Bandwidth-Constrained Distributed Skyline Computation

Akrivi Vlachou*
Department of Computer Science
NTNU
Trondheim, Norway
vlachou@idi.ntnu.no

Kjetil Nørvåg
Department of Computer Science
NTNU
Trondheim, Norway
Kjetil.Norvag@idi.ntnu.no

ABSTRACT

Skyline queries have been studied in centralized systems, but also in distributed environments, such as web information systems and peer-to-peer networks. Most of the existing work focuses on efficient processing of skyline queries that return the exact and complete result set. In this paper, we study skyline computation in a distributed environment under the assumption of a given upper bound on the bandwidth consumption. Supporting such a constraint is very important in a mobile environment, where data transmissions may lead to deterioration of query processing performance, while imposing high cost for the mobile device in terms of energy consumption. Therefore, the cost of transferring all data points that may contribute to the skyline result set to the querying server is often prohibitive. Our target is, given an upper bound of bandwidth consumption, to maximize the quality of the retrieved skyline approximation. For this purpose, we propose a framework for bandwidth-constrained skyline query processing, based on the intentional selection of a limited amount of data points from each participating mobile device. We propose a novel method to carefully select the most promising data points based on subspace dominations and analyze its properties. In our extensive experimental evaluation, we demonstrate that result sets of high quality are achieved, while reducing the communication cost considerably.

1. INTRODUCTION

The recent advances in wireless communications have enabled the widespread usage of mobile devices. Moreover, the capabilities of mobile devices – such as PDAs, cellular phones, netbooks – in terms of storage capacity, processing power and network connectivity have increased significantly. Thus, mobile devices store data locally and provide access to their data by means of wireless connectivity. For example, consider a tourist interested in highly ranked and cheap restaurants that are near by its location. Other mobile devices that are in the same region are likely to store information and personal opinions about local restaurants. Thus, a tourist could benefit from this information. In our scenario, the communication between mobile devices is facilitated through a base station. For the deployment of several useful applications, it is important to design distributed algorithms for efficient query processing over data stored by mobile devices.

Skyline queries help users make intelligent decisions over complex data, where different and often conflicting criteria are considered. Such queries return a set of interesting data points – from the user's perspective – that are not dominated by any other point on all dimensions [2]. Skyline queries have been studied in centralized systems [2], but also in distributed environments such as web information systems [1, 12], peer-to-peer networks [5, 7, 9, 17, 18, 19] and mobile ad-hoc networks [10]. Although distributed skyline query processing has been studied in a variety of domains, the focus has been on providing the exact and complete result set. However, our primary goal is to maximize the quality of the retrieved result set when given an upper bound of bandwidth consumption. The reason is two-fold: first, to reduce the latency of the system, which is associated with increased data transfers, and second, to restrict the energy consumption of the mobile device, which is affected by high rates of data transmissions.

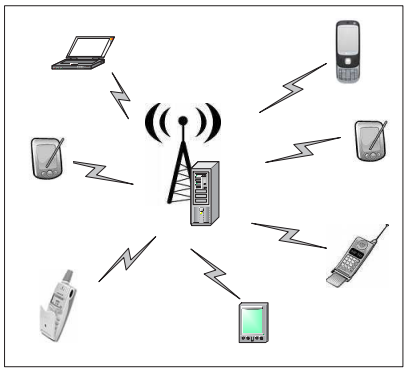


Figure 1: System architecture

In this paper, we study the efficient processing of bandwidth-constrained skyline queries. We assume a system architecture that consists of a server equipped with a wireless network access point, and several mobile devices in the vicinity of the server at a given time, as depicted in Figure 1. The

*This work was carried out during the tenure of an ERCIM "Alain Bensoussan" Fellowship Programme.

MobiDE'09, June 29, 2009, Providence, RI, USA.

server can directly communicate with any device and request data. In such a setting, each time the server sends a skyline request to a mobile device, all points that are not dominated locally (local skyline points) have to be transferred to the server, in order to ensure the exactness of the result set. Nevertheless, many points are actually discarded by the server, since they are dominated by points stored at other mobile devices. In our approach, each mobile device transfers only a fraction of the local skyline points, namely those that have the highest probability to belong to the skyline set. Quantifying the probability of a point to belong to the skyline result set poses inherent challenges due to the distribution of content and the lack of global knowledge.

In this context, we propose a framework for bandwidth-constrained skyline query processing, based on the intentional selection of the most promising local skyline points from each participating mobile device. We study the properties that local skyline points should have, in order to increase the probability of belonging to the global skyline. Towards this goal, we propose a novel method to carefully select local skyline points based on subspace dominations and compare it with two methods previously proposed in related work for ranking of skyline points. The main contributions of our work are:

- We propose a framework for processing bandwidth-constrained skyline queries over mobile devices, aiming at computing an approximate skyline result set that contains as many skyline points as possible.
- We analyze the main properties of skyline points that qualify them as skyline points.
- We present a novel method based on subspace dominations for selecting of a limited amount of local skyline points. We also identify two existing skyline ranking methods that fit in our framework and study their comparative performance.
- We conduct a thorough experimental evaluation that demonstrates the efficiency of our approach, in terms of high quality of retrieved approximations, with important savings in bandwidth consumption. Our algorithm consistently outperforms its competitors in all examined setups.

