Efficient Algorithms for Distance-Based Representative Skyline Computation in 2D Space

Taotao Cai[1], Rong-Hua Li[1], Jeffrey Xu Yu[2], Rui Mao[1], and Yadi Cai[1]

[1] Guangdong Province Key Laboratory of Popular High Performance Computers, Shenzhen University, China
[2] The Chinese University of Hong Kong, Hong Kong
{taotaocai1992,iyadicai}@gmail.com, {rhli,mao}@szu.edu.cn, yu@se.cuhk.edu.hk

Abstract. Representative skyline computation is a fundamental issue in database area, which has attracted much attention in recent years. A notable definition of representative skyline is the distance-based representative skyline (DBRS). Given an integer k, a DBRS includes k representative skyline points that aims at minimizing the maximal distance between a non-representative skyline point and its nearest representative. In the 2D space, the state-of-the-art algorithm to compute the DBRS is based on dynamic programming (DP) which takes $O(km^2)$ time complexity, where m is the number of skyline points. Clearly, such a DP-based algorithm cannot be used for handling large scale dataset due to the quadratic time cost. To overcome this problem, in this paper, we propose a new approximate algorithm called ARS, and a new exact algorithm named PSRS, based on a carefully-designed parametric search technique. We show that the ARS algorithm can guarantee a solution that is at most ϵ larger than the optimal solution. The proposed ARS and PSRS algorithms run in $O(k \log^2 m \log(T/\epsilon))$ and $O(k^2 \log^3 m)$ time respectively, where T is no more than the maximal distance between any two skyline points. We conduct extensive experimental studies over both synthetic and real-world datasets, and the results demonstrate the efficiency and effectiveness of the proposed algorithms.

1 Introduction

Skyline computation is a fundamental problem in database area [2], which has attracted considerable attention in database community. Given a set of d-dimensional points, the skyline is a subset of points that are not dominated by any other points. Here a point p dominating a point q ($p \neq q$) means that q's coordinate is no larger than p's coordinate in all dimensions. The skyline operator is particularly useful in multi-criteria decision making related applications [2,10].

However, in many applications, the number of skyline points are typically very large, and even may be comparable to the size of the entire dataset [14]. To overcome this issue, a promising method is to select k representatives to represent the entire skyline [8,14]. In the literature, there are several definitions of representative skyline [8,14], including the notable domination-based representative skyline [8] and distance-based representative skyline [14], which will be surveyed in Section 6. As illustrated in [14], the defect of the domination-based representative skyline is that it may allocate many

R. Cheng et al. (Eds.): APWeb 2015, LNCS 9313, pp. 116–128, 2015.
DOI: 10.1007/978-3-319-25255-1_10

representatives to a dense region of the skyline points, while the distance-based representative skyline (DBRS) can overcome this drawback. Therefore, in this paper, we focus on the problem of computing the DBRS efficiently. Specifically, given an integer k, a DBRS consists of k representative skyline points that aims to minimize the maximal distance between a non-representative skyline point and its nearest representative. In the 2D space, there is a dynamic programming (DP) algorithm to compute the DBRS [14], which takes $O(km^2)$ time complexity, where m is the number of skyline points. Obviously, such a DP-based solution cannot be applied to large datasets due to the quadratic time overhead.

To speedup the DP-based algorithm, in this paper, we first propose an efficient approximate algorithm, called ARS, which runs in $O(k \log^2 m \log(T/\epsilon))$ time, where T is no more than the maximal distance between any two skyline points. We show that the ARS algorithm can guarantee a solution that is at most ϵ larger than the optimal solution. Second, we propose an exact algorithm, called PSRS, which runs in $O(k^2 \log^3 m)$ time. Unlike the DP-based algorithm, both the ARS and PSRS algorithms are based on a carefully-designed parametric search technique. Specifically, to compute the DBRS, we first solve a related decision problem described as follows. Given a distance τ, can we find k representatives such that the distance between a non-representative point and its nearest representative is no larger than τ? If that is the case, we refer to τ as a feasible solution, and otherwise it is non-feasible. Clearly, the minimal feasible solution of the decision problem is the optimal solution of the DBRS problem. To solve the decision problem, we propose a binary-search based greedy algorithm. The idea is that the algorithm greedily covers the skyline points using a circle with radius τ, and then checks whether all the skyline points can be covered by k circles. We prove that if the optimal solution of the DBRS problem is no smaller than τ, then such a greedy algorithm returns *true*, otherwise returns *false*. Based on this, the ARS algorithm makes use of a binary search technique to search the minimal feasible τ, which can obtain a near-optimal solution. However, for the PSRS algorithm, we first observe that there are k feasible solutions that can be the candidates for the optimal solution. Based on this key observation, the PSRS algorithm utilizes a binary search procedure to find all these k candidates, and then reports the minimal one. We conduct extensive experiments over both synthetic and real-world datsets, the results show that the ARS and PSRS algorithms are at least four orders of magnitude faster than the DP-based algorithm in medium-sized datasets. The main contributions of this paper are summarized as follows.

