Discovering Representative Skyline Points over Distributed Data

Akrivi Vlachou[1⋆], Christos Doulkeridis[1,2⋆⋆], and Maria Halkidi[2]

[1] Norwegian University of Science and Technology (NTNU), Norway
[2] University of Piraeus, Greece
vlachou@idi.ntnu.no,{cdoulk,mhalk}@unipi.gr

Abstract. Skyline queries help users make intelligent decisions over complex data. The main shortcoming of skyline queries is that the cardinality of the result set is not known a-priori. To overcome this limitation, the representative skyline query has been proposed, which retrieves a fixed set of k skyline points that best describe all skyline points. Even though the representative skyline has been studied before in centralized environments, this is the first paper that addresses efficient computation of the representative skyline in distributed systems. The distributed nature of the environment makes the task of discovering truly representative skyline points even more challenging. In this paper, we propose a novel framework for discovering the representative skyline over distributed data sources. Our experimental study demonstrates the efficiency and effectiveness of our framework.

1 Introduction

Skyline queries [1] constitute a powerful tool for multi-objective optimization, especially in the case of multiple and conflicting criteria. An important shortcoming of skyline queries is that the size of the result set is not fixed, but largely depends on various factors such as the data distribution or the dimensionality of the data space. Thus, in contrast to other popular query types, such as top-k queries [3, 5] that return results of expected size, the cardinality of skyline set is unrestricted and can sometimes be comparable to the size of the complete data set. To alleviate this shortcoming, centralized approaches that select a restricted set of *representative skylines* have been proposed [7, 11].

As data management becomes inherently distributed due to massive content generation at disparate locations, the importance of distributed query processing is even more evident. Lately, this is also intensified by the advent of large-scale distributed data centers and cloud computing infrastructures. In such setups, servers store portions of the data set and the objective is to support efficient and effective techniques for query processing and advanced data analysis.

In this paper, we address for the first time the problem of discovering a set of k skyline representatives over distributed data, which is even more challenging

⋆ A. Vlachou was partially supported by the Greek State Scholarship Foundation.
⋆⋆ C. Doulkeridis was supported under the Marie-Curie IEF grant number 274063.

than the centralized representative skyline query due to the lack of global knowledge. As skyline points are equivalent by definition, any representative skyline query uses an error metric that captures the loss in the expressiveness of the skyline set due to the absence of the non-representative skyline points. Thus, the representative skyline query defines an optimization problem that aims to retrieve the k skyline points that minimize the error metric. Different error metrics have been proposed for representative skyline queries, such as dominance [7] and distance-based [11] error metric. Assuming a generic distributed setup where a set of N servers store portions of the entire data set autonomously, we introduce a novel framework for distributed skyline representative algorithms which supports any error metric and retrieves a representative skyline set of low representation error by only considering a fraction of the distributed data.

More precisely, our framework encompasses a baseline approach as well as two alternative efficient algorithms. The *distributed skyline algorithm* is used as a baseline and incorporates representative skyline computation in a distributed skyline query by transferring all local skyline points to the coordinating server. In case of error metrics that are not influenced by dominated points, such as [11], the distributed skyline algorithm returns exactly the same representative points as if the query was executed on all data by a centralized algorithm. Furthermore, we prove that any algorithm that transfers fewer local data than the distributed skyline query may report non-skyline points as representative skyline points, if only a single communication phase is employed.

Motivated by this observation, we propose the *distributed skyline representative algorithm* that relies on two communication phases in order to reduce the transferred data. In the first phase, each individual server discovers its k local representative skyline points. The local representatives are transmitted to the coordinating server, which extracts an initial set of k global skyline representatives. At the second communication phase, the currently defined global representatives are forwarded back to servers, to be tested for dominance by other local skyline points. The identified set of dominating points is then sent to the coordinating server, which then re-applies the representative skyline algorithm to extract the final set of k representative skyline points. Finally, we introduce the *distributed error-based representative* algorithm that processes the query in the same spirit as the distributed skyline representative algorithm, but also exploits the information about the value of error metric at local level to reduce further the induced error of the representative skyline set.

2 Related Work

Restricting the skyline cardinality is motivated by the fact that the skyline cardinality increases with the data set dimensionality. To deal with this *dimensionality curse*, one possibility is to restrict the cardinality of the result set, by choosing k skyline points out of the entire set. Towards this goal, the authors in [2] propose the *k-dominant* skyline query. The authors relax the idea of dominance to k-dominance, in order to increase the probability of one point dominating

another point, thereby restricting the skyline cardinality. *Skyline ordering* [8] is an approach that produces arbitrary size constrained skyline sets by employing skyline-based partitioning on the data set.

Selecting *representative skyline points* in centralized domains has recently attracted significant attention for retrieving exactly k points from the skyline set. In [7], the authors study the problem of selecting k skyline points, so that the number of points dominated by at least one of these k skyline points is maximized. In [11], an approach is presented for retrieving k representative skyline points, which are defined as the set of k points that minimize the maximum distance between a non-representative skyline point and its nearest representative. In [10], representative skylines are studied under the assumption that user preferences are expressed as thresholds. The thresholds indicate the worst value on each attribute that is acceptable from each user. The proposed approach relies on the probability distribution of the user's thresholds. Preference-based representative skyline queries are out of the scope of this paper.

This is the first paper that studies representative skyline queries in distributed systems. However, several approaches have been proposed for efficient skyline processing in distributed environments; a detailed survey of distributed skyline processing can be found in [6]. For example, subspace skyline computation over peer-to-peer network has been studied in [12, 13]. Cui *et al.* [4] proposed the PaDSkyline algorithm for skyline query processing in a generic distributed environment. In [14], a feedback-based distributed skyline (FDS) algorithm is proposed, which aims to minimize the bandwidth consumption. However, the aforementioned papers focus on the efficient computation of the skyline query, not the representative skyline query.

3 Preliminaries and Problem Statement

Preliminaries. In our system model, a set of N servers S_i participate in the distributed skyline computation, while a coordinator server S_C is responsible for communication with the servers in order to produce the desired representative skyline set. Data is distributed in the sense of horizontal partitioning, thus each server S_i stores locally a set of points P_i. The entire data set P is the union of all sets of points P_i stored locally at any server S_i ($P = \bigcup P_i$, $P_i \bigcap P_j = \emptyset$). A representative skyline query is initiated by the coordinator. In the following, we provide the necessary definitions and preliminaries.