The remaining of this paper is organized as follows: Section 2 reviews the related work. Then, in Section 3, we provide the necessary preliminaries and definitions. In Section 4, we present a system overview. In Section 5, we present a framework for bandwidth-constrained skyline query processing and propose a novel method that exploits subspace dominations to identify the most promising local points to belong to the skyline. The experimental evaluation is presented in Section 6. Finally, we conclude the paper in Section 7.

2. RELATED WORK

Skyline computation has received considerable attention in the database research community. Börzsönyi et al. [2] first investigated the skyline computation problem in the context of databases. Thereafter, several techniques have been proposed for efficient skyline computation. For example, in [13], a branch and bound algorithm (BBS) on a dataset indexed by an R-Tree, with guaranteed minimum I/O cost, was proposed. Subspace skyline retrieval was studied in [15], and the SUBSKY algorithm was presented. Pei *et al.* [14] discussed subspace skylines primarily from the view of query semantics. The authors in [21] present a pre-processing approach, called SKYCUBE, which is defined as the union of all skyline points of all possible non-empty subspaces. Chan *et al.* [3] propose the k-dominant skyline query to restrict the skyline cardinality. The authors relax the idea of dominance to k-dominance, in order to increase the probability of one point dominating another point. In [11], the authors study the problem of selecting k skyline points so that the number of points, which are dominated by at least one of these k skyline points is maximized. In [4], the authors introduce a new metric called skyline frequency, to compare and rank the interestingness of data points based on how often they are returned in the skyline, when different subspaces are considered.

There has been a growing interest in distributed [1, 5, 6, 7, 9, 12, 17, 18, 19, 22] and parallel [16, 20] skyline computation lately. In [1, 12], skyline processing is studied over distributed web sources. In both cases, the authors assume vertical partitioning of the dataset across a set of participating web accessible sources. This is entirely different from our setup, where the aim is to process skyline queries over a horizontally distributed dataset to mobile devices.

Huang *et al.* [10] assume a setting with mobile devices communicating via ad-hoc networks (MANETs) and study skyline queries that involve spatial constraints. The authors present techniques that aim to reduce both the communication cost and the execution time on each single device. The communication cost is reduced by sending a filter point to the devices, that discards local skyline points. This differs to our goals, since [10] does not focus on bounding the consumed bandwidth. Furthermore, the use of filter points is orthogonal to our approach and can be used additionally.

In [9], the authors focus on skyline computation in Peer Data Management Systems (PDMS), where each peer provides its own data with its own schema. Wang *et al.* [18] use the z-curve method to map the multidimensional data space to one-dimensional values, that can then be assigned to peers connected in a tree overlay. Later, this work has been extended resulting in SkyFrame [19]. Chen *et al.* [5] propose the iSky algorithm, which employs a different one-dimensional transformation, in order to assign data to peers in a tree overlay. SKYPEER [17] is a framework for subspace skyline query processing over super-peer architectures. Similarly, Fotiadou *et al.* [7] propose BITPEER that uses a bitmap representation in order to improve the performance of query processing. These approaches focus mainly on the efficient propagation of the skyline queries in the overlay network. In contrast, we do not assume the existence of an overlay network, and we primarily focus on restricting the transferred data points.

In [20], Wu *et al.* address the problem of parallelizing skyline queries over a shared-nothing architecture using space partitioning. Following a different approach, in [16], the authors present a novel angle-based space partitioning that outperforms the traditional grid-based partitioning for parallel skyline computation. In contrast to our work, which focuses on bounding bandwidth consumption, the focus of parallel skyline computation is mainly to minimize the response time and share the processing workload to all servers.

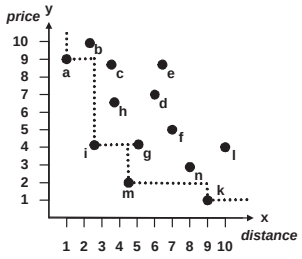


Figure 2: Skyline example

Recently, approaches that assume no overlay network have been proposed [6, 22]. Cui *et al.* [6] study skyline query processing in a distributed environment, where a coordinator can directly communicate with all servers. The authors propose the use of MBRs to summarize the data stored at each server. In [22], a feedback-based distributed skyline (FDS) algorithm is proposed, which also assumes no particular overlay network. The algorithm needs several roundtrips to compute the skyline result. In both approaches, similarly to our architecture, a server directly communicates with other servers, but their aim is to compute the exact skyline, thereby consuming excessive bandwidth in many occasions.

3. PRELIMINARIES

Given a data space D defined by a set of d dimensions $\{d_1, ..., d_d\}$ and a dataset S on D with cardinality n, a point $p \in S$ can be represented as $p = \{p_1, ..., p_d\}$ where p_i is a value on dimension d_i. Without loss of generality, let us assume that the value p_i in any dimension d_i is greater or equal to zero ($p_i \geq 0$) and that for all dimensions the minimum values are more preferable. A summary of the symbols used in this paper is given in Table 1.

DEFINITION 1. ***Skyline:*** *A point $p \in S$ is said to* dominate *another point $q \in S$, denoted as $p \prec q$, if (1) on every dimension $d_i \in D$, $p_i \leq q_i$; and (2) on at least one dimension $d_j \in D$, $p_j < q_j$. The* skyline *is a set of points $SKY \subseteq S$ which are not dominated by any other point. The points in SKY are called skyline points.*

Let us assume for example a database containing information about hotels. Each tuple of the database is represented as a point in a data space consisting of numerous dimensions. In our example, the y-dimension represents the price of a room, whereas the x-dimension captures the distance of the hotel to a point of interest such as the beach (Figure 2). According to the dominance definition, 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 tradeoffs between price and distance from the beach.