- We propose a near-optimal approximate algorithm ARS that runs in $O(k \log^2 m \log(T/\epsilon))$ time to solve the DBRS problem in 2D space.
- We also propose an efficient exact algorithm PSRS which takes $O(k^2 \log^3 m)$ time complexity to compute the DBRS in 2D space. Unlike the DP-based algorithm, both the ARS and PSRS algorithms are based on a new parametric search technique.
- We conduct comprehensive experimental studies to evaluate the proposed algorithms, and the results confirm our theoretical findings.

The rest of the paper is organized as follows. Section 2 describes the problem of distance-based representative skyline computation. In Sections 3 and 4, we propose the ARS and PSRS algorithms respectively. We report the experimental results in Section 5,

and survey the related work in Section 6. Finally, we conclude this work and point out several future directions in Section 7.

2 Preliminaries

Let D be a set of 2D points, and S be a set of skyline points of D. Denote by n and m the cardinality of D and S respectively (i.e., $n = |D|$ and $m = |S|$). Further, we let K be the representative skyline that contains k points from S. Following [14], the definition of representative error of K, denoted by $E(K, S)$, is given by

$$E(K,S) = \max_{p \in S \setminus K} \{\min_{q \in K} ||p,q||\}, \tag{1}$$

where $||p, q||$ denotes the distance between points p and q. The representative error denotes the maximal distance between the non-representative point from $S\backslash K$ and its nearest representatives, which quantifies the representation quality of K. A good representative skyline should be with small $E(K, S)$. The goal of the distance-based representative skyline (DBRS) is to find a representative skyline K with minimal representative error $E(K, S)$. More formally, the DBRS problem is formulated as follows.

$$\begin{aligned} &\min_{K} E(K,S) \\ &s.t. \quad |K| \leqslant k. \end{aligned} \tag{2}$$

In [14], the authors proposed a dynamic programming algorithm to solve the above problem in 2D space. In the rest of this paper, we assume that the skyline R has already been computed from the dataset D and the skyline points are sorted in ascending order of their x-coordinate, which is a very common assumption in representative skyline literature [14,8]. Let $S = \{p_1, \cdots, p_m\}$ be the skyline points that is sorted in non-decreasing order of their x-coordinate. Denote by $S_j = \{p_1, \cdots, p_j\}$ a subset of S that includes the first j elements in S. Let $optE(j, t)$, for $0 \leq j \leq m, 0 \leq t \leq k$, be the representative error of the optimal size-t representative skyline of R_j. Let $r(i, j)$, for $1 \leq i \leq j \leq m$, be the radius of the smallest circle that covers all points in $\{p_i \cdots, p_j\}$ and centers at one of these points. Then, we have the following recursive formula to compute $optE(m, k)$ [14].

$$\begin{aligned} optE(j,t) &= \min_{i=t}^{j-1} \{\max\{optE(i-1,t-1), r(i,j)\}\}. \\ optE(j,j) &= 0 \end{aligned} \tag{3}$$

Armed with the above formula, we can easily devise a dynamic programming algorithm to compute $optE(m, k)$. By a careful implementation as shown in [14], the DP-based algorithm runs in $O(km^2)$ time. Clearly, such a DP-based solution cannot be used for large datasets due to the quadratic time cost. To overcome this issue, in the following sections, we shall present two efficient algorithms to compute the optimal representative skyline in 2D space. In the rest of this paper, we will focus on computing the optimal representative error, and it is straightforward to extend the proposed techniques to find the optimal representative points.

Algorithm 1. Compute the radius of the smallest circle $r(i, j)$

```
1: i_0 ← i, j_0 ← j;
2: while i < j do
3:    mid ← ⌊(i + j)/2⌋, id ← ‖p_{i_0}, p_{mid}‖, jd ← ‖p_{j_0}, p_{mid}‖;
4:    if id < jd then
5:       i ← mid + 1;
6:    else if id > jd then
7:       j ← mid;
8:    else
9:       return id;
10: return max{id, jd};
```

3 The Approximate Algorithm

In this section, we propose a near-optimal approximate algorithm, called ARS, to compute the representative skyline in 2D space. Unlike the DP algorithm, the ARS algorithm is based on a dramatically different technique. The general idea of our technique is that to solve the minimization problem, we first solve a related decision problem which is described as follows. Given a distance τ, can we find a representative skyline with k points such that the distance between a non-representative point and its nearest representative is no larger than τ? If that is the case, we refer to τ as a feasible solution for the decision problem. It is easy to see that the representative error of the optimal representative skyline should be the minimal feasible solution of the decision problem. Once the decision problem can be efficiently solved, then we can use a binary search procedure (over all possible τ) to find the minimal feasible solution. Below, we first present an algorithm to solve the decision problem.