Given a data set P on a data space defined by a set of d dimensions $\{d_1, \ldots, d_d\}$, a point $p \in P$ is represented as p=$\{p[1], \ldots, p[d]\}$ where $p[i]$ is the value on dimension d_i. Without loss of generality, we assume that $\forall d_i : p[i] \geq 0$, and that smaller values are preferable.

Definition 1. *(Skyline set) A point $p \in P$ dominates another point $q \in P$, denoted as $p \prec q$, if (1) on every dimension d_i, $p[i] \leq q[i]$; and (2) on at least one dimension d_j, $p[j] < q[j]$. The skyline $\mathcal{S}(P)$ is a set of points that are not dominated by any other point in P.*

Consider the example in Fig. 1, where each point represents a hotel and the y-dimension represents the price of a room, while the x-dimension captures the distance of the hotel to the beach. A hotel dominates another hotel because it is cheaper and closer to the beach. Thus, the skyline points (a, i, m and k) are the best possible trade-offs between price and distance from the beach.

Problem statement. Unfortunately, as the dimensionality of the data set grows, the skyline operator loses its discriminating power and returns a large fraction of the data. The huge size of the result set hinders decision-making and motivates the ranking of skyline points. Therefore, users prefer to retrieve k representative points instead of the whole skyline set. The representative skyline points are chosen to best describe the tradeoffs among different dimensions offered by the full skyline. As skyline points are equivalent by definition, an error metric is defined to capture the *representativeness* of a set of k skyline points.

Definition 2. *(Representative skyline set) Given an integer k, the representative skyline of a data set P is a set $\mathcal{K}$ of k skyline points of $\mathcal{S}(P)$ that minimizes the error metric $Er(\mathcal{K})$.*

As we will elaborate in the following, different definitions of the error metric for the representative skyline have been proposed: dominance-based error metric [7] and distance-based error metric [11]. Both error metrics are supported by our distributed framework. Definition 2 leads to the following problem statement of this paper.

Definition 3. *(Distributed representative skyline set) Given a distributed data set $P = \bigcup P_i$, compute its representative skyline set $\mathcal{K}$ of size k.*

Dominance-based representative skyline. In [7], the representative skyline set is defined based on the dominated points. More precisely, the authors quantify the concept of representativeness by the total number of (distinct) data points dominated by one of the k representative skyline points. In other words, the k most representative skyline points are the ones that minimize the number of the data points that are not dominated by any representative point. Thus, the error metric is defined as:

$$Er(\mathcal{K}) = \{|\{p\}| : p \in P, p \notin \mathcal{K}, \nexists p' \in \mathcal{K} : p' \prec p\} .$$

For example, Fig. 2(a) depicts a data set P of hotels, along with its skyline points $\mathcal{S}(\mathcal{P})$. This data set contains 6 skyline points depicted with circles on a line. In addition, the representative skyline points that are derived from the dominance-based algorithm for k=3 are depicted with squares.

The problem is shown to be NP-hard when the dimensionality is 3 or more and it can be approximately solved by a polynomial time greedy algorithm. The proposed greedy algorithm starts by computing the skyline set and the representative error of each skyline point, i.e., the number of data points that are dominated by each skyline point. The algorithm picks as the first representative skyline point the skyline point that has the highest number of dominated points. After removing the data points that are dominated by the first representative skyline point, the representative error of each skyline point is re-computed and

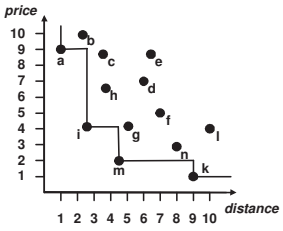
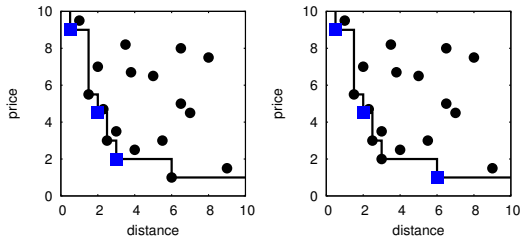


Fig. 1. Skyline example.



(a) Dominance-based. (b) Distance-based.

Fig. 2. Example of representative skyline.

again the skyline point with the maximum value is picked. This is repeated until k skyline points are selected. To overcome the high memory requirements of the greedy algorithm, a probabilistic counting technique can be applied for estimating the number of distinctly dominated data points. This leads to an index-based randomized algorithm for finding the representative skyline points.

Distance-based representative skyline. The error metric $Er(\mathcal{K})$ for the distance-based representative skyline [11] is defined as:

$$Er(\mathcal{K}) = \max_{\forall p \in \mathcal{S}(P)-\mathcal{K}}(\min_{\forall p' \in \mathcal{K}} d(p, p')),$$

where $d(p, p')$ is the Euclidean distance between points p and p'. Intuitively, a distance-based representative skyline set is good, if for every non-representative skyline point, there exists a representative skyline point nearby.

Fig. 2(b) depicts an example of the distance-based representative points $\mathcal{K}$ for illustrative purposes. The k=3 representative points are depicted with squares, while the skyline points depicted with circles on a line.

For dimensionality at least 3 the problem is NP-hard, thus the authors propose a greedy algorithm [11], namely I-greedy, to compute the distance-based representative. I-greedy assumes a multidimensional index on the data set and uses the concept of max-rep-dist for computing the representatives. Given a subtree in the R-tree, its max-rep-dist is a value that upper-bounds the representative distance of any potential skyline point p in this subtree. Initially, it takes as input an initial set $\mathcal{K}$ containing an arbitrary skyline point, which is used as a representative point. For example, this point can be the point with the smallest x-coordinate. I-greedy maintains the entries of the R-tree in a sorted list. In each iteration, I-greedy processes the entry E with the largest max-rep-dist. If the next entry is dominated by at least one point retrieved so far, the entry is discarded. Otherwise, I-greedy searches for the entry with the smallest L_1 distance to the origin among all entries in the sorted list whose min-corners dominate E. If such an entry exists, it must be an intermediate entry, so the entries of its child node are inserted, if they are not dominated by any point retrieved so far. If such an entry does not exist, then E is processed. If E is a point, it is inserted to $\mathcal{K}$ as the next representative skyline point. Otherwise, the entries of its child node are inserted in the list, if they are not dominated by any point retrieved so far.

