Finding Superior Skyline Points from Incomplete Data

Rahul Bharuka and P Sreenivasa Kumar

Department of Computer Science & Engineering
Indian Institute of Technology Madras
Chennai, India
{rahulb, psk}@cse.iitm.ac.in

ABSTRACT

The skyline query has proven to be an important tool in multi-criteria decision making and search space pruning. A skyline query returns the subset of points from a multi-dimensional dataset that are not dominated by any other point. Due to its wide applications, skyline query and its variants have been extensively studied in the past. However, skyline computation for incomplete domain, where points have missing values for some dimensions, has not received enough attention. The existing solutions for such incomplete datasets use weak pareto dominance relation which is non-transitive and cyclic. Hence, many of the desirable points are not included in the skyline. Consequently, the skyline no longer offers a reliable overview of the dataset. Moreover, the skyline set returned by these methods is unordered and has high cardinality. The end user does not have control over the result size. Therefore, we have adapted the top-k frequent skyline approach proposed for complete datasets to find *interesting* points from incomplete datasets. The proposed approach overcomes the above mentioned drawbacks and returns top-k points ordered by their *fractional skyline frequency*. Experimental results on both synthetic and real world datasets demonstrate the ability of our approach to find superior skyline points from incomplete datasets.

1. INTRODUCTION

Owing to its wide applications in multi-criteria decision making and search space pruning, skyline queries have been extensively studied in the last decade. Given a set of points in d-dimensional space $\mathcal{S}$, a *skyline* query returns all the points that are not *dominated* by any other point in the set. Here, a point p is said to dominate another point q, if p is *better than or equal to* q in all dimensions and p is *strictly better than* q in some dimension s_i of $\mathcal{S}$. This dominance relation is also known as *pareto dominance* in the literature. Such non-dominated points returned by a skyline query are called *skyline points* and the set of skyline points is known as the *skyline*.

The 19th International Conference on Management of Data (COMAD), 19th-21st Dec, 2013 at Ahmedabad, India.

Consider a customer looking for a mobile phone based on features such as price, battery backup and screen size. Now, take two phones $p_1 = (\$100, 4hrs, 5in)$ and $p_2 = (\$200, 4hrs, 4in)$. Here, p_1 dominates p_2 because each feature of p_1 is at least as good as the corresponding feature of p_2. Such manual examination of all phones is not feasible for large databases. Phones which are not worse than any other phone with respect to all features, are worth considering for the user. Firing a skyline query for such scenario will eliminate all undesirable phones and would present a manageable set of attractive phones to the customer.

Skyline queries and its variants have been extensively studied in the literature. However, with the exception of [4, 13, 17, 29], all skyline query processing approaches assume completeness of data, i.e., values corresponding to all dimensions of data points are known beforehand. But this assumption may not hold in many real world applications such as sensor data, questionnaires and product/service rating websites. In the mobile phone example, the price of certain phone may not be available due to some reasons, making the data point corresponding to that phone *incomplete*. A modified dominance relation, which is called *weak pareto dominance* in this paper, has been used in [4, 13] to compute skyline for such incomplete datasets. In weak pareto dominance, while comparing a pair of points, only the subset of dimensions where values are known for both the points, are considered (ignoring dimensions where value is missing for at least one point). Due to the non-transitive and cyclic nature of this dominance relation, many of the desirable points are not returned in the skyline.

For example, consider points m_1, m_2 and m_3 from the sample movie-rating dataset shown in Table 1. Each entry r_{ij} in the table corresponds to the rating given to movie i by user j on a scale of 1-5. Now, based on weak pareto dominance, m_1 dominates m_2, m_2 dominates m_3, but m_1 does not dominate m_3. Instead, m_3 dominates m_1. Therefore, we can see that, the dominance relation may become non-transitive as well as cyclic. Here, none of the three points belong to the skyline, since each one is dominated by at least one other point. Observe that, m_1 is a desirable point (due to reasonably high ratings given by users) even though it is not a skyline point.

One of the shortcomings of traditional skyline query is the large size of the skyline when the dataset has many dimensions. The reason is, for a pair of points p and q, if p is preferred over q in some dimension s_i and q is preferred over p in dimension s_j ($\neq s_i$), then p and q become *incomparable* i.e., they do not dominate each other. This is the minimum

requirement for any pair of points to become incomparable. While computing skyline for high-dimensional datasets, we are more likely to find such dimensions s_i and s_j. Thus, a large number of points become incomparable with others and belong to the skyline.

In the mobile phone example, in addition to above mentioned features, consider that memory, operating system and camera resolution are also important for the customer. In such a case, most of the phones may have to be included in the skyline since there may not exist a single phone which is better than other in all the features. However, phones which are better than others in majority of the features may exist but are indistinguishable in the skyline set. Thus, having such a large skyline does not offer any interesting insights into the dataset.

Consider another example of NBA[1] dataset that has 17,791 records and 17 dimensions. Its skyline has 1,077 records. Clearly, returning such a long unordered list of records is hardly of any help to a user looking for best players in the NBA league. It would be more appropriate if a manageable set of records, ordered on some *interestingness* measure is returned. Interestingness of a point is a subjective measure and its several variants have been proposed in the past. The most prominent of these are discussed in Section 3.2.

The skyline for incomplete datasets also suffers from the curse of dimensionality. For example, the corresponding 20% incomplete NBA dataset has 300 records in the skyline, which is still quite large. Weak pareto dominance relation needs just one common dimension with known values to compare a pair of incomplete data points. Consequently, points having poor values in common dimensions, will not be included in the skyline. Thus, incomplete data points having few bad values are penalized heavily in weak pareto dominance. Therefore, finding manageable set of superior skyline points from incomplete datasets is difficult and this is the problem we are trying to address in this paper.

One possible way to address this problem is to rank skyline points using some preference function [1, 14]. However, this approach is not always applicable since users need to provide a scoring function i.e., weight assignments pertaining to their preferences. Assigning weights for large number of preferences is difficult and requires sufficient domain knowledge which can not be expected from an average user. While various approaches that return interesting points without using any scoring function have been proposed for complete datasets [7,18,26,28], to our knowledge, the problem of finding and ranking interesting points from incomplete datasets has not been addressed yet.