3.1 Solve the Decision Problem

Here we devise a greedy algorithm to solve the decision problem. The idea is that the algorithm greedily covers the skyline points from p_1 to p_m using a circle with radius τ, and then verifies whether all points can be covered by k circles. If so, then the distance τ is a feasible solution. Otherwise, τ is non-feasible which will be proved in Lemma 1. Before we proceed further, we first present an algorithm to compute the radius of the smallest circle that covers all points in $\{p_i, \cdots, p_j\}$ for $i \leq j$ and centers at one of these points, which will be frequently invoked in the ARS algorithm.

The general idea of our algorithm to compute the radius of the smallest circle is as follows. Let $d(i, t)$ (for $t \in \{i + 1, \cdots, j\}$) be the distance between the point i and point t. Then, one can easily derive that $d(i, t)$ is an increasing function with increasing t. Similarly, we can find that $d(j, t)$ is an decreasing function with increasing t. Since the center of the circle must be a point from $\{p_i, \cdots p_j\}$, the radius $r(i, j)$ of the smallest circle must be the distance $d(i, t)$ or $d(j, t)$ for a certain t (the one with minimal $|d(i, t) - d(j, t)|$). Based on the monotonic properties of the distance functions $d(i, t)$ and $d(j, t)$, we are able to devise a binary search algorithm to search the smallest radius. The detailed description of our algorithm is shown in Algorithm 1.

It is worth emphasizing that our binary-search based technique (Algorithm 1) is completely different from the technique developed in [14] to compute the smallest circle, in

Algorithm 2. TestDistance(τ, k)

```
1: i ← 1 , i_1 ← 1;
2: for l = 1 to k do
3:     j ← m,j_1 ← m;
4:     while i_1 < j_1 do
5:         mid ← ⌊(i_1 + j_1)/2⌋;
6:         if r(i, mid) ≤ τ then
7:             i_1 ← mid + 1;
8:         else
9:             j_1 ← mid;
10:    if i_1 == m then
11:        return true;
12:    if r(i, mid) ≤ τ then
13:        i ← i_1 + 1, i_1 ← i;
14: return false;
```

which the authors present a *collative pass* technique to compute the optimal radiuses for all pair (i, j) in $O(m^2)$ time. Instead of computing all radiuses, we focus on computing $r(i, j)$ on demand (i.e., we invoke Algorithm 1 only when $r(i, j)$ is requested.). It is easy to derive that the worst-case time complexity of Algorithm 1 is $O(\log m)$.

Equipped with Algorithm 1, we are ready to present the greedy algorithm to solve the decision problem. Specifically, we detail the greedy algorithm in Algorithm 2. In Algorithm 2, the algorithm sequentially find k circles with radius τ to cover the points from p_1 to p_m (lines 2-13). For the l-th circle, denoted by circle-l, the algorithm determines the rightmost skyline point covered by the circle by using a binary search procedure (lines 4-9). If the rightmost skyline point covered by circle-l is p_m and $l \leq k$, then we know that we can use at most k circles with radius τ to cover all the skyline points, and thus τ is a feasible solution (lines 10-11). Otherwise, τ is non-feasible, and thereby the algorithm returns false (line 14). In addition, it is worth mentioning that in line 6, the algorithm invokes Algorithm 1 to compute the radius $r(i, mid)$, which takes $O(\log m)$ time complexity. Putting it all together, we can easily derive that the worst-case time complexity of Algorithm 1 is $O(k \log^2 m)$, and the space complexity of Algorithm 2 is $O(m)$. The following lemma shows the correctness of the algorithm.

Lemma 1. *Let $optE(m, k)$ be the optimal representative error of the DBRS problem. If $optE(m, k) > \tau$, then TestDistance(τ, k) returns false, otherwise returns true.*

Proof. Since $optE(m, k)$ is the optimal solution, it is impossible to use k circles with radius smaller than $optE(m, k)$ to cover all the skyline points. Therefore, if $optE(m, k) > \tau$, TestDistance(τ, k) must return false. It remains to show when $optE(m, k) \leq \tau$, TestDistance(τ, k) returns true. For the algorithm TestDistance(τ, k), we assume that p_{s_i} be the rightmost point covered by the i-th circle with radius τ, where s_i is the index of this rightmost point. Since $optE(m, k) \leq \tau$, there is a way of covering all skyline points by using k circles with radius no more than τ. If the first circle of this covers includes the points $\{p_1, \cdots, p_t\}$, then $t \leq s_1$ (by the fact of the greedy procedure). As a result, there exists a way of covering points $\{p_{s_1+1}, \cdots, p_m\}$ by using $k - 1$ circles with radius no more than τ. This is because we can remove

Algorithm 3. ARS(S, m, ϵ, k)

```
1: ub ← r(1, m), lb ← 0;
2: while lb + ε ≥ ub do
3:     τ ← ⌊(ub + lb)/2⌋;
4:     if TestDistance(τ, k) then
5:         ub ← τ;
6:     else
7:         lb ← τ;
8: return ub;
```

all the points in $\{p_{t+1}, \cdots, p_{s_1}\}$, and then use the optimal algorithm to covers all the points $\{p_{s_1+1}, \cdots, p_m\}$ by using $k-1$ circles with radius no larger than τ. The same argument can be applied to the second circle, the third circle, and so forth. Hence, the lemma is established.