Skyline analysis applications often provide numerous candidate attributes resulting in high dimensional data spaces. In our running example, the hotel database could contain various other attributes, such as the number of rooms, the size of room and the star rating. The notion of skyline can be extended to subspaces, where given a set of d-dimensional points, a subspace skyline query only refers to a user-defined subset of attributes. Each non-empty subset U of D ($U \subseteq D$) is referred to as a *subspace* of D. The data space D is also referred as full space of the dataset S.

Symbols	Description
S	Dataset
d	Data dimensionality
n	Cardinality of the dataset
$p = \{p_1, ..., p_d\}$	Data point
$D = \{d_1, ..., d_d\}$	Data space
$U \subseteq D$	Subspace of D
SKY	Skyline set
SKY_U	Skyline set on subspace U
N	Number of mobile devices
M_i	Mobile device($i = 1..N$)
S_i	Partition of dataset on M_i
SKY^i	Skyline set of M_i

Table 1: Overview of symbols

DEFINITION 2. ***Subspace Skyline:*** *A point $p \in S$ is said to* dominate *another point $q \in S$ on subspace $U \subseteq D$, denoted as $p \prec_U q$, if (1) on every dimension $d_i \in U$, $p_i \leq q_i$; and (2) on at least one dimension $d_j \in U$, $p_j < q_j$. The* skyline *of subspace U is a set of points $SKY_U \subseteq S$ which are not dominated by any other point on subspace U. The points in SKY_U are called* skyline points *on subspace U.*

Consider for example the dataset depicted in Figure 2. The skyline points are $SKY = \{a, i, m, k\}$, while for the subspace $U = \{x\}$ the subspace skyline set is $SKY_U = \{a\}$.

4. SYSTEM OVERVIEW

The system architecture consists of one central server, also called coordinator, and a set of mobile devices M_i ($1 \leq i \leq N$), which exist within the coordinator's area of coverage, at a given time. The coordinator can directly communicate with any device M_i. A mobile device M_i has enhanced capabilities in terms of local storage capacity and processing power. Each device stores a portion S_i of the complete dataset S, thus horizontal partitioning of data to devices is assumed $S = \cup_{1 \leq i \leq N} S_i$ and $S_i \subseteq S$. Moreover, a device is capable of processing the skyline of its local data. Given a skyline query, the coordinator is responsible for computing the skyline set of the entire dataset S.

OBSERVATION 1. *A point $p \in S$ cannot be a skyline point, unless there exist a partition $S_i \subseteq S$ ($1 \leq i \leq N$) with $p \in S_i$ and $p \in SKY^i$, where SKY^i denotes the skyline set of the data points in S_i.*

In other words, the skyline points over a horizontally partitioned dataset are a subset of the union of the skyline points of all partitions. For sake of simplicity, we assume that each mobile device M_i stores locally the skyline set SKY^i based on the local data partition S_i (mentioned also as local skyline points), and we do not take into consideration the local processing cost. This is because we assume that the dataset stored at each mobile device changes infrequently, while the most dynamic part of our system is caused by the mobility of devices that enter or leave the region of responsibility of the coordinator. The coordinator gathers the local skyline points SKY^i of all mobile devices. Then, the local skylines are merged to the overall skyline set, by computing the skyline of the local skyline sets. Thus, local skyline points that are dominated by points stored by another mobile device are discarded. The above observation guarantees that the distributed skyline computation returns the exact skyline set, after merging the local

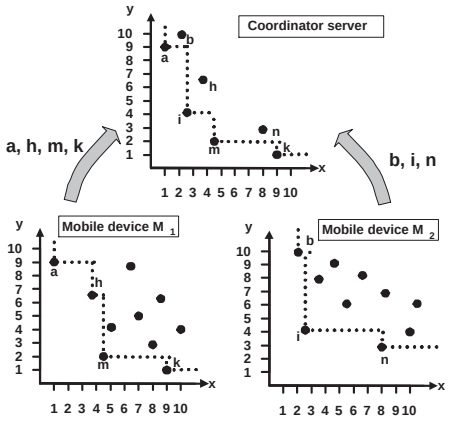


Figure 3: Distributed query processing

skyline results sets. For example, consider Figure 3 where two mobile devices participate in the skyline computation. Each mobile device computes its local skyline set, namely the sets $SKY^1 = \{a, h, m, k\}$ and $SKY^2 = \{b, i, n\}$ respectively. Then, these points are transmitted to the coordinator server. During the merging phase, points b, h and n are discarded, since they are not part of the skyline result set.

It is obvious that given a set of N mobile devices, the overall performance of skyline query processing mostly depends on the number of local skylines that are transferred to the coordinator. In a mobile environment, the amount of transferred data can be excessive, in terms of both bandwidth and energy consumption caused by the data transmission. Therefore, in this paper, we focus on bandwidth-constrained skyline queries, where an upper limit for the amount of transmitted data is given. The aim is to retrieve as many of the skyline points of dataset S as possible, by transferring only a limited number of local skyline points from each device.