Algorithm 1 *Distributed skyline algorithm DSA*

1: **INPUT:** k, Coordinator S_C, Servers S_i
2: **OUTPUT:** Representative skyline $\mathcal{K}$
3: **for** $(\forall S_i : i \in [1, N])$ **do**
4: $\quad \mathcal{S}(P_i) \leftarrow S_i.\underline{\text{skyline}}()$
5: **end for**
6: $\mathcal{K} \leftarrow S_C.\underline{\text{representative}}(\bigcup \mathcal{S}(P_i))$
7: **return** $\mathcal{K}$

4 Distributed Representative Skyline Algorithms

In this section, we present our framework that encompasses two algorithms for discovering the representative skyline points over distributed data. Our generic framework is parameterized by a centralized skyline representative algorithm that is executed locally at the participating servers. Any such algorithm for local skyline representative computation can be plugged in our framework. Currently, we have incorporated in our framework the error metrics of two existing skyline representative algorithms studied in the related work: distance-based representative [11] and dominance-based representative [7].

4.1 Distributed skyline algorithm (DSA)

In a generic distributed system, processing the representative skyline query can be performed by integrating the representative skyline computation in a distributed skyline algorithm. Algorithm 1, termed *DSA*, serves as a baseline and adheres to this strategy to produce a representative skyline set.

DSA is processed at S_C by first sending a skyline query to all servers S_i, which in turn process the query locally over their data P_i (line 4). Then, each server S_i reports its *local skyline set* $\mathcal{S}(P_i)$ to S_C. Similar to the case of distributed skyline query processing, a centralized algorithm for finding the representative skyline set, such as [7, 11], is processed at the coordinator server S_C, in order to obtain the *representative skyline set* $\mathcal{K}$ (line 6).

An important property of the skyline operator is that the skyline set of a distributed data set P is a subset of the union of the local skyline sets of all partitions $\mathcal{S}(P) \subseteq \bigcup \mathcal{S}(P_i)$. This property of the skyline set leads to an interesting observation about the *DSA* algorithm. As long as the error metric used for defining the representative skyline is not influenced by non-skyline points of the data set, the retrieved representative skyline set of *DSA* is equivalent to the representative skyline set of the entire data set $P = \bigcup P_i$. Moreover, this is accomplished without requiring the transfer of all local data points P_i, but only the local skyline points $\mathcal{S}(P_i)$. Notice that the distance-based error metric satisfies the afore-described observation, therefore *DSA* produces the identical result of the centralized distance-based representative algorithm.

However, *DSA* has an important drawback; it needs to transfer the complete local skyline sets to the coordinator. Under certain circumstances, depending on

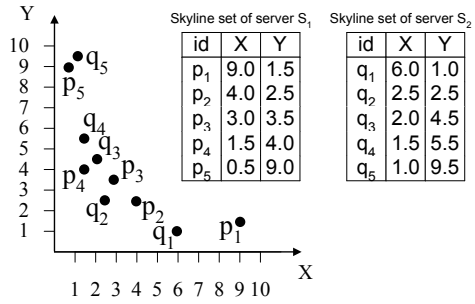


Fig. 3. Example of Lemma 1.

the dimensionality or data distribution, the local skyline sets $\mathcal{S}(P_i)$ are comparable in size to the local data sets P_i. Obviously, this leads to increased network traffic, which is undesirable especially in the case of bandwidth-constrained networks. Motivated by this shortcoming, we introduce an algorithm that produces the representative skyline set $\mathcal{K}$ by transferring only a limited number of points, which is independent of the actual cardinality of local skyline sets.

4.2 Distributed skyline representative algorithm (DSR)

In the following, we first show that any distributed approach that transfers fewer data points than the local skyline points requires two communication phases, in order to ensure that the representative skyline set is valid, i.e., all representative skyline points belong to the global skyline set. Then, we describe in detail the proposed algorithm, termed *DSR*.

Two communication phases. The design of the *DSR* algorithm is guided by the observation that we do not wish to transfer local skyline sets to the coordinator, as this would result in unrestricted size of transferred points. Consequently, our premise is to transfer to the coordinator only a fraction of the local skyline points $\mathcal{S}(P_i)$, namely only the local representative skyline points $\mathcal{K}_i$. However, the following lemma shows that this method does not guarantee that the produced representative points $\mathcal{K}$ are actually global skyline points $\mathcal{S}$.

Lemma 1. *A distributed skyline representative algorithm that produces a representative skyline set $\mathcal{K}$ over the union of local representative skyline sets $\bigcup \mathcal{K}_i$ may result in non-skyline points p, i.e., $p \in \mathcal{K} \bigwedge p \notin \mathcal{S}$.*

Proof. It suffices to construct an example where the algorithm will falsely report a dominated point as representative skyline point. We use the distance-based error metric, but a similar example can be constructed for the dominance error metric. Consider the example of Fig. 3, where the skyline sets of two servers S_1 and S_2 are depicted. Assume that the representative skyline set is requested for k=3. Applying the distance-based representative algorithm on $\mathcal{S}(P_1)$ and $\mathcal{S}(P_2)$ produces the sets $\mathcal{K}_1$=$\{p_1, p_3, p_5\}$ and $\mathcal{K}_2$=$\{q_1, q_3, q_5\}$ respectively. It is easy to

Algorithm 2 *Distributed Skyline Representative DSR*

1: **INPUT:** k, Coordinator S_C, Servers S_i
2: **OUTPUT:** Representative skyline $\mathcal{K}$
3: **for** $(\forall S_i : i \in [1, N])$ **do**
4: $\quad \mathcal{K}_i \leftarrow S_i$.representative$(\mathcal{P}_i)$
5: **end for**
6: $\mathcal{K}' \leftarrow S_C$.representative$(\bigcup \mathcal{K}_i)$
7: **for** $(\forall S_i : i \in [1, N])$ **do**
8: $\quad \mathcal{D}_i \leftarrow S_i$.dominate$(\mathcal{K}')$
9: **end for**
10: $\mathcal{K} \leftarrow S_C$.representative$((\bigcup \mathcal{D}_i) \bigcup \mathcal{K}')$
11: **return** $\mathcal{K}$

see that q_1 dominates p_1, and p_5 dominates q_5, thus S_C will take as input the set $\{p_3, p_5, q_1, q_3\}$ to produce $\mathcal{K}$. Obviously, since k=3 at least one of p_3, q_3 belongs to $\mathcal{K}$. However, q_2 dominates p_3 and p_4 dominates q_3, therefore the algorithm falsely reports a dominated point as representative skyline point.