3.2 The ARS Algorithm

Recall that to solve the DBRS problem, we have to find the minimal feasible solution of the decision problem. To that end, we first find that the decision problem satisfies the monotonic property. That is to say, for any $\tau_1 > \tau_2$, if τ_2 is a feasible solution, then τ_1 is also a feasible solution. Then, based on the monotonic property of the decision problem, we can apply a binary search procedure to find the minimal feasible solution.

The ARS algorithm is detailed in Algorithm 3. Since τ is a real value, we may not obtain the minimal feasible solution exactly by using binary search. Therefore, in Algorithm 3, we use a parameter ϵ to balance the tradeoff between the accuracy and running time of the algorithm. In the experiments, we will show how ϵ affects the accuracy and the running time of the algorithm. Note that in Algorithm 3, one can easily derive that $ub = r(1, m)$ and $lb = 0$ are the upper and lower bounds of τ, thus we can use these bounds as two starting points for the binary search procedure. In addition, it is straightforward to show that the ARS algorithm can guarantee a solution that is at most ϵ larger than the optimal solution. The worst-case time complexity of the ARS algorithm is $O(k \log^2 m \log(T/\epsilon))$. Here $T = ub - lb$, which is no more than the maximal distance between any two skyline points. Additionally, it is easy to verify that the space complexity of ARS is $O(m)$.

4 The New Exact Algorithm

Here we present an exact algorithm to solve the DBRS problem. For a pair (i, j) with $1 \leq i \leq j \leq m$, let $c(i, j)$ be the circle that covers points $\{p_i, \cdots, p_j\}$ and centers at one of these points, and $r(i, j)$ be the radius of the circle $c(i, j)$. Then, we can easily verify that the optimal representative error for the DBRS problem must be equal to $r(i, j)$ for a certain pair (i, j). Therefore, to find the optimal solution, a naive algorithm is to enumerate all such pairs. However, the naive algorithm is not efficient, because there are $O(m^2)$ pairs needed to be enumerated in the worst case. The challenge is how to search the optimal solution over all $O(m^2)$ pairs efficiently.

Algorithm 4. PSRS (S, m, k)

```
1: s_0 ← 1, s_k ← m;
2: for i = 1 to k − 1 do
3:    left ← s_{i−1}, right ← m;
4:    while left < right do
5:       mid ← ⌊(left + right)/2⌋;
6:       τ ← r(s_{i−1}, mid);
7:       if TestDistance(τ, k) then
8:          right ← mid;
9:       else
10:         left ← mid + 1;
11:   s_i ← right;
12: return min_{i=1}^{k} {r(s_{i−1}, s_i)};
```

To tackle this challenge, we shall propose an efficient algorithm based on a deep analysis of the problem. Specifically, we find that to compute the optimal solution of the DBRS problem, we only need to check k candidates, which is significantly smaller than $O(m^2)$. Below, we first introduce some useful notations, and then detail our techniques to achieve this goal.

For $i = 0, \cdots, k$, we let $s_i \in \{1, \cdots, m\}$ be the indices of k skyline points with $s_0 = 1$, $s_k = m$, and $s_i < s_{i+1}$ for $i = 1, \cdots, k-1$. We refer to p_{s_i} as the i-th breakpoint, because all the s_i for $i = 0, \cdots, k$ partition the entire skyline into k parts. Let $r(s_{i-1}, s_i)$ be the radius of the smallest circle denoted by $c(s_{i-1}, s_i)$ that covers all the points in $\{p_{s_{i-1}}, \cdots p_{s_i}\}$ and centers at one of these points. For completeness, we define $r(i, j) = -1$ whenever $j < i$. Then, we can compute the optimal representative error of the DBRS problem (i.e., $optE(m, k)$) based on the following lemma.

Lemma 2. *If the inequalities* $r(s_{i-1}, s_i - 1) < optE(m, k) \leq r(s_{i-1}, s_i)$ *for all* $i = 1, \cdots, k-1$ *hold, then* $optE(m, k) = \min_{i=1}^{k}\{r(s_{i-1}, s_i)\}$.