5. BANDWIDTH-CONSTRAINED SKYLINE QUERIES

Given an upper bound on the bandwidth consumption, we present a framework for supporting bandwidth-constrained skyline queries. We assume that it is not possible to transmit all the local skyline points to the coordinator. Thus, the coordinator requests only a fraction of the local skyline points from each mobile device. The number k of requested local skyline points is straightforwardly derived from the given upper bound (B) on the bandwidth consumption, the number of participating mobile devices (N) and the size of each data point ($|p|$), as: $k = \lfloor \frac{B}{N * |p|} \rfloor$.

5.1 Motivation and Aims

The cardinality of the skyline set SKY depends on the data distribution and increases rapidly as the dimensionality increases. The high cardinality of the skyline operator, leads to a high number of local skyline points, which in turn causes high amount of transferred data during skyline query processing. During the merging phase many local skyline points are discarded, since they are dominated by skyline points of other mobile devices. For example, in Figure 3, points b, h and n are not part of the skyline result set, and these points could have been avoided to be transferred to the coordinator.

In a mobile environment, where the dataset is horizontally distributed, the local skyline points are many more than the skyline points. For example, if every mobile device holds approximately an equal number of random points, then each peer is expected to have approximately the same number of skyline points $s = |SKY^i|$. Therefore, the total number of local skyline points is equal to $N * s$. It has been shown [8] that the expected number of skyline points for a random dataset depends mainly on the dimensionality of the dataset, while the cardinality of the dataset influences only slightly the cardinality of the skyline set. Thus, the number of skyline points is expected to be around s, much smaller than the total number of local skyline points, i.e., $|SKY| << \sum_{i=1}^{N} |SKY^i|$. Therefore, only a fraction of the local skyline points is necessary to compute the skyline result set, and the aim is to discover the local skyline points that are not dominated by points stored on other mobile devices.

Based on a given upper bound of bandwidth consumption, each mobile device selects only k local skyline points to send to the coordinator. This approach reduces the amount of transferred data, however it also leads to approximate skyline results. Since some local skyline points are not returned to the coordinator, we can not guarantee that the coordinator receives all the points that are necessary to compute the skyline set. The quality of the approximate skyline set, i.e., the number of retrieved skyline points, depends on which local skyline points are selected to be transmitted to the coordinator. Given a scoring function that ranks higher local skyline points that have a high probability of belonging to the skyline set, only a small fraction of the local skyline points are necessary, to compute the complete skyline set. Therefore, in the following, we investigate scoring functions that rank highly skyline points that have a high probability of dominating local skyline points of other partitions and a small probability of being dominated by other skyline points.

5.2 Methods for Selecting Skyline Points

In [13] the importance of skyline points is defined based on the number of dominated points. [13] discusses enumerating queries that return for each skyline point p, the number of points dominated by p. Similarly, in [11], the number of points that a skyline point dominates is considered as a measure of importance of the skyline points. In the running example of Figure 2, hotel m dominates 6 points, whereas hotel a dominates only one point. Under the assumption that all data partitions follow the same data distribution, a local skyline point p that dominates more data points in a partition S_i, probably dominates also many data points in all other partitions. In this spirit, we define a scoring function based on the number of dominated points.

DEFINITION 3. ***Number of dominated points:*** *The score of a skyline point p is defined as the number points q that are dominated by p, i.e.,* $score_{DOM}(p) = |\{q \in S : p \prec q\}|$.

The number of dominated points is equivalent to the pruning power of a skyline point, and its importance has been recognized by many approaches, for example for selecting filter points. On the other hand, Pei et al. [14] aim to answer the question why a point belongs to the skyline set, mentioned as skyline membership query, based on which subspace skyline sets a point belongs to.

OBSERVATION 2. *A skyline point $p \in SKY_U$ on a subspace $U \subseteq D$ is either a skyline point on D, or is dominated on D by another skyline point $q \in SKY_U$ for which $p_i = q_i$, $\forall i : d_i \in U$.*

Under the assumption that all the data points have distinct values on each dimension, this means that a subspace skyline point is definitely a skyline point. In our running example (Figure 2) skyline point a is a subspace skyline point on $U = \{distance\}$, which indicates that point a is a skyline point because it has the best value on the distance, while the price of the hotel is not important. On the other hand, point i is a skyline point only when both distance and price are considered, thus point i qualifies as a skyline point, since the price combined with the distance are better than any other point. Following this concept, in [4] the skyline frequency, i.e., how often a point is returned in the skyline when different subspaces are considered, was proposed as a scoring function for skyline points.

DEFINITION 4. ***Skyline frequency:*** *The skyline frequency of a skyline point p is defined as the number of different subspaces for which p is in the subspace skyline result set, i.e., $score_{FREQ}(p) = |\{U \subseteq D : p \in SKY_U\}|$.*

Intuitively, as stated in [4], a skyline point p with high skyline frequency has a higher probability of dominating other points, since p dominates points in many subspaces. Let us for example assume that our hotel dataset has 4-dimensions in total: distance, price, size of room and the star rating. Let us further assume that point i is skyline point not only in the subspace $\{distance, price\}$, but also in the subspace $\{star\ rating\}$. Under the assumption of distinct values, this means that i is also a skyline points for all subspaces U, for which $\{star\ rating\} \subseteq U$ or $\{distance, price\} \subseteq U$. Therefore, skyline point i is also a skyline point in the 4 dimensional data space. Even if skyline point i is dominated by a point stored by another device in subspace $\{star\ rating\}$, point i may still qualify as a skyline point based on subspace $\{distance, price\}$.