Skyline frequency [8] is one of the interestingness measures proposed for complete datasets. Given a d-dimensional dataset, the skyline frequency of a point is the number of times it appears in the skyline corresponding to each of the $2^d - 1$ non-empty subspaces possible. The intuition here is that, points that are dominated in fewer combinations of dimensions are *interesting*. This approach ranks points according to their skyline frequency. Therefore, a *top-k frequent skyline query* combines top-k and skyline query features without needing weight assignments.

Skyline frequency is the most suitable interestingness metric that can be adapted for incomplete datasets as it can be used without weak pareto dominance relation. A naïve way of computing top-k frequent skyline points is to first find skyline for each of the $2^d - 1$ subspaces using any of the existing skyline computation approaches and then calculate skyline frequency of each point by counting their occurrences. However, skyline computation is expensive and number of subspaces is exponential in the number of dimensions. Chan et al. [8] have proposed an efficient algorithm to address this problem based on concept of *dominating subspaces*. We have adopted this approach to rank incomplete data points by *fractional skyline frequency*. We call our algorithm as *Incomplete Data Frequent Skyline (IDFS)*. IDFS overcomes the drawback of weak pareto dominance relation by replacing it with pareto dominance relation and succeeds in finding superior skyline points from incomplete datasets. In brief, the contributions of this paper are as follows:

Point	u_1	u_2	u_3	u_4
m_1	2	3	4	-
m_2	-	2	4	2
m_3	3	-	-	1
m_4	1	2	4	2
m_5	-	-	-	5

Table 1: Sample Movie-Rating Dataset

- We identify the problem of finding superior skyline points in the context of incomplete datasets and explain the need for a new solution to the same. To our knowledge, there is no prior work on this problem.
- We introduce *fractional skyline frequency* metric based on the concept of skyline frequency and explain why it is the most suitable metric that can be used for ranking incomplete data skyline points.
- We propose IDFS algorithm that returns superior skyline points from incomplete datasets ordered by their fractional skyline frequency.
- We demonstrate the effectiveness and efficiency of our approach using detailed experiments.

The rest of the paper is organized as follows. Section 2 explains the key concepts used in top-k frequent skyline points computation. The related work is surveyed in Section 3. In Section 4, we give a detailed description of IDFS algorithm. The experimental results are discussed in Section 5 and finally Section 6 concludes the paper.

2. PRELIMINARIES

In this section, we formally define various concepts related to IDFS. We would be using sample movie-rating dataset given in Table 1 as our running example.

Consider an incomplete dataset D, defined on space $\mathcal{S}$, with dimensions $S = \{s_1, s_2, \ldots, s_d\}$. A point $p \in D$ can be represented as $p = \{p.s_1, p.s_2, \ldots, p.s_d\}$ where $p.s_i$ denotes the value of point p on dimension s_i. Missing value for any dimension of a point is represented as '-'. Throughout this paper, without loss of generality, we assume that greater values are preferred over the smaller ones. There are at most $2^d - 1$ distinct non-empty subsets of $\mathcal{S}$. Hereafter, each of them is referred to as a *subspace*.

[1] `http://basketballreference.com`

A data point $p \in D$ is said to be *complete* on subspace $\mathcal{S}'$ defined by dimensions S' if p has known values for all dimensions in S'.

2.1 Subspace Skyline

Consider a pair of points p and q that are *complete* on subspace $\mathcal{S}' \subseteq \mathcal{S}$ defined by dimensions $S' \subseteq S$. p is said to dominate q if and only if $\forall s_i \in S'$, $p.s_i \geq q.s_i$ and $\exists s_j \in S'$, $p.s_j > q.s_j$.

In other words, we use pareto dominance to compare a pair of points having known values for all dimensions in subspace $\mathcal{S}'$. Points that are not *complete* on $\mathcal{S}'$ are not considered for computing skyline of $\mathcal{S}'$. The reason for not using weak pareto dominance relation here, is stated towards the end of this section. Therefore, the skyline of subspace $\mathcal{S}'$ is defined as a subset of points in D that are (1) *complete* on $\mathcal{S}'$ and (2) not dominated by any other point on $\mathcal{S}'$.

In Table 1, while computing skyline of subspace $\mathcal{S}_{12}$ formed by dimension sets $\{u_1, u_2\}$, we consider only points m_1 and m_4, since they are *complete* on $\mathcal{S}_{12}$. Here, m_4 is dominated by m_1 and thus, the skyline of $\mathcal{S}_{12}$ is $\{m_1\}$.

2.2 Skyline Frequency

The skyline frequency of a point p, $f(p)$, is the number of unique subspaces $\mathcal{S}' \subseteq \mathcal{S}$ in which p is a skyline point.

For example, by definition 2.1, point m_2 from Table 1 is a skyline point on subspaces formed by dimensions $\{u_3\}$, $\{u_2, u_4\}$, $\{u_3, u_4\}$ and $\{u_2, u_3, u_4\}$. Thus, $f(m_2) = 4$. Similarly, $f(m_5) = 1$.

Due to the definition of subspace skyline, data points with a large number of missing values will have small skyline frequency. Moreover, skyline frequency of points will not be consistent across the dataset since data points often have varying number of missing values. Therefore, we propose *fractional skyline frequency*, denoted by $f'(p)$, as the skyline frequency of a point divided by the total number of *complete* subspaces for p. That is, $f'(p) = f(p)/(2^k - 1)$, where k is the number of known dimensions for point p. The usefulness of this metric follows from the fact that, a point with k ($<= d$) known dimensions can be dominated in maximum of $2^k - 1$ subspaces. Thus, $f'(m_2) = 4/(2^3 - 1) = 4/7$ and $f'(m_5) = 1/(2^1 - 1) = 1$.

Intuitively, k points with highest fractional skyline frequency in a dataset are called as *top-k frequent skyline points*.

2.3 Dominating Subspace

Given a data point $p \in D$, if $\exists q \in D$ that dominates p on a subspace $\mathcal{S}' \subseteq \mathcal{S}$, then $\mathcal{S}'$ is said to be a *dominating subspace* for p.

In Table 1, subspace $\mathcal{S}_1$ defined by dimension $\{u_1\}$ is a dominating subspace for m_1 since point m_3 dominates m_1 on $\mathcal{S}_1$.

2.4 Dominating Frequency

The *dominating frequency* of a data point $p \in D$, $d(p)$, is the number of unique dominating subspaces for p.

Intuitively, skyline frequency and dominating frequency are duals of each other. Hence, dominating frequency of the point p can be described as: $d(p) = 2^d - 1 - f(p)$. Analogous to the concept of fractional skyline frequency, we propose *fractional dominating frequency* as $d'(p)$. Therefore, $d'(p) = 1 - f'(p)$ and we have, $d'(m_2) = 3/7$ and $d'(m_5) = 0$.