Lemma 1 practically means that no algorithm that is based solely on transfer of local representative skyline points to the coordinator can guarantee that the representative points $\mathcal{K}$ belong to the global skyline set $\mathcal{S}$, i.e., $\mathcal{K} \subset \mathcal{S}$. Consequently, we propose the *DSR* algorithm that employs two communication phases in order to guarantee that the representative skyline set consists of skyline points. In the first phase, the coordinator requests from each server the representative skyline set based on the locally stored data. Then, a centralized algorithm for representative skyline computation is applied on the union of local representative skyline sets to produce a set of representative skyline points. In the second phase, the produced representative skyline points are sent to all servers, and each server sends back a set of points $\mathcal{D}_i$ that consists of the local skyline points that dominate at least one representative skyline point. Finally, the coordinator applies again the skyline representative algorithm to produce the final set of representative skyline points $\mathcal{K}$. *DSR* improves the efficiency of *DSA* in terms of communication by requesting only the local representative skyline points.

Algorithmic description and correctness. *DSR* is described in Algorithm 2. First, each server S_i $(i \in [1, N])$ executes a skyline representative query[3] on the locally stored data (P_i) to produce a set $\mathcal{K}_i$ of k local representative skyline points (line 4). The coordinator assembles the sets $\mathcal{K}_i$ $(i \in [1, N])$ and produces a new set of k representative skyline points (line 6), denoted as $\mathcal{K}'$, by applying the centralized skyline representative algorithm. Then, the coordinator sends the set $\mathcal{K}'$ to all servers S_i. Each server S_i computes all local skyline points $\mathcal{D}_i$ that dominate at least one of the points in $\mathcal{K}'$ (line 8). The coordinator merges its set $\mathcal{K}'$ together with the union of sets of local points $\mathcal{D}_i$ that dominate points of $\mathcal{K}'$, and applies the representative skyline algorithm (line 10) to produce the final set $\mathcal{K}$, which is reported to the user (line 11).

[3] This query is performed by using any of the algorithms proposed in [7, 11].

One issue that needs further elaboration is the computation of the set $\mathcal{D}_i$ at a server S_i (line 8 of *DSR* algorithm). To support efficient processing, the data set P_i is indexed by a multidimensional index structure, such as an R-tree. Then, the sets $\mathcal{D}_i$ can be computed efficiently by applying a branch-and-bound algorithm on the R-tree similar to a constrained skyline query [9]. For each intermediate representative point $p_i \in \mathcal{K}'$, the constraint is defined by point p_i and the origin of the data space and entries of the R-tree that do not overlap with the constraint are discarded. The set $\mathcal{D}_i$ contains the union of the results for all intermediate representative points $p_i \in \mathcal{K}'$. We emphasize that the use of the R-tree is simply to increase the efficiency, and it is by no means a strict prerequisite of *DSR*. Other non-indexed techniques for computing the sets $\mathcal{D}_i$ can be used instead.

Finally, Lemma 2 ensures the correctness of *DSR* by providing guarantees that Algorithm 2 always returns representative skyline points that belong to the skyline set, i.e., the set of representative skyline points is valid.

Lemma 2. *(*Correctness*) Any point p of the representative skyline set $\mathcal{K}$ produced by* DSR *belongs to the skyline set, i.e., if $p \in \mathcal{K}$ then $p \in \mathcal{S}$.*

Proof. Let us assume that $p \in \mathcal{K}$ and $p \notin \mathcal{S}$. Then, there exists a point $p' \in \mathcal{S}$ such that p' dominates p. We also conclude that $p' \notin \bigcup(\mathcal{K}_i \bigcup \mathcal{D}_i)$ because otherwise $p \notin \mathcal{K}$. Let us denote as S_j the server that stores p' locally. We distinguish two cases: (a) $p \in \bigcup K_i$, then $p' \in \mathcal{D}_j$ which leads to a contradiction, or (b) $p \notin \bigcup K_i$, then $p \in \bigcup D_i$, which means that there exists a point $q \in \bigcup \mathcal{K}_i$ such that p dominates q. Due to the properties of dominations we conclude that p' dominates q, which in turn leads to $p' \in \mathcal{D}_j$ which is a contradiction.

4.3 Distributed error-based representative algorithm (DER)

As *DSA* and *DSR* transfer only a fraction of data points to the coordinator, the representation error of the produced skyline representative points may be higher than in the case of the centralized skyline representative algorithm applied on the union of the local data points ($P = \bigcup P_i$). Even though *DSA* manages to return the same representative skyline set as the centralized algorithm on P, when the error metric does not depend on dominated data points, this does not hold for all error metrics such as for example the dominance error metric. The main reason of the higher representation error is that *DSA* and *DSR* use only restricted knowledge about the underlying data due to its distribution. Thus, our premise is to additionally use the information about the error metric at each server locally (resulting from the local representative skyline query) in order to improve the representation quality of the skyline representative set $\mathcal{K}$.

In the following, we describe a generic algorithm that produces representative skyline points for any error metric by taking into account scores of the candidate representative points derived from the local query processing. Then, we demonstrate the applicability of our algorithm for both the dominance and the distance-based error metric.