The above scoring functions capture two parameters that influence whether a point is a skyline point. In the following, we combine the two parameters into a novel scoring function. We define the subspace dominance scoring function that returns a score for skyline point p based on the number of skyline points that p dominates in different subspaces.

DEFINITION 5. ***Number of subspace dominations:*** *The score of a skyline point p is defined as the number of skyline points q that are dominated by p, such as $score_{SDOM}(p) = |\{\exists q, V, U : p \in SKY_U, q \in SKY_V, p \prec_U q, U \subseteq V, |U - V| = 1\}|$, where V, U subspaces of D.*

To explain this better, let us consider the example depicted in Figure 2. Point a is a subspace skyline on subspace $U = \{distance\}$ and dominates the skyline points i, m, k on subspace U. Therefore $score_{SDOM}(a) = 3$, since point a does not belong to any other subspace skyline set. Now, let us assume that the dataset is 3-dimensional, namely $\{distance, price, star\ rating\}$ and that point h is a skyline in the 3-dimensional data space. Point i dominates h on the subspace $\{distance, price\}$, so the score of i increases by one. On the other hand, a does not dominate h on the subspace $\{distance, price\}$, even though a is a skyline point. Nevertheless, a dominates h in the subspace $\{distance\}$, but this does not increase the score of point a, since h is not a skyline point in subspace $\{distance, price\}$. This is because, each time we take into consideration dominations of skyline points, only if the condition $|U - V| = 1$ holds. The intuition is that if i did not exist, then h would be a subspace skyline in $\{distance, price\}$, and therefore also in the 3-dimensional space, even if a dominates h in $\{distance\}$. Notice that during the computation of the number of subspace dominations, only the skyline points are taken into account, in order to capture better the distribution of the skyline points that are sent to the server by other mobile devices.

To summarize, a skyline point p is highly ranked based on $score_{SDOM}(p)$, if it dominates many other skyline points in many subspaces. Therefore, skyline point p has a high probability to dominate skyline points of other mobile devices in many subspaces. Thus, skyline point p also has a higher probability to dominate points in the original data space (Observation 2).

Notice that other approaches have been also proposed for determining a set of k skyline points [3, 11], based on k-dominance or on representativeness. However, in our framework, we are interested in supporting arbitrary k values per query, as k is a parameter that is dynamically defined by the server, based on the available bandwidth and the number of mobile devices at a given time. Therefore, we do not evaluate approaches that require pre-specified values for k in our framework.

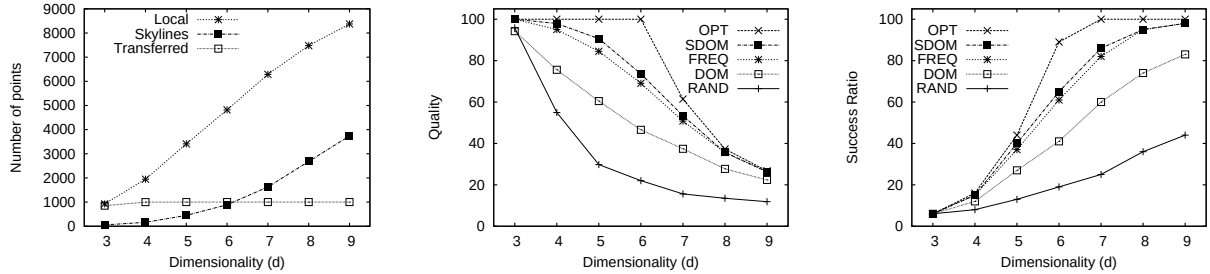
6. EXPERIMENTAL STUDY

In our experimental evaluation, we studied the performance of our framework using simulations. The simulator was implemented in Java and in order to test the algorithms with a realistic number of mobile devices, we simulated multiple instances of mobile devices on the same machine and simulated the network communication.

6.1 Experimental Setup

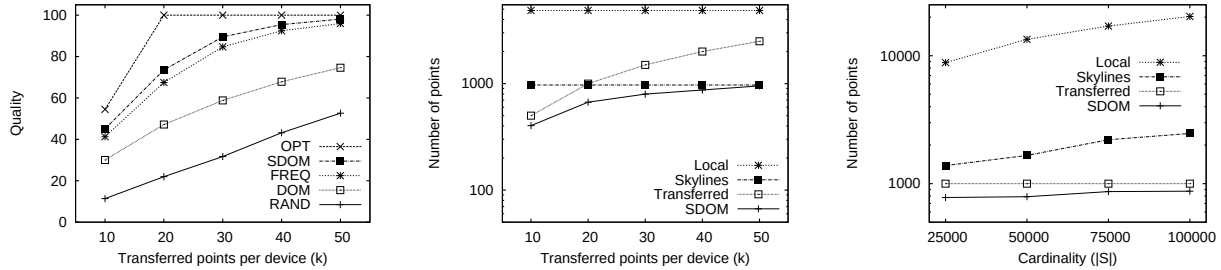
In the following, we study the performance of the approximate skyline set, under the assumption that only a fraction of the local skyline points of the mobile devices are transferred to the coordinator. We evaluate different scoring functions for selecting the points that are transferred, namely the skyline frequency (FREQ), dominance (DOM) and subspace dominance (SDOM) methods. Moreover, we compare against the case where local skyline points are selected at random (RAND) to be sent to the coordinator.