Proof. First, we show that $r(s_{k-1}, s_k)$ is a feasible solution for the decision problem. To see this, we partition of the skyline into k parts which are $\{p_{s_0}, \cdots, p_{s_1} - 1\}$, $\cdots$, $\{p_{s_{k-2}}, \cdots, p_{s_{k-1}} - 1\}$, and $\{p_{s_{k-1}}, \cdots, p_{s_k}\}$. Clearly, $\tau = \max\{\max_{i=1}^{k-1}\{r(s_{i-1}, s_i - 1), r(s_{k-1}, s_k)\}$ is a feasible solution for the decision problem, i.e., $\tau \geq optE(m, k)$. Since $r(s_{i-1}, s_i - 1) < optE(m, k)$ for all $i = 1, \cdots, k-1$ hold, we have $\max_{i=1}^{k-1}\{r(s_{i-1}, s_i - 1) < r(s_{k-1}, s_k)$, and thereby $\tau = r(s_{k-1}, s_k)$ which is feasible. Second, we let $\tau_{\min} = \min_{i=1}^{k}\{r(s_{i-1}, s_i)\}$. Since $optE(m, k) \leq r(s_{i-1}, s_i)$ for all $i = 1, \cdots, k-1$ hold, we have $optE(m, k) \leq \tau_{\min}$. To prove the lemma, it is sufficient to show that $optE(m, k) < \tau_{\min}$ is impossible. We assume to the contrary that $optE(m, k) < \tau_{\min}$. Suppose that $\tau_{\min} = r(s_{i-1}, s_i)$ for a certain i. It is easy to see that for the optimal solution, the best possible for the rightmost point covered by the first circle is $p_{s_1 - 1}$. Similarly, we can find that the best possible for the rightmost point covered by the i-th circle is $p_{s_i - 1}$ for $i = 1, \cdots, k-1$. Following this logic, for the optimal solution, the points $\{p_{s_{k-1}}, \cdots, p_{s_k}\}$ must be covered by a circle, which contradicts to $optE(m, k) < \tau_{\min}$. This completes the proof.

Table 1. Datasets (m is the number of skyline points)

Dataset	m	Property
ID 1	14,735	Independent dataset
ID 2	1,087,893	Independent dataset
ACD 1	18,576	Anti-Correlated dataset
ACD 2	1,142,369	Anti-Correlated dataset
Urban	1,662	Real-world dataset

Based on Lemma 2, we only need to find all the s_i for $i = 1, \cdots, k-1$ that meet the condition $r(s_{i-1}, s_i - 1) < optE(m, k) \leq r(s_{i-1}, s_i)$. Once we determine all such s_i, then $r(s_{i-1}, s_i)$ for each $i = 1, \cdots, k-1$ is a candidate for the optimal solution, and we only need to take the minimal over all $r(s_{i-1}, s_i)$. We can determine all the s_i by sequentially invoking a binary search procedure based on the monotonic property of the decision problem. Specifically, if $r(s_{i-1}, s_i)$ is a feasible solution for the decision problem, then $r(s_{i-1}, s_i) \geq optE(m, k)$ and for all $s > s_i$ we have $r(s_{i-1}, s) \geq r(s_{i-1}, s_i)$. As a result, we are able to devise a binary search algorithm to determine the index of the first breakpoint s_1 such that $r(s_0, s_1 - 1) < optE(m, k) \leq r(s_0, s_1)$, and then given s_1, we can use the same binary search procedure to find s_2, and so on. The detailed description of our algorithm is outlined in Algorithm 4. We can easily derive that the time and space complexity of Algorithm 4 are $O(k^2 \log^3 m)$ and $O(m)$ respectively. The correctness of Algorithm 4 can be guaranteed by Lemma 2.

5 Experiments

In this section, we conduct comprehensive experiments to evaluate the proposed algorithms. To this end, we use the state-of-the-art DP based algorithm [14], called DPRS, as the baseline. We use the running time and the representative error as the metrics, and then we compare our algorithms ARS and PSRS with the baseline algorithm DPRS using these metrics. For the ARS algorithm, we set the parameter $\epsilon = 0.01$, unless otherwise specified. All algorithms are implemented in C++. All the experiments are conducted on a computer with 3.20GHz Intel(R) Core(TM) i5-3470 CPU and 6GB memory running Windows 8.1 operation system.