Thus, computing k points with smallest fractional dominating frequencies in a dataset would be same as computing top-k frequent skyline points. We will be using fractional skyline/dominating frequency for computing top-k frequent skyline points.

2.5 Dominating Subspace Set

The set of all subspaces on which a point q dominates another point p is known as *dominating subspaces of q over p* and is denoted by $DS(q, p)$.

This set can be quite large and difficult to enumerate. Hence, a pair of subspaces (U, V) is used to describe it concisely where, (a) $U \subseteq S$ such that $\forall s_i \in U$, $q.s_i > p.s_i$ and (b) $V \subseteq S$ such that $\forall s_i \in V$, $q.s_i = p.s_i$. Now, the total number of subspaces covered by $DS(q, p)$ is given by, $|DS(q, p)| = (2^{|U|} - 1)2^{|V|}$.

Consider points m_1 and m_2 from Table 1. m_1 dominates m_2 in subspaces $\mathcal{S}_2 = \{u_2\}$ and $\mathcal{S}_{23} = \{u_2, u_3\}$. Thus, $DS(m_1, m_2) = \{\mathcal{S}_2, \mathcal{S}_{23}\}$ and can also be represented as $(U, V) = (\{u_2\}, \{u_3\})$.

A crucial lemma from [8] for defining Maximal Dominating Subspace Set is given below:

Lemma: Consider a pair of dominating subspace sets $DS(q, p) = (U_q, V_q)$ and $DS(r, p) = (U_r, V_r)$, where $U_q \neq \emptyset$ and $U_r \neq \emptyset$. $DS(q, p)$ covers $DS(r, p)$ if and only if (a) $(U_r \cup V_r) \subseteq (U_q \cup V_q)$ and (b) $U_r \subseteq U_q$.

2.6 Maximal Dominating Subspace Set

Given a point $p \in D$, if $\nexists r \in D$ such that $DS(r, p)$ covers $DS(q, p)$, then $DS(q, p)$ is said to be a *maximal dominating subspace set* (MDSS) for p. Therefore, dominating subspaces of p is the union of dominating subspaces covered by all such MDSSs for p. Hence, $d(p) = |\bigcup_{M_i \in \mathcal{M}} M_i|$, where $\mathcal{M} = \{DS(q, p) \mid q \in D, DS(q, p)$ is a MDSS for $p\}$.

From Table 1, let us calculate $DS(q, m_4)$ for each point $q \in D \backslash \{m_4\}$. Then we have $DS(m_1, m_4) = (\{u_1, u_2\}, \{u_3\})$, $DS(m_2, m_4) = \emptyset$, $DS(m_3, m_4) = (\{u_1\}, \emptyset)$ and $DS(m_5, m_4) = (\{u_4\}, \emptyset)$. Here, $DS(m_3, m_4)$ is covered by $DS(m_1, m_4)$ and hence it is not a MDSS of m_4. Whereas, $DS(m_1, m_4)$ and $DS(m_5, m_4)$ do not cover each other. Thus, the set of MDSSs, $\mathcal{M} = \{DS(m_1, m_4), DS(m_5, m_4)\}$. We have $|DS(m_1, m_4)| = 6$ and $|DS(m_5, m_4)| = 1$. Therefore, dominating frequency of m_4, $d(m_4) = 7$ since there are no common dominating subspaces covered by both $DS(m_1, m_4)$ and $DS(m_5, m_4)$. Thus, $d'(m_4) = 7/15$ and similarly, we have $d'(m_1) = 1/7$, $d'(m_2) = 3/7$, $d'(m_3) = 1/3$ and $d'(m_5) = 0$. Hence, top-3 frequent skyline points from Table 1 are m_5, m_1 and m_3, which is fairly reasonable.

Now, consider point m_1 from Table 1. If weak pareto dominance relation is used to compare the points, then $d'(m_1)$ would be $4/7$. The reason is, m_3 dominates m_1 in all subspaces having dimension u_1 i.e., $\{u_1\}$, $\{u_1, u_2\}$, $\{u_1, u_3\}$ and $\{u_1, u_2, u_3\}$. Though in reality m_1 is dominated only in subspace $\{u_1\}$, due to the nature of dominance relation used, it appears to be dominated in the above mentioned four subspaces. Thus, using weak pareto dominance relation is not appropriate for computing subspace skyline of incomplete datasets. In the proposed approach, $d'(m_1) = 1/7$, which is reasonable since m_1 is dominated in only one subspace.

3. RELATED WORK

In this section, we discuss the existing work on traditional skyline queries, its variants and skyline queries for incomplete domains.

3.1 Traditional Skyline Queries

Börzsönyi et al. [5] first introduced the Skyline operator for relational databases and proposed BNL, D&C and an algorithm using B-tree for skyline evaluation. Since then, various skyline query processing algorithms, using different approaches, have been proposed by the research community. For instance, index structures are used in [15, 18, 22] to progressively report the skyline. Whereas, SFS [9] and SaLSa [3] algorithms sort input data using monotone sorting function.

Skyline queries have been proposed for domains such as partially-ordered [6], spatial [21], uncertain [19] and distributed databases [2]. Comprehensive surveys of skyline query processing in highly distributed environments and uncertain domains are presented in [10] and [24], respectively. Unfortunately, all of the above mentioned skyline evaluation approaches were proposed for complete data and can not be easily adapted for incomplete data.

3.2 Variants of Skyline Queries

The traditional dominance relation is quite strict, resulting in large size of the skyline. In case of correlated datasets, a point good in one dimension tends to be good in all other dimensions as well. Consequently, few "good" points dominate large number of points, resulting in small size of the skyline. Hence, several variants of traditional skyline query have been proposed to control the skyline size and to find the interesting subset of the skyline. These variants employ some interestingness metric to find superior points from the dataset. The skyline returned by these approaches depend on the dominance relation and interestingness metric used. Some of the most prominent variants are discussed below:

Thick skyline [11] query allows users to increase the skyline size by including points within ϵ-distance of skyline points. Papadias et al. [18] proposed *top-k skyline* that ranks skyline points based on user defined monotone scoring function. The authors also proposed *k-skyband* query that returns points that are dominated by at most k other points. This approach allows users to explore other non-skyline points in case the traditional skyline is too small. Zhang et al. [28] proposed the concept of δ-subspace to find *strong skyline points*. A subspace whose skyline contains less than δ points, is called δ-subspace and the union of all skyline points in such δ-subspace is called strong skyline points. The *k-dominant* skyline query [7] relaxes the dominance relation from considering all dimensions to any subset of size k. Therefore, more points get dominated and a few high quality points can be returned. The *top-k representative skyline points* [16] (top-k RPS) finds the k points such that the total number of unique data points dominated by them is maximized. *ϵ-skyline* [25] enables user to increase/decrease the skyline size by allowing points to dominate other if their normalized values are within a constant of ϵ. This approach allows users to assign weights and rank points using built-in order. *Distance based representation skyline* [23] returns the k most representative skyline points of the dataset i.e., k points that minimizes the maximum distance between a non-representative skyline point and its closest representative.