In order to assess the performance of the approximation, we compare the retrieved skyline set, using the aforementioned scoring functions, to the actual skyline computed on the total dataset S, as if S were available centrally at one server. We define as *quality* of the approximation the percentage of skyline points that are retrieved. Furthermore, we define as *success ratio* the percentage of transferred points that are skyline points, indicating how many points we could avoid transferring. In order to evaluate our approach, we also define an optimal (OPT) upper bound both for the quality and the success ratio. In the case of quality, the bound is computed as: $\frac{min(kN, |SKY|)}{|SKY|}$, whereas in the case of success ratio, the bound is: $\frac{min(kN, |SKY|)}{kN}$. The optimal



(a) Number of points (b) Quality (c) Success ratio

Figure 4: Comparative performance of scoring functions vs. dimensionality for uniform datasets



(a) Quality (b) Number of points for varying k (c) Number of points for varying $|S|$

Figure 5: Comparative performance of scoring functions vs. transferred data (k) and cardinality ($|S|$)

value represents the case where all transferred data points are skyline points under the condition that the skyline points are more than the transferred data points.

In order to evaluate the scalability of our framework we use synthetic data collections, namely uniform, anti-correlated and correlated datasets, horizontally partitioned evenly among the mobile devices. The synthetic datasets are generated as described in [2], for varying dimensionality (d=3 − 9) and cardinality (S=10K − 100K). Furthermore, we use different numbers of mobile devices (N=25 − 100) to test the scalability in terms of the size of the system. Additionally, we study how increasing values of transferred skyline points (k=10 − 50) improve the approximation. All experiments are repeated 20 times and the average values are depicted.

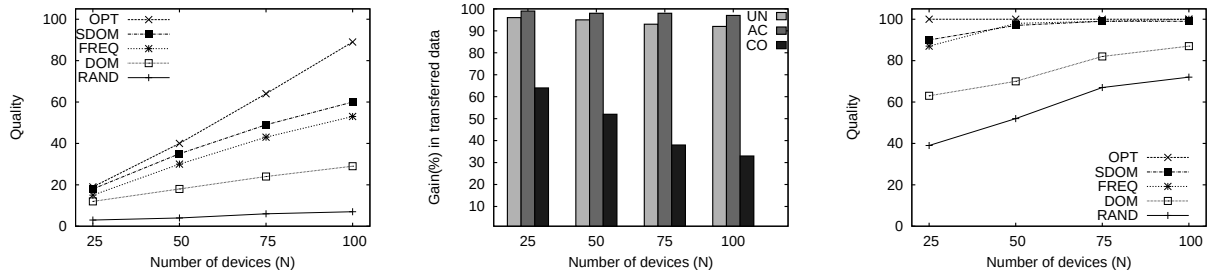
6.2 Experimental Results

In the first experiment, we use a set of N=50 mobile devices and each device stores $|S_i|$=200 points that follow a uniform data distribution. Thus, the cardinality of the dataset is $|S|$=10K points. Each device sends to the coordinator only its k=20 highest ranked skyline points and we evaluate the result using only this small subset of local skyline points, in comparison to the case where each mobile device sends all local skyline points. Figure 4(a) depicts the number of skyline points compared to the total number of local skyline points at all mobile devices, verifying that the local skyline points are many more that then actual skyline points. As the dimensionality increases, the number of skyline points increases rapidly, but the number of local skyline points increases even faster. The total number of transferred data points are 1000, except for the case of d=3 where some mobile devices have less than 20 local skyline points, resulting in less than 1000 transferred data points.

In Figure 4(b), we present the achieved quality of the approximation as we vary the dimensionality. The results show that the SDOM approach performs better, followed by FREQ. The decreasing tendency in all cases as the dimensionality increases is because the cardinality of the skyline set also increases. Thus, as each mobile device sends only k=20 points, after a certain dimensionality value these points are not sufficient to provide higher quality values. Consequently, the values of OPT decrease as well, which reflect the best quality that can be achieved by transferring k points per device. However, SDOM performs as good as OPT for low dimensionality values (d=2 − 3) and high dimensionality values (d=8 − 9). For the lower dimensionality, this is caused by the small cardinality of the skyline set ($|SKY|$), which also explains why the RAND approach performs sufficiently well. In contrast, for higher dimensionality values, the performance of RAND deteriorates, while SDOM performs effectively since it exploits the high number of subspaces ($2^d - 1$), in order to select and transfer points that will belong to the skyline of S with higher probability.

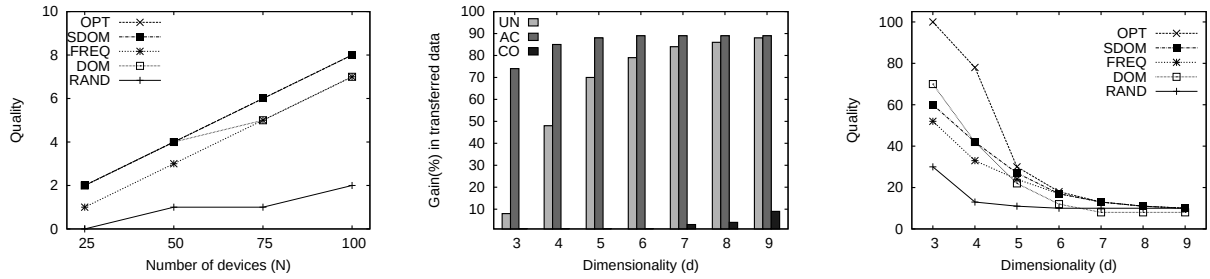
In Figure 4(c), we measure the success ratio for the same experimental setup. Again, the overall performance of the SDOM is the best, followed by FREQ. The low values of success ratio for small dimensionality values are due to the small cardinality of the skyline set. As the skyline cardinality increases, the success ratio also increases. Again, SDOM performs almost as good as OPT.