Datasets. In the experiments, we use five datasets. The first four datasets, named ID 1, ID 2, ACD 1 and ACD 2 respectively, are the synthetic datasets which are generated based on the method proposed in [2]. Specifically, for the ID 1 and ID 2 datasets, the (x, y)-coordinates of the skyline points are generated based on a uniform distribution. We generate 14,735 and 1,087,893 skyline points for the ID 1 and ID 2 datasets respectively. The ACD 1 and ACD 2 datasets are the anti-correlated dataset where the points are good in one dimension but they are bad in the other dimension. We use the method proposed in [2] to generate the ACD 1 and ACD 2 datasets which contain 18,576 and 1,142,369 skyline points respectively. The last dataset, called Urban, is a real-world dataset including 1,662 skyline points, which is previously used in urban computing community [20]. We download this dataset from `http://research.microsoft.com/en-us/projects/urbancomputing/`. In our experiments, we extract two dimensions of

Table 2. Running time of different algorithms in ID 1 dataset (in second)

k	5	10	15	20	25	30
DPRS	327.314	381.451	436.307	490.713	545.053	580.225
ARS	0.024	0.025	0.029	0.037	0.041	0.047
PSRS	0.023	0.024	0.024	0.027	0.028	0.030

Table 3. Running time of different algorithms in ID 2 dataset (in second)

k	5	10	15	20	25	30
DPRS	-	-	-	-	-	-
ARS	2.482	2.485	2.50	2.501	2.502	2.518
PSRS	2.50	2.528	2.531	2.602	2.634	2.66

each data point in the original dataset, where the first dimension represents the time, and the second dimension denotes the traffic flow, and then we compute the skyline. For a fair comparison, we normalize the (x, y)-coordinates of the points in all the datasets into the range $[0, 100]$. The detailed information of our datasets are shown in Table 1.

Efficiency. Here we report the running time of the DPRS, ARS, and PSRS algorithms. The results for varying k in all datasets are shown in Tables 2, 3, 4, 5, and 6 respectively. As can be seen, both the ARS and PSRS algorithms are significantly faster than the DPRS algorithm (the baseline algorithm), and the ARS algorithm is slightly more efficient than the PSRS algorithm. For example, in the ID 1 dataset (Table 2), we can see that the ARS and PSRS algorithms are 17,548 and 13,807 times faster than the DPRS algorithm on average. That is to say, the proposed ARS and PSRS algorithms are at least four orders of magnitude faster than the state-of-the-art DPRS algorithm in medium-sized datasets. Even in the smallest dataset Urban (Table 6), the ARS and PSRS algorithms are at least two orders of magnitude faster than the DPRS algorithm. In addition, in our two large datasets ID 2 and ACD 2 (Tables 3 and 5), the DPRS cannot get the solution in one day, while our ARS and PSRS algorithms can compute the solution in around 2.5 second. As desired, over all datasets, the running time of all the algorithms increase with increasing k. These results are consistent with the theoretical analysis presented in Sections 3 and 4.

Scalability. Here we test the scalability of the ARS and PSRS algorithms. To this end, we vary the number of skyline points from 10,000 to 1,000,000, and test the running time of our algorithms in ID 2 and ACD 2 datasets. We set $k = 10$, and similar results can also be observed for other k values. The results are depicted in Fig. 1. From Fig. 1, we can see that both the ARS and PSRS algorithms scale very well. When the datasets is large than 100,000, the increasing speed of the running time of the ARS and PSRS algorithms are lower than the increasing speed of the number of skyline points. The reason is that the time complexity of our algorithms are sublinear. The results further confirm that our algorithms can be applied to handle large datasets.

Effect of the Parameter ϵ. Here we study how the parameter ϵ affects the efficiency and effectiveness of the ARS algorithm. For this purpose, we vary ϵ from 0.001 to 0.1,

Table 4. Running time of different algorithms in ACD 1 dataset (in second)

k	5	10	15	20	25	30
DPRS	366.957	414.995	481.716	510.982	568.816	609.458
ARS	0.027	0.028	0.030	0.034	0.035	0.039
PSRS	0.028	0.030	0.031	0.046	0.052	0.063

Table 5. Running time of different algorithms in ACD 2 dataset (in second)

k	5	10	15	20	25	30
DPRS	-	-	-	-	-	-
ARS	2.727	2.750	2.751	2.753	2.768	2.794
PSRS	2.735	2.766	2.788	2.816	2.854	2.930

Table 6. Running time of different algorithms in Urban dataset (in second)

k	5	10	15	20	25	30
DPRS	2.371	3.546	4.672	7.768	8.881	10.053
ARS	0.015	0.015	0.015	0.015	0.016	0.016
PSRS	0.015	0.016	0.016	0.016	0.017	0.019

and use the running time and the representative error as two metrics to evaluate the efficiency and effectiveness of the ARS algorithm, respectively. Also, we set $k = 10$, and similar results can be observed for other k values. We test the ARS algorithm in ID 2 and ACD 2 datasets. The results of running time with varying ϵ are shown in Fig. 2. As can be seen, the running time decreases with increasing ϵ. When $\epsilon \leq 0.01$, the running time of the ARS algorithm dramatically decreasing with increasing ϵ. However, when $\epsilon > 0.01$, the curves are relatively smooth. The reason is that if ϵ is very small, then the binary search procedure in the ARS algorithm will take a relatively long time to converge to the near-optimal approximate solution. However, when ϵ is large, the binary search procedure converges quickly.