Unfortunately, all the above variants were proposed in the context of complete data. The dominance relation in all these variants will not work if values corresponding to some dimensions of data points are missing. Hence, any of these approaches, if adapted for incomplete domain, would need weak pareto dominance relation for comparing points. This relation is not preferred since it may discard many desirable points from the result set, as explained in Section 1.

3.3 Skyline Queries for Incomplete Data

Khalefa et al. [13] introduced the problem of skyline computation for incomplete datasets. They defined a modified dominance relation (weak pareto dominance) for comparing incomplete data points and proposed three algorithms namely, Replacement, Bucket and ISkyline for skyline computation. Among these, ISkyline is incremental in nature and the most efficient. Zhang et al. [29] proposed to convert an incomplete dataset to corresponding complete dataset by plugging in estimated values for the missing dimensional values. Now, any of the existing skyline computation approaches can be applied to find the skyline over the complete dataset. However, such substitution of values is only viable when incompleteness percentage is small. This approach is also not suitable for critical applications where false results are not acceptable. The Sort-based Incomplete Data Skyline (SIDS) [4] algorithm outperforms ISkyline, but does not support incremental addition of data points.

Since both ISkyline and SIDS algorithms use non-transitive and cyclic weak pareto dominance relation, some of the interesting points may not be included in the skyline. Miao et al. [17] tried to address this problem by proposing k-skyband queries for incomplete data. However, some desirable points may still not be included in the skyline, since a point that is dominated by k other points on a subspace can have better values on other subspaces. Moreover, this approach returns an unordered set of skyline points and the end user does not have control over the size of the skyline. On the other hand, the proposed approach returns exactly k points ordered by their fractional skyline frequency.

4. PROPOSED APPROACH

As mentioned earlier, the naïve approach of computing top-k frequent skyline points for incomplete datasets is to first compute skyline for each of the $2^d - 1$ subspaces using any traditional skyline computation algorithm and then determine the skyline frequency of each point by counting the number of subspaces for which the point belongs to the skyline. Though several efficient approaches have been proposed to compute precise skyline for all subspaces, such a computation is still very expensive (e.g., [20, 27]).

Chan et al. [8] proposed the *frequent skyline* metric and an efficient *top-k frequent skyline* algorithm, based on the concept of MDSS. Their approach first finds the set of MDSSs for a point and then computes the dominating frequency for that point either precisely or approximately. However, this approach was originally proposed for complete datasets. We have adopted it for incomplete datasets to rank skyline points by *fractional skyline frequency* and called it as *Incomplete Data top-k Frequent Skyline (IDFS)*. In the following sections, we explain in detail the working of IDFS.

4.1 Incomplete Data Frequent Skyline

This approach is based on the fact that each dominating subspace of a point p is covered by at least one MDSS for p. Therefore, dominating frequency of p, $d(p)$, is the number of unique subspaces covered by the set of MDSSs ($\mathcal{M}$) for p. Hence, after computing $\mathcal{M}$ and $d(p)$ for each point, k points

Algorithm 1: IDFS(D, S, k)

Input : d-dimensional incomplete dataset D, k
Output: $ResultSet$, the set of top-k frequent skyline points

```
1  initialize fractional frequency threshold θ = 1
2  ResultSet = ∅
3  foreach p ∈ D do
4      initialize M = ∅
5      flag = ComputeSetOfMDSS (p, k, θ, |ResultSet|, M)
6      if flag then
7          d(p) = CountDS(M)
8          d'(p) = d(p)/(2^dimCount(p) − 1)
9          if (|ResultSet| < k) or (d'(p) < θ) then
10             if |ResultSet| = k then
11                 remove the point with highest fractional
                   dominating frequency in ResultSet
12             end
13             insert p into ResultSet
14             update θ to be the highest fractional dominating
               frequency in ResultSet
15         end
16     end
17 end
18 return ResultSet
```

with lowest fractional dominating frequency are returned as the top-k frequent skyline points.

The order of processing data points affects the pruning capacity of a skyline algorithm. Thus, Chomicki et al. [9] proposed to sort data points by some monotone sorting function. One such function is to sort data points in non-increasing order of sum of their dimension values. The intuition here is that, points with higher sum are likely to have higher skyline frequency and thereby help in early pruning of other points. Therefore, we give pre-sorted data as input to the IDFS algorithm.

The pseudo-code of our approach is given in Algorithm 1. The algorithm takes as input, a d-dimensional incomplete dataset D and the number of frequent skyline points to return, k. To avoid explicitly sorting points by their fractional dominating frequency, we initialize θ with threshold of 1 (Step 1). θ keeps track of highest fractional dominating frequency in $ResultSet$. $ResultSet$ is the set of top-k frequent skyline points and is initialized in Step 2.

At the beginning of each iteration, $\mathcal{M}$, the set of MDSSs for a point p, is reset to $\emptyset$ (Step 4). The algorithm computes the set $\mathcal{M}$ for point p in Step 5 by calling procedure $ComputeSetOfMDSS()$ (given in Algorithm 2). This procedure returns $false$ if p can not be a top-k frequent skyline point; and $true$ otherwise. In case p is a potential top-k point, $\mathcal{M}$ would be the set of MDSSs when Algorithm 2 terminates. For such cases, the dominating frequency, $d(p)$, is computed by calling $CountDS(\mathcal{M})$ (Step 7). The fractional dominating frequency of p, $d'(p)$, is calculated in Step 8 by using $dimCount(p)$, which is the number of known dimensions for p. If $ResultSet$ has k points and $d'(p)$ is smaller than θ, then the point with the highest fractional dominating frequency is removed (Step 10-12) and p is inserted into $ResultSet$ (Step 13). On the other hand, when $ResultSet$ has less than k points, point p is directly inserted into $ResultSet$. The value of θ is updated in Step 14 to the highest fractional dominating frequency in $ResultSet$. Finally, the algorithm terminates when all points have been processed and returns $ResultSet$ as the set of top-k frequent skyline points.