Algorithmic description The distributed error-based skyline representative algorithm (*DER*) processes the representative skyline query similarly to *DSR*. It

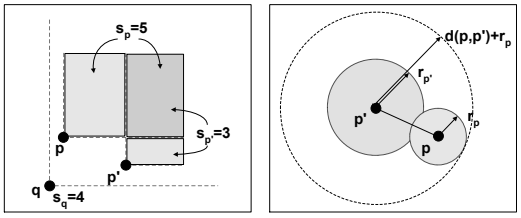
Algorithm 3 *Error-based Representative Selection*

1: **INPUT:** k, Local representative skyline $\bigcup \mathcal{K}_i$
2: **OUTPUT:** Representative skyline $\mathcal{K}$
3: $p \leftarrow argmax_{\forall p \in (\bigcup(\mathcal{K}_i))}(\underline{\text{score}}(p))$
4: $\mathcal{K} = \{p\}$
5: **while** $(|\mathcal{K}| < k)$ **do**
6: $\quad p \leftarrow argmax_{\forall p \in (\bigcup(\mathcal{K}_i) - \mathcal{K})}(\underline{\text{score}}(p, \mathcal{K}))$
7: $\quad \mathcal{K} = \mathcal{K} \bigcup \{p\}$
8: **end while**
9: **return** $\mathcal{K}$

consists of two phases that guarantee that the resulting set is valid. Moreover, *DER* produces candidate representative points by applying a skyline representative algorithm at local servers. The main difference to the previous algorithms is that each local representative skyline point $p \in \mathcal{K}_i$ is associated with a score of representativeness s_p. Then, the coordinator does not process a plain representative skyline query on $\bigcup \mathcal{K}_i$, but instead takes into account the score of representativeness of each point in order to minimize the error metric. In this way, an optimization problem is defined that aims to identify the k representative skyline points that minimize the error metric, given a set of candidate representative points $\bigcup \mathcal{K}_i$ each of them annotated by a score. As the representative skyline query has been shown to be NP-hard [7, 11], we propose a greedy algorithm to solve our optimization problem.

The *DER* algorithm assumes that each local representative point $p \in \mathcal{K}_i$ is augmented with a numeric value (score of representativeness) that indicates its goodness. Clearly, the definition of the score depends on the selection of the representative skyline algorithm, which is applied at the local servers. After the representative points $\mathcal{K}_i$ are collected at the coordinator, Algorithm 3 is assigned with the task of selecting k representative points, i.e., by solving the optimization problem. For this purpose, the algorithm uses the *score()* function that estimates the goodness of each candidate representative point. After selecting the first representative point (line 3), in each iteration the algorithm picks as a next representative point the one that maximizes the estimated score (line 6). After the selection of k representative points $\mathcal{K}'$, the points are sent to all servers for the verification step. Local skyline points $\mathcal{D}_i$ that dominate a point in $\mathcal{K}'$ are sent to the coordinator. Finally, the coordinator produces the final k representative skyline points, by solving again the same optimization problem over the union of points in sets $\mathcal{K}'$ and $\mathcal{D}_i$. Thus, Algorithm 3 is invoked taking as input $(\bigcup \mathcal{D}_i) \bigcup \mathcal{K}'$.

The *DER* algorithm is generic and allows any error metric, i.e., any centralized representative skyline algorithm, to be plugged in our framework. *DER* is parameterized by a function *score()* that computes the error metric. For any error metric that is plugged in (or equivalently for any centralized representative skyline algorithm that should be supported), we need to define an appropriate score of each representative skyline point and the implementation of the ab-



(a) Dominance. (b) Distance-based.

Fig. 4. Example of *score()* function.

stract function *score()*. In the sequel, we demonstrate how the dominance and the distance-based error metric are easily integrated and supported by *DER*.

Dominance error metric At a local level, the score s_p of a local representative skyline point $p \in P_i$ is defined as the number of points q that it dominates from the local data set P_i:

$$s_p = |\{q \in \mathcal{P}_i : p \prec q\}|$$

Notice that this definition makes the score of a representative skyline point dependent on the data points. Furthermore, we can accurately compute the aggregated score of a set of representative skyline points when all belong to different servers, as they dominate different data points.

As already mentioned, a set of representative skyline points $\bigcup \mathcal{K}_i$ is collected at the coordinator, each accompanied by its score. To use the *DER* algorithm to solve the optimization problem, we need to define the function *score()*. We use two versions of this function. The first, *score(p)*, computes the goodness for a representative skyline point p individually. This is useful in order to select the first representative skyline point. For this purpose, the most promising point is selected, therefore the function is defined as:

$$score(p) = s_p + \sum_{\forall q \in \bigcup \mathcal{K}_i : p \prec q} s_q$$

Intuitively, we select the point p that dominates points in $\mathcal{K}_i$ with maximum number of dominated points in total.

The second, *score(p, $\mathcal{K}$)*, computes the error when p is selected for inclusion in the set $\mathcal{K}$. As in each step we wish to add the next most promising point to the result set, we pick the point that maximizes the following function:

$$score(p, \mathcal{K}) = s_p + \sum_{\forall q \in \bigcup \mathcal{K}_i : p \prec q \wedge \nexists p' \in \mathcal{K} : p' \prec q} s_q$$

Intuitively, we compute as score an upper bound of the gain in the attained representation quality, when p is added to $\mathcal{K}$. This value is an upper bound because data points dominated by two representative skyline points are double-counted, since computing the distinctly dominated points is not feasible in practice. In the example of Fig. 4(a), only the scores s_p and $s_{p'}$ are known and not the exact values of the dominated points, thus the exact number of local data points dominated by q is not known. The $score(q)$ is estimated as $s_q + s'_p + s_p = 12$, which is an upper bound of the actual number of dominated points by q.

Distance-based error metric The score of a local representative skyline point $p \in \mathcal{K}_i$ is defined as the maximum distance of p to any non-representative skyline

point q for which there is no other representative p' closer to q than p. Formally, the score is defined as:

$$s_p = max_{\forall q \in S(\mathcal{P}_i)}\{d(p,q) : \nexists p' \in \mathcal{K}_i, d(q,p') < d(q,p)\}$$

To select the first representative skyline point, we follow the same strategy as iGreedy [11], and define the value of the first coordinate: $score(p) = -p[1]$. Finally, the score is defined as:

$$score(p,\mathcal{K}) = \left\{ \begin{matrix} 0,\ if\ \exists p' \in \mathcal{K} : d(p,p') + s_p < s_{p'} \\ min_{\forall p' \in \mathcal{K}}\{d(p,p') + s_p\},\ otherwise \end{matrix} \right\}$$

The function score s_p essentially defines a covering radius $r_p = s_p$ for a hypersphere centered at p, as depicted in Fig. 4(b). This hypersphere represents the region of the space with the following property: any non-representative skyline point in this region is closer to p than to any other representative skyline p'. The score function calculates the new radius of a hypersphere centered at p that covers all skyline points that are closer to p or p' than all other representative points. The estimation of the error is an upper bound of the actual error. In worst case, the distance between a candidate point p and its closer representative point p' is $d(p,p') + s_p$. If this is smaller than $s_{p'}$, then the representation quality does not decrease by not selecting the candidate as representative, thus the estimated error is set to zero.