We report the results of the representative error of the ARS algorithm with different ϵ in Fig. 3. From Fig. 3, we can find that the representative error increases with increasing ϵ as desired. This is because ϵ controls the accuracy of the ARS algorithm. A large ϵ results in a low accuracy. Also, we can see that when $\epsilon = 0.01$, the representative error of the ARS algorithm is close to the exact representative error.

6 Related Work

Skyline Computation. The skyline operator was first introduced into database community by Borzsonyi et al. [2] in 2001, although the skyline computation problem can date back to 1975 [7]. Since then, a large number of skyline computation algorithms have been proposed. Previous results mainly cover three aspects: 1) centralized skyline query processing, including the BNL algorithm (block nested loop) [2], the SFS algorithm (sort filter skyline) [3], the divide and conquer algorithm [2], the bitmap algorithm [13],

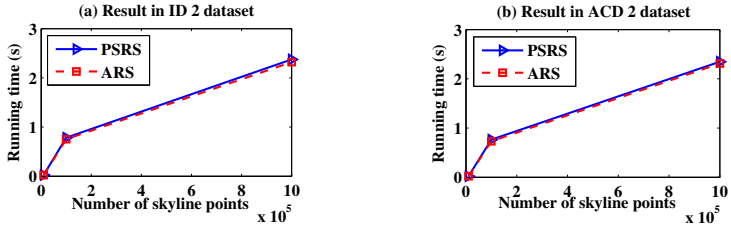

Fig. 1. Scalability of the PSRS and ARS algorithm

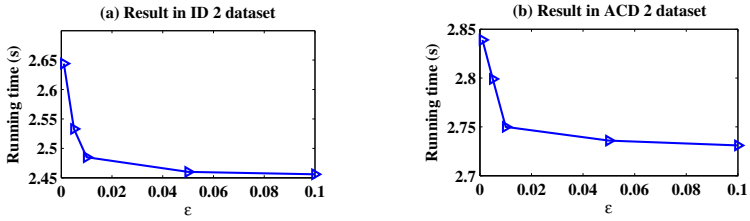

Fig. 2. Running time vs. ϵ (for the ARS algorithm)

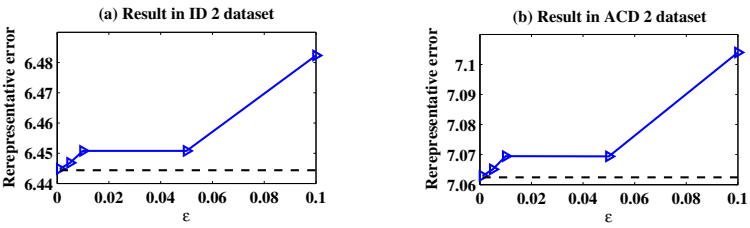

Fig. 3. Representative error vs. ϵ (dashed line denotes the exact representative error)

the index-based algorithm [13], the NN algorithms (Nearest Neighbor) [6], as well as the BBS (branch and bond skyline) algorithm [11]; 2) distributed skyline query processing, including skyline computing in the traditional distributed environment [1], skyline computing in the mobile distributed environment [5], and skyline computing in peer-to-peer networks [17,18], and so on; 3) Other skyline computing problems, including size estimation of the skyline [4], skyline query processing in any subspace [16], skyline query processing in all subspace [19,11], and skyline query processing in data flow [15], and so on.

Representative Skyline Computation. Since the traditional skylines has too many skyline points, in decision-making related applications, a better way is to find a small number of points that can best represent the entire skyline. Motivated by this, many new definitions of representative skyline have been proposed recently. Notable definitions include the domination-based representative skyline which is proposed by Lin et al. [8]. The goal of the domination-based skyline is to find k skyline points to maximize the number of points that can be dominated by any of the selected k skyline points.

Subsequently, Tao et al. [14] introduced a distance-based representative skyline definition, in which the goal is to minimize the distance between a non-representative skyline point and its closest neighbor. Sarma et al. [12] presented a representative skyline model based on the users' preference distributions, aiming at finding k representative skyline points such that the probability that a random user would click on one of them is maximized. More recently, Magnani et al. [9] proposed a new representative skyline definition which takes both significance and diversity of the data points into account. In this paper, we focus on developing efficient algorithms for computing the distance-based representative skyline in 2D space.

7 Conclusion

Representative skyline has a number of application in database community, especially in multi-objective decision making and real-time online service. A notable definition of representative skyline is the distance-based representative skyline (DBRS). However, in 2D space, the state-of-the-art algorithm for DBRS computation is based on dynamic programming (DP) which is costly for large scale datasets due to the quadratic time overhead. In this paper, we propose two much more efficient algorithms, called PSRS and ARS respectively, based on a carefully-designed parametric search technique. We show, in theory and experiments, that both PSRS and ARS algorithms are very efficient which are at least four orders of magnitude faster than the state-of-the-art algorithm in medium-sized datasets. There are several future directions needed to further investigate. First, it would be interesting to devise parallel and distributed counterparts for our algorithms. Second, the proposed techniques are tailored to two-dimensional data. An immediate question is how to extend our techniques to compute the DBRS in high dimensional data. Finally, it would also be interesting to further optimize our algorithms by developing non-trivial pruning techniques in the parametric search procedure.