Algorithm 2: ComputeSetOfMDSS($p, k, \theta, r, \mathcal{M}$)

Input : $p, k, \theta, r, \mathcal{M}$
Output: boolean value and $\mathcal{M}$ for point p is computed in part or full

```
1  initialize chkPoint = 2
2  foreach q ∈ D\{p} do
3      let U ⊆ S such that ∀s_i ∈ U, q.s_i > p.s_i
4      let V ⊆ S such that ∀s_i ∈ V, q.s_i = p.s_i
5      if (r = k) and
       ((((2^|U| − 1)2^|V|)/(2^dimCount(p) − 1)) > θ) then
6          return false
7      end
8      if (r = k) and (|M| = chkPoint) then
9          d(p) = CountDS(M)
10         d'(p) = d(p)/(2^dimCount(p) − 1)
11         if (d'(p) > θ) then
12             return false
13         end
14         chkPoint = 2 * chkPoint
15     end
16     initialize isMaximal = true
17     foreach MDSS (P, Q) ∈ M do
18         if (U ∪ V ⊆ P ∪ Q) and (U ⊆ P) then
19             isMaximal = false
20             break out of the loop
21         end
22         else if (P ∪ Q ⊆ U ∪ V) and (P ⊆ U) then
23             remove (P, Q) from M
24         end
25     end
26     if isMaximal then
27         insert (U, V) into M
28     end
29 end
30 return true
```

4.2 Computing the set of MDSSs

The procedure *ComputeSetOfMDSS()* that computes $\mathcal{M}$ for a point p is given in Algorithm 2. Apart from p, the procedure takes k, θ and cardinality of $ResultSet$ as input. It returns $false$ if p is determined not to be a top-k frequent skyline point and $true$ otherwise. The procedure compares p with all other points $q \in D$ one at a time and computes the dominating subspaces of q over p, $DS(q, p)$, as a subspace pair (U, V) described earlier. The pair (U, V) is determined by comparing points p and q across individual dimensions (Step 3-4). Here, we note that, only the subset of dimensions where values are known for both the points are considered. Steps 5-15 list two optimizations which are explained later in this section. Steps 17-25 check whether subspace (U, V) is a MDSS or not. By lemma 1, if (U, V) is covered by some MDSS $(P, Q) \in \mathcal{M}$, then (U, V) is not a MDSS for p and is ignored immediately (Step 18-21). Similarly, if (P, Q) is covered by (U, V) then it is not a MDSS and is thus removed from $\mathcal{M}$ (Step 22-24). Finally, if (U, V) is not covered by any $(P, Q) \in \mathcal{M}$ then it is added to set $\mathcal{M}$ (Step 26-28). The procedure terminates once p has been compared with all other points unless terminated early because of optimization.

Steps 5-15 list two optimizations performed in the algorithm for early pruning of points. The idea here is to avoid exact computation of set $\mathcal{M}$ when a point is determined not to be a top-k frequent skyline point. Specifically, in the first optimization, if $ResultSet$ has k points and the fractional dominating frequency of p, considering only $DS(q, p)$,

exceeds θ (the highest fractional dominating frequency in $ResultSet$), then point p can not be a top-k frequent skyline point. Therefore, further computation of $\mathcal{M}$ is terminated and $false$ is returned to prune point p (Step 5-7).

We have also implemented check-point optimization as proposed in [8] and it is explained below. This optimization periodically checks whether a point p can be pruned early. This can be achieved by checking if "$ResultSet$ has k points and $d'(p)$ exceeds the threshold θ" (pruning test). If it does, then the procedure terminates early and returns $false$ (Steps 8-15). In the first optimization, the pruning test is applied only for a single MDSS $DS(q,p)$. Data points often have multiple MDSSs in $\mathcal{M}$. It is possible that a single $DS(q,p)$ may not cover too many dominating subspaces but the total number of unique subspaces covered by all sets in $\mathcal{M}$ is large enough to pass the pruning test. Thus, it would be beneficial to perform pruning test at regular "checkpoints". Once a check point is reached, procedure $CountDS()$ is invoked to count the unique dominating subspaces covered by intermediate set $\mathcal{M}$ and then pruning test is performed. We set check points when $|\mathcal{M}| = 2^t$, where $t = 1, 2, \ldots, n$.

4.3 Counting of Dominating Subspaces

Once $\mathcal{M}$, the set of MDSSs for a point p, has been computed, we need to determine $d(p)$ (the number of unique dominating subspaces for p). Data points usually have several maximal dominating subspace sets (M_i) and thus some of the subspaces covered by them overlap. Hence, each subspace in $\mathcal{M}$ needs to be checked for duplicates and should be counted only once.

Chan et al. [8] proposed two approaches for counting the number of dominating subspaces viz. (1) Exact Counting and (2) Approximate Counting. Both of these approaches take $\mathcal{M}$ as input and return $d(p)$ as the answer. We have adopted both of these approaches as they are. The detailed description and pseudo-code of these approaches can be found in [8] and are explained in brief below.

4.3.1 Exact Counting

This approach counts dominating subspaces by enumerating all dominating subspaces covered by $\mathcal{M}$. That is, for each MDSS $M_i \in \mathcal{M}$, the procedure enumerates all dominating subspaces $(P,Q) \in M_i$ and checks if (P,Q) has been covered by some M_j for $1 \leq j < i$. As the number of subspaces is exponential in the number of dimensions, this approach does not scale for high dimensions.

4.3.2 Approximate Counting

This approach is based on Monte-Carlo Counting Algorithm [12]. The procedure takes two parameters viz δ and ϵ, along with $\mathcal{M}$ as input. These two parameters determine the number of samples (T) to be generated and is given by $T = 2n \ln(2/\delta)/\epsilon^2$. The idea here is to count unique dominating subspaces found among T repeatable samples and using that, approximate the total number of unique dominating subspaces. The approximate answer returned by this procedure has maximum error of ϵ with confidence of at least 1 - δ.