The algorithm proceeds as above; it first picks one representative skyline point. Then, in each iteration, the error is estimated that will be introduced if the candidate representative skyline is not selected. The point with the highest error is selected as the next representative skyline, because otherwise the error metric will become equal to the highest error.

5 Experimental Evaluation

In this section, we provide an extensive study of our framework. We developed all algorithms (the baseline *DSA*, as well as our two proposed algorithms *DSR* and *DER*) in Java and simulated the distributed aspects of our framework. We implemented the distance-based representative algorithm (I-greedy algorithm [11]), as well as the greedy algorithm proposed in [7].

We employed synthetic data sets to examine different distributions, namely uniform (UN), clustered (CL) and anti-correlated (AC). For the clustered data set (CL), each server picks 10 cluster centroids randomly and the points follow a Gaussian distribution on each axis with variance 0.05, and a mean equal to the corresponding coordinate of the centroid. The anti-correlated (AC) data set was generated as described in [1]. For our experiments on synthetic data, we report the average results over 10 different instances of the data set. In addition, we employ another synthetic data set, called *Island* (IS), which is 2-dimensional and contains 63383 points. This data set is used in [11] to demonstrate the effectiveness of distance-based representative. We also use a real data set (NBA), which consists of 17265 5-dimensional tuples representing a player's performance per year. Since our setup is distributed, we distribute IS and NBA to the N servers by choosing a server per point uniformly at random. Again, we perform this process 10 times and report average values.

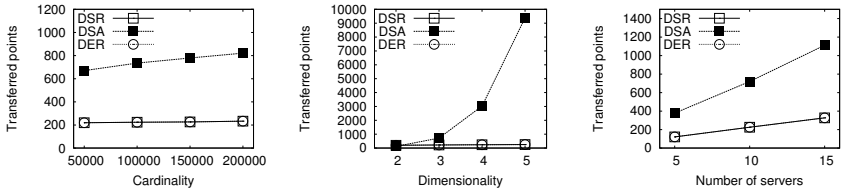
To evaluate the performance of our framework, we vary N from 5 to 15, d from 2 to 5, the cardinality n from 250K to 3M (which is evenly distributed to the N servers in advance), k from 10 to 50, the network speed from 1KB/sec to 100KB/sec, and we test different data distributions (UN,AC,CL,IS,NBA). We observed that the use of larger data sets increases the total time due to increased processing time, while the networking time is not significantly affected, since the number of transferred data remains relatively stable. Unless explicitly mentioned, the default setup is N=10, d=3, $|n_i|$=100K, k=10, network speed 50KB/sec, and we employ the UN data set. We note that when $k < \mathcal{S}(P_i)$, then k is set to $\mathcal{S}(P_i)$. The experimental evaluation focuses on two axes; the performance of our approach and the achieved quality of results. Our main performance metrics include: (i) the amount of transferred data and (ii) the total time, which is the time until the final result is produced at S_C (including network transfer time).

To evaluate the quality of our algorithms, we employ the normalized error metric. In the case of distance-based representative, the normalized error metric is $Er(\mathcal{K})$/MAX_DIST, where MAX_DIST represents the maximum distance of the space. Assuming a d-dimensional set of points where the value of each dimension belongs to $[0, U]$, then MAX_DIST=$U\sqrt{d}$. In the case of dominance representative, the normalized error metric is $Er(\mathcal{K})/n$. In all cases the normalized error takes values that belong to the range $[0, 1]$.

5.1 Experiments with distance-based representative

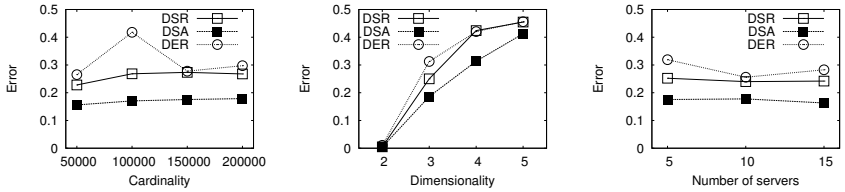
Evaluation for UN. In Fig. 5, we measure the amount of transferred data for various setups. In Fig. 5(a), we study the effect of increasing the cardinality of the data set from 50K to 200K points per server. *DSA* needs to transfer all local skyline points, thus resulting in much more traffic than *DSR* or *DER*. In Fig. 5(b), the number of transferred data points increases rapidly for *DSA*, due to the increase of each server's skyline cardinality as the dimensionality grows. Instead, *DSR* and *DER* show a much more stable performance, demonstrating the merits of the approaches that transfer only representative skyline points, rather than local skyline sets. In Fig. 5(c), we gradually increase the number of servers N in the system. The traffic induced by *DSA* (Fig. 5(c)) increases linearly with the number of servers. In contrast, *DSR* and *DER* scale gracefully.

Then, Fig. 6 shows the normalized error metric for different setups. Recall that in the case of distance-based representative, the error of *DSA* is equal to the error of the centralized distance-based representative algorithm and is greater than 0, unless all skyline points are reported as representative skyline points. Fig. 6(a) shows that the savings in network communication (depicted in Fig. 5(a)) cause *DSR* and *DER* to have higher error than *DSA*. It is also noteworthy that the increased cardinality does not affect the performance of the algorithms significantly. The reason is that the important factor is the skyline cardinality and not the data cardinality. In Fig. 6(b), the induced normalized error is reported for all algorithms, which increases with dimensionality. Notice that the difference between the algorithms remains practically the same. In Fig. 6(c), the error remains relatively stable for all algorithms regardless of N.



(a) Cardinality n_i. (b) Dimensionality d. (c) Number of servers N.

Fig. 5. Transferred data vs. cardinality, dimensionality and number of servers (UN).