Acknowledgements. The work was supported in part by (i) NSFC Grants (61402292, 61170076, U1301252, 61033009) and Natural Science Foundation of SZU (grant no. 201438); (ii) Research Grants Council of the Hong Kong SAR, China, 14209314 and 418512; (iii) China 863 (no. 2012AA010239) and Guangdong Key Laboratory Project (2012A061400024); (iv) National Key Technology Research and Development Program of the Ministry of Science and Technology of China2014BAH28F05. Dr. Rong-Hua Li is the corresponding author of this paper.

References

1. Balke, W.-T., Güntzer, U., Zheng, J.X.: Efficient distributed skylining for web information systems. In: Bertino, E., Christodoulakis, S., Plexousakis, D., Christophides, V., Koubarakis, M., Böhm, K. (eds.) EDBT 2004. LNCS, vol. 2992, pp. 256–273. Springer, Heidelberg (2004)
2. Börzsönyi, S., Kossmann, D., Stocker, K.: The skyline operator. In: ICDE, pp. 421–430 (2001)
3. Chomicki, J., Godfrey, P., Gryz, J., Liang, D.: Skyline with presorting. In: ICDE, pp. 717–719 (2003)

4. Godfrey, P.: Skyline cardinality for relational processing. In: Seipel, D., Turull-Torres, J.M.a. (eds.) FoIKS 2004. LNCS, vol. 2942, pp. 78–97. Springer, Heidelberg (2004)
5. Huang, Z., Jensen, C.S., Lu, H., Ooi, B.C.: Skyline queries against mobile lightweight devices in manets. In: ICDE, p. 66 (2006)
6. Kossmann, D., Ramsak, F., Rost, S.: Shooting stars in the sky: An online algorithm for skyline queries. In: VLDB, pp. 275–286 (2002)
7. Kung, H.T., Luccio, F., Preparata, F.P.: On finding the maxima of a set of vectors. J. ACM 22(4), 469–476 (1975)
8. Lin, X., Yuan, Y., Zhang, Q., Zhang, Y.: Selecting stars: The k most representative skyline operator. In: ICDE, pp. 86–95 (2007)
9. Magnani, M., Assent, I., Mortensen, M.L.: Taking the big picture: representative skylines based on significance and diversity. VLDB J. 23(5), 795–815 (2014)
10. Papadias, D., Tao, Y., Fu, G., Seeger, B.: An optimal and progressive algorithm for skyline queries. In: SIGMOD, pp. 467–478 (2003)
11. Pei, J., Jin, W., Ester, M., Tao, Y.: Catching the best views of skyline: A semantic approach based on decisive subspaces. In: VLDB, pp. 253–264 (2005)
12. Sarma, A.D., Lall, A., Nanongkai, D., Lipton, R.J., Xu, J.J.: Representative skylines using threshold-based preference distributions. In: ICDE, pp. 387–398 (2011)
13. Tan, K.-L., Eng, P.-K., Ooi, B.C.: Efficient progressive skyline computation. In: VLDB, pp. 301–310 (2001)
14. Tao, Y., Ding, L., Lin, X., Pei, J.: Distance-based representative skyline. In: ICDE, pp. 892–903 (2009)
15. Tao, Y., Papadias, D.: Maintaining sliding window skylines on data streams. IEEE Trans. Knowl. Data Eng. 18(2), 377–391 (2006)
16. Tao, Y., Xiao, X., Pei, J.: SUBSKY: efficient computation of skylines in subspaces. In: ICDE, p. 65 (2006)
17. Wang, S., Ooi, B.C., Tung, A.K.H., Xu, L.: Efficient skyline query processing on peer-to-peer networks. In: ICDE, pp. 1126–1135 (2007)
18. Wu, P., Zhang, C., Feng, Y., Zhao, B.Y., Agrawal, D.P., El Abbadi, A.: Parallelizing skyline queries for scalable distribution. In: Ioannidis, Y., et al. (eds.) EDBT 2006. LNCS, vol. 3896, pp. 112–130. Springer, Heidelberg (2006)
19. Yuan, Y., Lin, X., Liu, Q., Wang, W., Yu, J.X., Zhang, Q.: Efficient computation of the skyline cube. In: VLDB, pp. 241–252 (2005)
20. Zhang, F., Wilkie, D., Zheng, Y., Xie, X.: Sensing the pulse of urban refueling behavior. In: UbiComp, pp. 13–22 (2013)