5. EXPERIMENTAL RESULTS

This section discusses the experimental results and settings used for IDFS algorithm. We have implemented both Exact Count (IDFS-EC) and Approximate Count (IDFS-AC) variant of Incomplete Data Top-k Frequent Skyline (IDFS) algorithm in C++. The Approximate Count Without Sorting (ACWS) and Approximate Count Without Checkpoint (ACWC) variants were not implemented for IDFS since they were reported to perform worse than AC for complete data in [8]. Since this is the first attempt to find superior skyline points from incomplete datasets, we could not compare the performance results with any of the existing approaches for incomplete data skyline computation.

5.1 Experimental Settings

We have evaluated IDFS-AC and IDFS-EC algorithms on both synthetic as well as real world datasets. Synthetic datasets of varying data size, dimensions and distributions, have been generated using synthetic data generator provided by Börzsönyi et al. [5]. In particular, synthetic datasets having correlated, independent or anti-correlated distribution have been used for evaluation.

We have also evaluated our approach on real world NBA dataset. This dataset has 17791 records about 17 basketball skills of players during regular season in the period 1946-2005. The dataset has multiple records for a single player corresponding to each year he has played in the league. NBA is a correlated dataset where all attribute values are positively correlated. Since NBA is a complete dataset, we have explicitly removed values from it (in random fashion) to generate 20% incomplete dataset.

Unless specified otherwise, the default parameters for experiments are: 20% incomplete 15-dimensional independent synthetic dataset having 100K data points, $\epsilon = 0.2$, $\delta = 0.05$ and $k = 10$. All the experiments are performed on Intel Quad-core 2.83GHz processor machine, having 4GB RAM and running Ubuntu 12.04 LTS with kernel 3.5.0-34-generic. We report performance based on execution time taken by algorithms.

Note that, top-k frequent skyline points returned by both Top-k Frequent Skyline Algorithm (EC) [8] and IDFS-EC on any *complete dataset* are exactly same and follow the same order.

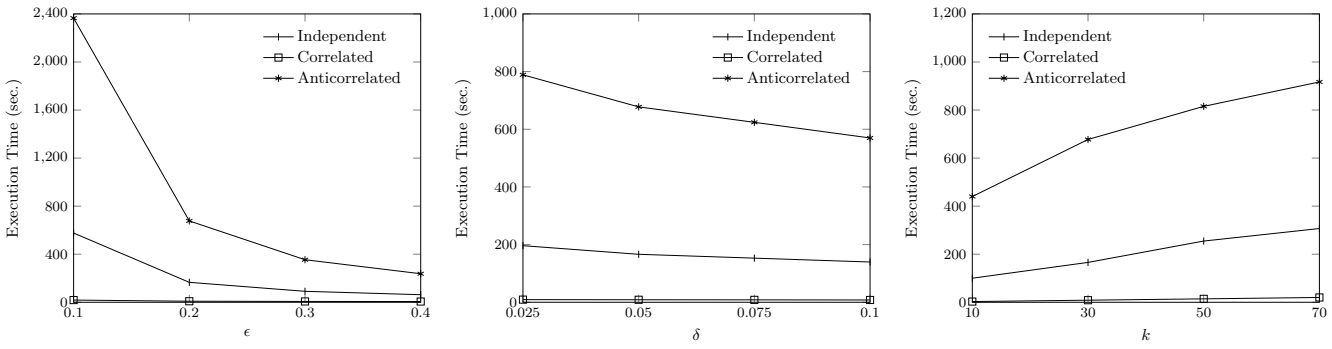
5.2 Performance Results

The first three subsections deal with the synthetic datasets, while the last one is about NBA dataset.

5.2.1 Tuning δ and ϵ

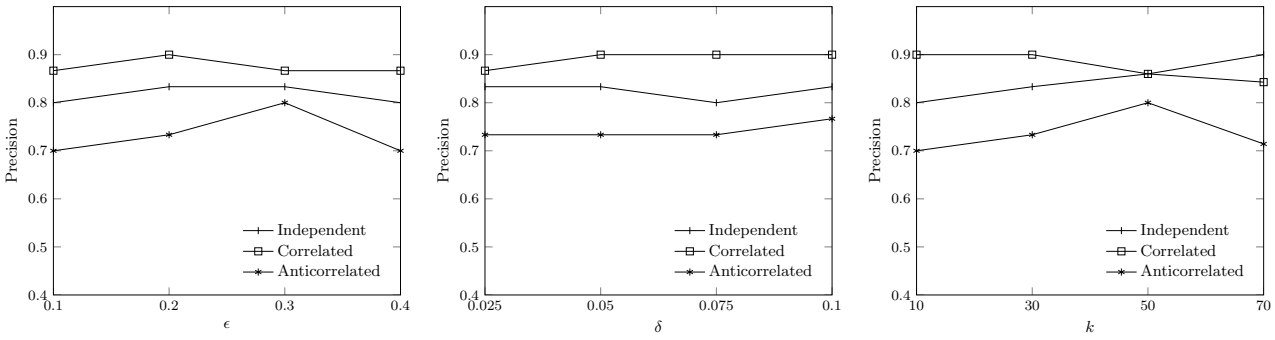
The AC variant of IDFS algorithm requires δ and ϵ as two additional input parameters. These parameters determine the number of samples to be drawn for approximate counting of dominating frequency. In this section, we discuss their effect on execution time and precision of the algorithm. Note that, the default value of k is set to 30 for these experiments.

Figure 1 shows the execution time taken by IDFS-AC algorithm as a function of ϵ, δ and k. The number of samples in $ApproxCountDS()$ procedure is calculated as, $T = 2n \ln(2/\delta)/\epsilon^2$. Since T is inversely proportional to $1/\epsilon^2$, the execution time of the algorithm decreases steadily when ϵ is varied from 0.1 to 0.4 (Fig. 1(a)). In contrast, when δ is varied from 0.025 to 0.1, the value of T does not change much since it is linearly related to $\ln(2/\delta)$. Hence, the execution time taken by the algorithm in Fig. 1(b), is stable. From Fig. 1(c), the execution time of the algorithm increases almost linearly with k since more points are returned.