Then, for the same experimental setup, we vary the number of points (k=10 − 50) each mobile device sends to the coordinator, while the dimensionality of the dataset is set to d=6. In Figure 5(a), the values of quality are depicted for varying number of k. As expected, the quality of the approximation improves as the number of transferred data point increases. An interesting observation is that even by transferring only a small subset of local skyline points, SDOM manages to achieve results of high quality. Furthermore,



(a) Quality (b) Gain in transferred data (c) Correlated data

Figure 6: Comparative performance of scoring functions vs. number of mobile devices (N)



(a) Quality (b) Gain in transferred data (c) Quality

Figure 7: Comparative performance of scoring functions for anti-correlated data

for k=50, SDOM achieves almost the optimal value of quality, while RAND produces results of very low quality. Figure 5(b) depicts (in logarithmic scale) the number of skyline points and local skyline points that are obviously not influenced by the increasing values of k. In addition, the number of transferred data is shown, which increases as k increases. However, the transferred data points are much fewer than the local skyline points. Figure 5(b) also depicts the number of skyline points that SDOM retrieves, verifying that for k=50 almost all skyline points are retrieved.

In the next experiment, we vary the cardinality of the dataset ($|S|$=$25K - 100K$), thus each mobile device holds from $|S_i|$=500 to $|S_i|$=2000 points respectively. Again, each mobile device (N=50) sends only 20 local skyline points. The dimensionality of the dataset is set to d=6. Figure 5(c) verifies that the number of skyline points is not significantly influenced by the cardinality of the dataset. Notice that $SDOM$ succeeds in transferring points that are skyline points, independently of the cardinality of the dataset. Thus, the success ratio is also not influenced by the cardinality, whereas the quality of the result set decreases only slightly. These figures are omitted due to space limitations.

In Figure 6(a), we vary the number of mobile devices (N=$25 - 100$), while the cardinality and the dimensionality of the dataset are set to $|S|$=$100K$ and d=6 respectively. In Figure 6(a), the quality of the approximation is depicted. Again SDOM outperforms the other scoring functions. Notice that for all scoring functions the quality of the approximation increases with the number of mobile devices, since the total number of transferred data increases (k=20 per device), while the number of skyline points does not change. Nevertheless, in all cases the transferred data are less than 8% of the local skyline points (that also increase as the number of devices increases), which leads to more than 92% savings in bandwidth consumption.

Figure 6(b) depicts the savings (in %) in transferred data points against the naive approach (transferring all local skyline points) for varying number of participating mobile devices and for different data distributions, namely uniform (UN), anti-correlated (AC) and correlated (CO). Clearly, the savings are higher for uniform and anti-correlated data, since in the case of correlated data the local skyline points are fewer. Therefore, for correlated data, approximations of high quality values are retrieved, as also depicted in Figure 6(c). For the case of anti-correlated data the savings in transferred data are very high (Figure 6(b)), since the number of local skyline points is huge, i.e., up to 85% of the dataset. Notice that the cardinality of the skyline set is also high, around $|SKY|$=$24K$ data points. This leads to very low absolute values of quality, as depicted in Figure 7(a), even though SDOM and FREQ return as good approximations as OPT. Nevertheless, for anti-correlated data, a fraction of the result set is sufficient, since for example in the case of N=50 mobile devices, there exist $|SKY|$=24512 skyline points and SDOM manages to achieve 4% quality, which means that 996 out of 1000 transferred points are skyline points that is sufficient for the majority of applications. Also notice that the success ratio is near 100%.

In the next series of experiments, we study the scalability in terms for dimensionality of our approach for all data distributions. The number of mobile devices is set to N=50 and each of them holds $|S_i|$=200 points ($|S|$ = $10K$). Figure 7(b) depicts the savings in transferred data. We notice that for the correlated dataset the gain is only marginal, due to the small number of local skyline points, caused by the small number of points per mobile device. The savings for uniform and anti-correlated data are more significant. Figure 7(c) depicts the quality of the retrieved result set for anti-correlated data. It is obvious that the quality drops rapidly with increasing dimensionality, much faster

than the uniform case (Figure 4(b)). This is due to the high cardinality of the skyline result set, but most of the points that SDOM transfers to the coordinator are skyline points, therefore for higher dimensions SDOM achieves the optimal value. For example, for d=6 the success ratio of SDOM is 99%, whereas for RAND it is only 57%.

To summarize, the high cardinality of the skyline results causes a high amount of data to be transferred in the network, whereas the proposed approach significantly restricts the number of transferred points. This leads to important gains in bandwidth consumption and, subsequently, energy savings for the mobile devices. Even though the exact skyline set can not be guaranteed, our experiments have shown that result sets of high quality are computed and moreover a high fraction of transferred data are skyline points.

7. CONCLUSIONS

In a mobile environment, exact skyline computation requires transferring the local skyline points of the mobile devices to the server, which computes the skyline set based on received points. However, the high cardinality of the local skyline sets leads to excessive communication costs. In this paper, we study bandwidth-constrained skyline queries. In such queries, a fixed amount of data is transferred to the server, by requesting only a limited set of the most promising local skyline points from each mobile device, thus leading to communication savings. The goal is to retrieve as many skyline points as possible, even if only a fraction of the local skyline points are transferred through the network. For this purpose, we study different methods that intentionally select the local skyline points that most probably belong to the skyline result set. In our experimental evaluation, we study the performance of our approach with respect to the quality of the retrieved skyline set, showing that in the most cases results of high quality are retrieved.