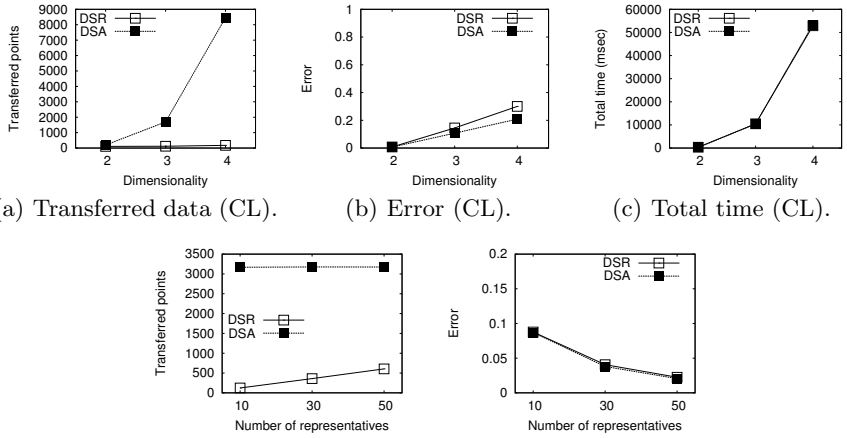
(a) Cardinality n_i. (b) Dimensionality d. (c) Number of servers N.

Fig. 6. Error vs. cardinality, dimensionality and number of servers (UN).

These experimental results indicate that *DER* does not improve the performance of *DSR*. This behavior is expected because the induced error of the distance-based representative skyline algorithm depends only on the skyline points. Consequently, *DSR* achieves results of high quality even with limited knowledge. Therefore, in the remaining experimental study of the distance-based representative, we omit *DER* from the charts.

Evaluation for CL. In Fig. 7(a), the amount of transferred data is depicted for our algorithms for CL data. We emphasize that each server picks cluster centroids randomly, therefore different servers have different clusters of data. Notice that *DSR* is practically unaffected by the increased dimensionality, thus demonstrating its merits when the data set is clustered. In contrast, the traffic induced by *DSA* increases with dimensionality. Then, in Fig. 7(b), we depict the normalized error metric. As in the case of UN, *DSR* exhibits higher error values than *DSA*, however here the difference is smaller than for UN. Also, the absolute error values are smaller than in the case of UN.

In addition, we measure the total time in Fig.7(c), which increases with dimensionality for both algorithms. This is expected, as the performance of any skyline or representative skyline algorithm deteriorates with increased dimensionality. Both algorithms have similar performance in terms of total time. In our experiments, we noticed that the processing time dominates the total execution time. This happens because the time required for transferring data is quite low, due to the assumed network speed of 50KB/sec. For more network-constrained networks, the transfer time is significant and affects the total time.



(a) Transferred data (CL). (b) Error (CL). (c) Total time (CL).

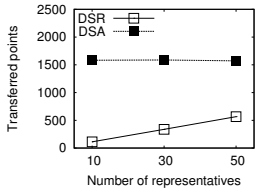
(d) Transferred data (AC). (e) Error (AC).

Fig. 7. Experimental results for CL and AC data sets.

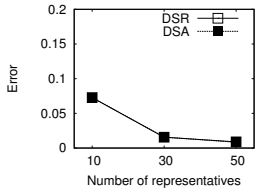
Evaluation for AC. In Figs. 7(d) and 7(e), we evaluate both algorithms for the 2-dimensional anti-correlated data distribution. We note that this distribution is the most challenging for skyline computation, since it results in high skyline cardinality, even for small dimensionality values. The aim of this experiment is to explore the behavior of our algorithms, when the local skyline sets at servers S_i are of high cardinality. First, in Fig. 7(d), *DSA* is unaffected by k, as it always transfers all local skyline sets regardless of k. Obviously, *DSR* needs to transfer more data as k increases. The important finding is that *DSR* requires to transfer one order of magnitude fewer data, thus demonstrating its appropriateness when the local skyline size is significant and network resources are limited. In Fig. 7(e), the normalized error is depicted. Notice that *DSR* shows marginally equal error with *DSA*, which is another strong argument in favor of *DSR*. Both algorithms exhibit a decreasing tendency with increased values of k. This is expected because as more representative skyline points are reported, the representative set describes more closely the real skyline set, thereby decreasing the error.

Evaluation for IS. For the IS data set, *DSR* is again much more communication-efficient than *DSA* as shown in Fig. 8(a), especially when the requested value k is small. Fig. 8(b) shows that the error is practically the same for both algorithms and drops for increased values of k. In Fig. 8, we also depict the data set and the representative skyline points discovered by the two algorithms. The two plots share 8 common points out of the total 10. The error is identical for both algorithms. When compared to the plot of Fig. 8(c), it is clear that both algorithms produce representative points that capture the shape of the skyline.

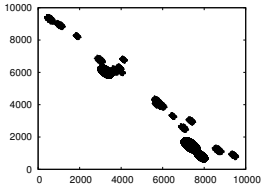
Evaluation for NBA. Then, in Fig. 9, we see that the conclusions drawn from the synthetic data sets are validated also in the case of the real data set. *DSR* always incurs significantly less network traffic (Fig. 9(a)), while the induced error is practically the same for both algorithms (Fig. 9(b)).



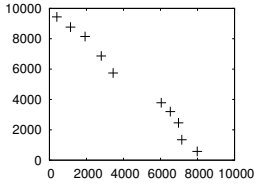
(a) Transferred data vs. k.



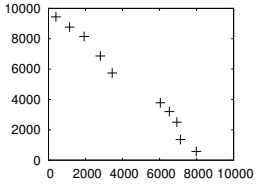
(b) Error vs. k.



(c) The *Island* data set.

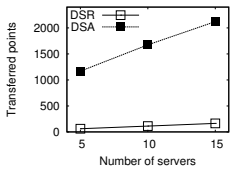


(d) *DSA* representatives.

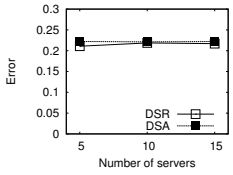


(e) *DSR* representatives.

Fig. 8. Experimental results for IS data set.



(a) Transferred data



(b) Error vs. N.

Fig. 9. Experimental results for NBA data set.

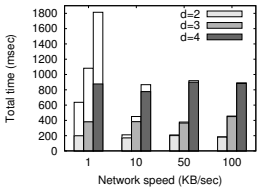
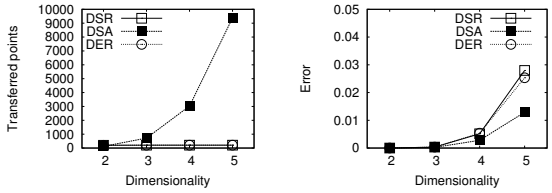


Fig. 10. Varying network speed.