(a) epsilon (b) delta (c) k

Figure 1: Efficiency with respect to different parameters



(a) epsilon (b) delta (c) k

Figure 2: Precision with respect to different parameters

In Fig. 2, we study the precision of IDFS-AC algorithm with respect to parameters ϵ, δ and k. Here, *precision* is defined as the fraction of top-k frequent skyline points returned by IDFS-AC variant compared to those returned by IDFS-EC. As shown in the figure, IDFS-AC has precision of at least 70% on all data distributions and parameter values. The precision of the algorithm is known to decrease with higher value of ϵ because of less number of samples drawn (Fig. 2(a)). However, the precision is very stable with respect to parameters δ and k. In terms of data distributions, correlated datasets have highest precision that is always close to 85%. Whereas precision on anti-correlated datasets is least, and found to be a bit inconsistent. Datasets with independent distribution have precision between that of correlated and anti-correlated datasets.

From these experiments, we can see that IDFS-AC algorithm has good enough execution time and precision when $\epsilon = 0.2$ and $\delta = 0.05$. Therefore, we will be using these parameter values in the rest of our experiments.

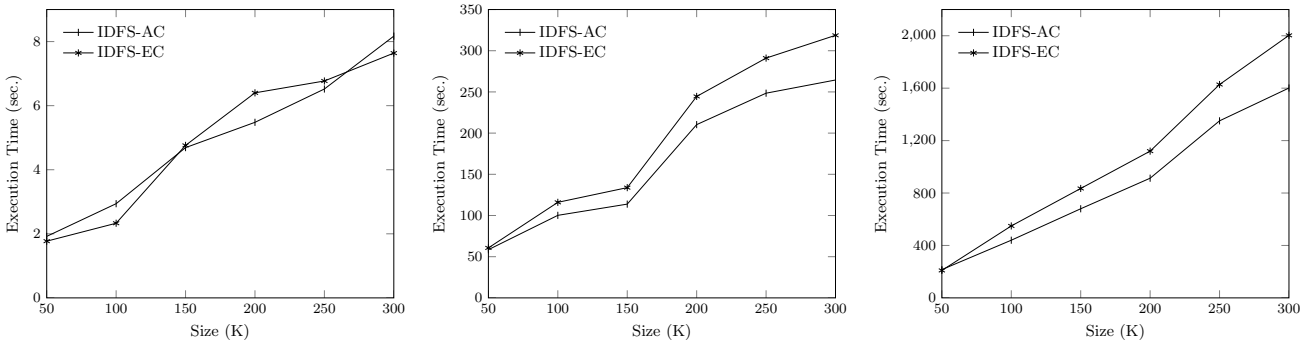
5.2.2 Scalability

This experiment compares the execution time of algorithms when data size is varied from 50K to 300K. As shown in Figure 3, the execution time taken by both algorithms grows linearly with the data size. From Fig. 3(a), IDFS-EC outperforms IDFS-AC when dataset is correlated (for majority of the cases). This is because, top-k points in correlated datasets have small dominating frequencies. Hence, the number of dominating subspaces enumerated by IDFS-EC is far lesser than the number of samples required by IDFS-AC. On the other hand, for independent and anti-correlated datasets, IDFS-AC steadily outperforms IDFS-EC.

5.2.3 Dimensionality

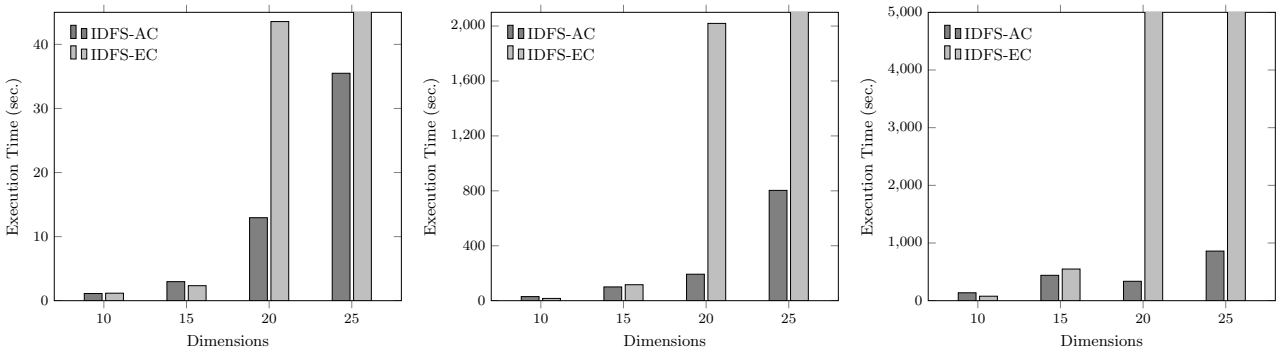
Fig. 4 shows the effect of varying the number of dimensions, d, on execution time of algorithms. For this experiment, d is varied from 10 to 25. When $d = 10$, algorithm IDFS-EC outperforms IDFS-AC. The reason is, on lesser dimensions, the total number of dominating subspaces (including duplicates) covered by set of MDSSs ($\mathcal{M}$) is far less than the number of samples required for approximate counting. On the other hand, for large dimensions, the number of samples needed is much less than the number of dominating subspaces enumerated for precise counting. Hence, for high-dimensional datasets, execution time of IDFS-AC is far less than that of IDFS-EC.

Observe that, in Fig. 4(c), the execution time required by IDFS-AC for processing the 20-dimensional anti-correlated dataset is slightly lower than that of 15-dimensional dataset. The reason is, compared to 15-dimensional dataset, more points get pruned early in 20-dimensional dataset. Note that, some of the bars corresponding to IDFS-EC algorithm in Fig. 4 are truncated to the maximum value plotted in the graph because of their very large values. For instance, in Fig. 4(b), IDFS-EC takes around 25,500 seconds for com-



(a) Correlated Synthetic Dataset (b) Independent Synthetic Dataset (c) Anticorrelated Synthetic Dataset

Figure 3: Scalability



(a) Correlated Synthetic Dataset (b) Independent Synthetic Dataset (c) Anticorrelated Synthetic Dataset

Figure 4: Dimensionality

Table 2: Top-10 Frequent Skyline for (a) Complete NBA Dataset (b) 20% Incomplete NBA Dataset. (c) Effect of varying incompleteness percentage of datasets on match count.