8. REFERENCES

[1] W.-T. Balke, U. Güntzer, and J. X. Zheng. Efficient distributed skylining for web information systems. In *Int. Conf. on Extending Database Technology (EDBT)*, pages 256–273, 2004.

[2] S. Börzsönyi, D. Kossmann, and K. Stocker. The skyline operator. In *Int. Conf. on Data Engineering (ICDE)*, pages 421–430, 2001.

[3] C. Y. Chan, H. V. Jagadish, K.-L. Tan, A. K. H. Tung, and Z. Zhang. Finding k-dominant skylines in high dimensional space. In *Int. Conf. on Management of Data (SIGMOD)*, pages 503–514, 2006.

[4] C. Y. Chan, H. V. Jagadish, K.-L. Tan, A. K. H. Tung, and Z. Zhang. On high dimensional skylines. In *Int. Conf. on Extending Database Technology (EDBT)*, pages 478–495, 2006.

[5] L. Chen, B. Cui, H. Lu, L. Xu, and Q. Xu. iSky: Efficient and progressive skyline computing in a structured P2P network. In *Int. Conf. on Distributed Computing Systems*, pages 160–167, 2008.

[6] B. Cui, H. Lu, Q. Xu, L. Chen, Y. Dai, and Y. Zhou. Parallel distributed processing of constrained skyline queries by filtering. In *Int. Conf. on Data Engineering (ICDE)*, pages 546–555, 2008.

[7] K. Fotiadou and E. Pitoura. BITPEER: continuous subspace skyline computation with distributed bitmap indexes. In *Int. Workshop on Data Management in Peer-to-Peer Systems (DAMAP)*, pages 35–42, 2008.

[8] P. Godfrey. Skyline cardinality for relational processing. In *Foundations of Information and Knowledge Systems (FoIKS)*, pages 78–97, 2004.

[9] K. Hose, C. Lemke, and K.-U. Sattler. Processing relaxed skylines in PDMS using distributed data summaries. In *Int. Conf. on Information and Knowledge Management (CIKM)*, pages 425–434, 2006.

[10] Z. Huang, C. S. J. H. Lu, and B. C. Ooi. Skyline queries against mobile lightweight devices in MANETs. In *Int. Conf. on Data Engineering (ICDE)*, page 66, 2006.

[11] X. Lin, Y. Yuan, Q. Zhang, and Y. Zhang. Selecting stars: the k most representative skyline operator. In *Int. Conf. on Data Engineering (ICDE)*, pages 86–95, 2007.

[12] E. Lo, K. Y. Yip, K.-I. Lin, and D. W. Cheung. Progressive skylining over web-accessible databases. *Data Knowledge Engineering (DKE)*, 57(2):122–147, 2006.

[13] D. Papadias, Y. Tao, G. Fu, and B. Seeger. Progressive skyline computation in database systems. *ACM Transactions on Database Systems (TODS)*, 30(1):41–82, 2005.

[14] J. Pei, W. Jin, M. Ester, and Y. Tao. Catching the best views of skyline: A semantic approach based on decisive subspaces. In *Int. Conf. on Very Large Data Bases (VLDB)*, pages 253–264, 2005.

[15] Y. Tao, X. Xiao, and J. Pei. Subsky: Efficient computation of skylines in subspaces. In *Int. Conf. on Data Engineering (ICDE)*, page 65, 2006.

[16] A. Vlachou, C. Doulkeridis, and Y. Kotidis. Angle-based space partitioning for efficient parallel skyline computation. In *Int. Conf. on Management of Data (SIGMOD)*, pages 227–238, 2008.

[17] A. Vlachou, C. Doulkeridis, Y. Kotidis, and M. Vazirgiannis. SKYPEER: Efficient subspace skyline computation over distributed data. In *Int. Conf. on Data Engineering (ICDE)*, pages 416–425, 2007.

[18] S. Wang, B. C. Ooi, A. K. H. Tung, and L. Xu. Efficient skyline query processing on peer-to-peer networks. In *Int. Conf. on Data Engineering (ICDE)*, pages 1126–1135, 2007.

[19] S. Wang, Q. H. Vu, B. C. Ooi, A. K. H. Tung, and L. Xu. Skyframe: a framework for skyline query processing in peer-to-peer systems. *VLDB Journal*, 18(1):345–362, 2009.

[20] P. Wu, C. Zhang, Y. Feng, B. Y. Zhao, D. Agrawal, and A. E. Abbadi. Parallelizing skyline queries for scalable distribution. In *Int. Conf. on Extending Database Technology (EDBT)*, pages 112–130, 2006.

[21] Y. Yuan, X. Lin, Q. Liu, W. Wang, J. X. Yu, and Q. Zhang. Efficient computation of the skyline cube. In *Int. Conf. on Very Large Data Bases (VLDB)*, pages 241–252, 2005.

[22] L. Zhu, Y. Tao, and S. Zhou. Distributed skyline retrieval with low bandwidth consumption. *Transactions on Knowledge and Data Engineering (TKDE)*, 21(3):384–400, 2009.