan, *Probabilistic Skylines on Uncertain Data*, in Proc. of Very Large Data Bases (VLDB), 15–26, 2007.

[20] J. Pei, W. Jin, M. Ester, Y. Tao, *Catching the Best Views of Skyline: A Semantic Approach Based on Decisive Subspaces*, in Proc. of Very Large Data Bases (VLDB), 253–264, 2005.

[21] N. Sarkas, G. Das, N. Koudas, A. K. H. Tung, *Categorical skylines for streaming data*, in Proc. of ACM Management of Data (SIGMOD), 239–250, 2008.

[22] M. Sharifzadeh, C. Shahabi, *The Spatial Skyline Queries*, in Proc. of Very Large Data Bases (VLDB), 751–762, 2006.

[23] K.-L. Tan, P.-K. Eng, B. C. Ooi, *Efficient Progressive Skyline Computation*, in Proc. of Very Large Data Bases (VLDB), 301–310, 2001.

[24] Y. Tao, L. Ding, X. Lin, J. Pei, *Distance-Based Representative Skyline*, in Proc. of Int'l Conference on Data Engineering (ICDE), 892–903, 2009.

[25] Y. Tao, X. Xiao, J. Pei, *SUBSKY: Efficient Computation of Skylines in Subspaces*, in Proc. of Int'l Conference on Data Engineering (ICDE), 65–65, 2006.

[26] A. Vlachou, C. Doulkeridis, Y. Kotidis, *Angle-based space partitioning for efficient parallel skyline computation*, in Proc. of ACM Management of Data (SIGMOD), 227–238, 2008.

[27] S. Wang, B. C. Ooi, A. K. H. Tung, L. Xu, *Efficient Skyline Query Processing on Peer-to-Peer Networks*, in Proc. of Int'l Conference on Data Engineering (ICDE), 1126–1135, 2007.