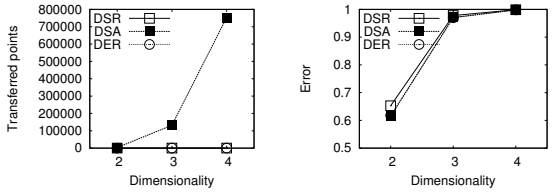
Varying network speed. In Fig. 10, we vary the network speed for the *DSR* algorithm. The total length of each bar corresponds to the total time, while the colored part corresponds to processing time. Larger values of network speed ($\geq$50KB/sec) do not affect performance, because the total time is dominated by the processing time, while the network transfer time is very small. In the case of smaller values of bandwidth (1KB/sec), we see that network transfer time increases and affects the total time.

5.2 Experiments with dominance representative

Evaluation for UN. Figs. 11(a) and 11(b) show the results of dominance representative for varying dimensionality. Both *DSR* and *DER* transfer significantly fewer data points, and the gain increases with d. An important finding is that *DER* improves the performance of *DSR* in terms of the error metric (Fig. 11(b)).
Evaluation for AC. In the next experiment, we test the performance of all algorithms in a hard setup (AC data distribution). As expected, when the dimensionality increases, the size of the local skyline sets increases rapidly, and

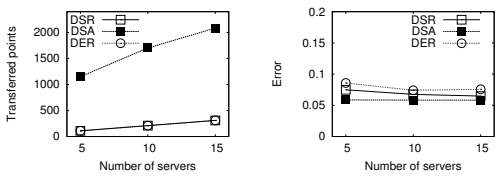


(a) Transferred data (UN). (b) Error (UN).



(c) Transferred data vs. d. (d) Error vs. d.

Fig. 11. Experimental results for UN and AC data sets.



(a) Transferred data. (b) Error vs. N.

Fig. 12. Experimental results for NBA data set.

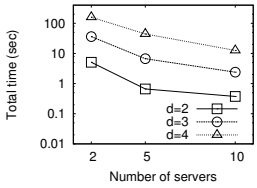


Fig. 13. Speed-up for *DSR* algorithm.

DSA needs to transfer too many data points, thus becoming impractical. In contrast, both *DSR* and *DER* scale gracefully in terms of transferred data. When the error is considered (Fig. 11(d)), all algorithms induce significant error values, but *DER* is better than *DSR*. *DSA* exhibits lower error values because it transfers a significant part of the local data sets to the coordinator, thus easing the task of selecting representative points.

Evaluation for NBA. Then, in Fig. 12, we test the performance of all algorithms for the NBA data set. We vary the number of servers, in order to study their behavior for increased network sizes. Fig. 12(a) shows that the increase in the number transferred points as the number of servers grows is smaller for *DSR* and *DER* than *DSA*. Fig. 12(b) depicts the induced error as N increases. The error of all algorithms remains practically unaffected, which shows that our framework is not significantly affected when more servers are employed.

Speed-up. Finally, in Fig. 13, we perform an experiment using a data set of 1M data points and distribute it to 2, 5, and 10 servers respectively. We test the *DSR* algorithm for the dominance representative. Clearly, when a higher number of

servers is used, the data set is distributed to smaller fragments, thus each server processes a smaller amount of data. In consequence, the processing cost of local computation on each server is reduced. This demonstrates that in the case of *DSR* runtime can be reduced by employing more servers.

6 Conclusions

In this paper, we addressed the challenging problem of discovering representative skyline points over distributed data, which naturally arises in various application domains, and it is mainly motivated by the unrestricted size of skyline cardinality. To address the problem effectively, we introduce a novel framework for processing the distributed skyline representative query. Our framework supports all metrics proposed for representative skyline queries in centralized settings.

References

1. Börzsönyi, S., Kossmann, D., Stocker, K.: The skyline operator. In: Proc. of ICDE (2001)
2. Chan, C.Y., Jagadish, H.V., Tan, K.L., Tung, A.K.H., Zhang, Z.: Finding k-dominant skylines in high dimensional space. In: Proc. of SIGMOD (2006)
3. Chaudhuri, S., Gravano, L.: Evaluating top-k selection queries. In: Proc. of VLDB (1999)
4. Cui, B., Lu, H., Xu, Q., Chen, L., Dai, Y., Zhou, Y.: Parallel distributed processing of constrained skyline queries by filtering. In: Proc. of ICDE (2008)
5. Fagin, R., Lotem, A., Naor, M.: Optimal aggregation algorithms for middleware. In: Proc. of PODS (2001)
6. Hose, K., Vlachou, A.: A survey of skyline processing in highly distributed environments. VLDBJ, to appear (2011)
7. Lin, X., Yuan, Y., Zhang, Q., Zhang, Y.: Selecting stars: the k most representative skyline operator. In: Proc. of ICDE (2007)
8. Lu, H., Jensen, C.S., Zhang, Z.: Flexible and efficient resolution of skyline query size constraints. IEEE TKDE 23(7), 991–1005 (2011)
9. Papadias, D., Tao, Y., Fu, G., Seeger, B.: Progressive skyline computation in database systems. ACM TODS 30(1), 41–82 (2005)
10. Sarma, A.D., Lall, A., Nanongkai, D., Lipton, R.J., Xu, J.J.: Representative skylines using threshold-based preference distributions. In: Proc. of ICDE (2011)
11. Tao, Y., Ding, L., Lin, X., Pei, J.: Distance-based representative skyline. In: Proc. of ICDE (2009)
12. Vlachou, A., Doulkeridis, C., Kotidis, Y., Vazirgiannis, M.: SKYPEER: Efficient subspace skyline computation over distributed data. In: Proc. of ICDE (2007)
13. Vlachou, A., Doulkeridis, C., Kotidis, Y., Vazirgiannis, M.: Efficient routing of subspace skyline queries over highly distributed data. IEEE TKDE 22(12), 1694–1708 (2010)
14. Zhu, L., Tao, Y., Zhou, S.: Distributed skyline retrieval with low bandwidth consumption. IEEE TKDE 21(3), 384–400 (2009)