(a)

Player Name	Season
Wilt Chamberlain	**1961**
Michael Jordan	**1986**
Michael Jordan	**1987**
George Mcginnis	**1974**
Michael Jordan	1988
Bob Mcadoo	**1974**
Julius Erving	**1975**
Charles Barkley	1987
Kobe Bryant	**2002**
Kareem Abdul-Jabbar	1975

(b)

Player Name	Season
Wilt Chamberlain	**1961**
Bob Mcadoo	**1974**
Julius Erving	**1975**
Jerry Stackhouse	2000
George Mcginnis	**1974**
Kobe Bryant	**2002**
Michael Jordan	**1987**
Michael Jordan	1989
Michael Jordan	**1986**
Charles Barkley	1985

(c)

Incomplete Percentage	Match Count			
	NBA	Corr	Indep	Anti
10	8	6	5	3
20	7	6	2	2
30	7	5	1	3
40	4	2	2	1
50	4	2	1	0

puting top-10 frequent skyline points on 25-dimensional independent synthetic dataset. The effect of dimensionality is approximately the same for all the data distributions. However, anti-correlated datasets (Fig. 4(c)) have higher execution time than others.

5.2.4 NBA Dataset

On 20% incomplete NBA dataset, IDFS-AC algorithm takes 1.08 seconds against 1.54 seconds taken by IDFS-EC. We do not report scalability and dimensionality results for NBA dataset since its characteristics are very similar to correlated synthetic datasets.

5.3 Quality of top-k points

In this section we discuss the ability of our approach in finding truly *interesting* points from an incomplete dataset.

Tables 2(a) shows the top-10 records returned by IDFS-EC algorithm on *complete* NBA dataset. People who follow

basketball will agree that these are some of the best NBA players. Thus, *skyline frequency* metric indeed gives most interesting points from the dataset. For the corresponding 20% incomplete NBA dataset, the skyline returned by ISkyline [13] and SIDS [4] algorithms has 300 records. The kISB [17] algorithm allows to further increase the skyline size by increasing k. Sine these approaches use weak pareto dominance relation, many non-desirable points are included in the skyline and its difficult to isolate the interesting points among them. On the other hand, for the same 20% incomplete NBA dataset, IDFS-AC algorithm returns top-10 records ranked by their *fractional skyline frequency* (Table 2(b)). Rows with bold text font denote matching records between Table 2(a) and 2(b). As can be seen from the tables, same 7 records are returned in top-10 (neglecting the order) even when data is 20% incomplete.

Table 2(c) summarizes the *match count*, i.e. the number of top-10 frequent skyline points for an incomplete dataset (using IDFS-AC) matching with that of corresponding *complete* dataset (using IDFS-EC). Here, incompleteness percentage of datasets is varied from 10 to 50%. The match count is highest for correlated datasets (both NBA and synthetic) and least for anti-correlated ones. The number of such matching records drops gradually with the increase in incompleteness percentage for all the datasets. From this experiment, we can see that, the proposed approach returns reasonable matching records even though the dataset has up to 50% missing values. Therefore, we conclude that, our approach can return high-quality points from an incomplete dataset.

6. CONCLUSION

In this paper, we introduced an efficient algorithm, called Incomplete Data Frequent Skyline (IDFS), to find superior skyline points from incomplete datasets. To the best of our knowledge, ours is the first attempt to address this problem. There is a need for such an approach, since all existing approaches for skyline computation over incomplete data use weak pareto dominance relation, and may discard many interesting points. We have adapted the *skyline frequency* interestingness measure for incomplete data and proposed *fractional skyline frequency*. Our approach returns k points ordered by their fractional skyline frequency. Experimental results demonstrate the superior quality of our skyline points over the ones given by any of the existing approaches for incomplete data skyline computation. In the future, we intend to improve the efficiency of our algorithm by exploring alternative techniques that can reduce the examined space for determining frequent skyline points.

7. REFERENCES

[1] R. Agrawal and E. L. Wimmers. A framework for expressing and combining preferences. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 297–306, New York, NY, USA, 2000. ACM.

[2] W.-T. Balke, U. Güntzer, and J. X. Zheng. Efficient distributed skylining for web information systems. In *Advances in Database Technology - EDBT*, volume 2992, pages 256–273. Springer, Berlin, Heidelberg, 2004.

[3] I. Bartolini, P. Ciaccia, and M. Patella. Salsa: computing the skyline without scanning the whole sky. In *Proceedings of the 15th International Conference on Information and Knowledge Management*, pages 405–414, New York, NY, USA, 2006. ACM.

[4] R. Bharuka and P. S. Kumar. Finding skylines for incomplete data. In *Proceedings of the 24th Australasian Database Conference*, pages 109–117, Adelaide, Australia, 2013. Australian Computer Society.

[5] S. Börzsönyi, D. Kossmann, and K. Stocker. The skyline operator. In *Proceedings of the 17th International Conference on Data Engineering*, pages 421–430, Washington, DC, USA, 2001. IEEE Computer Society.

[6] C.-Y. Chan, P.-K. Eng, and K.-L. Tan. Stratified computation of skylines with partially-ordered domains. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 203–214, New York, NY, USA, 2005. ACM.

[7] 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 *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 503–514, New York, NY, USA, 2006. ACM.

[8] C.-Y. Chan, H. V. Jagadish, K.-L. Tan, A. K. H. Tung, and Z. Zhang. On high dimensional skylines. In *Proceedings of the 10th international conference on Advances in Database Technology*, pages 478–495, Berlin, Heidelberg, 2006. Springer-Verlag.

[9] J. Chomicki, P. Godfrey, J. Gryz, and D. Liang. Skyline with presorting. In *Proceedings of the 19th International Conference on Data Engineering*, pages 717–719, Washington, DC, USA, 2003. IEEE Computer Society.

[10] K. Hose and A. Vlachou. A survey of skyline processing in highly distributed environments. *The VLDB Journal*, 21(3):359–384, 2012.

[11] W. Jin, J. Han, and M. Ester. Mining thick skylines over large databases. In *Proceedings of the 8th European Conference on Principles and Practice of Knowledge Discovery in Databases*, pages 255–266, New York, NY, USA, 2004. Springer-Verlag New York, Inc.

[12] R. M. Kapp, M. Luby, and N. Madras. Monte-carlo approximation algorithms for enumeration problems. *J. Algorithms*, 10(3):429–448, 1989.

[13] M. E. Khalefa, M. F. Mokbel, and J. J. Levandoski. Skyline query processing for incomplete data. In *Proceedings of the 24th International Conference on Data Engineering*, pages 556–565, Washington, DC, USA, 2008. IEEE Computer Society.

[14] W. Kießling. Foundations of preferences in database systems. In *Proceedings of the 28th International Conference on Very Large Data Bases*, pages 311–322. VLDB Endowment, 2002.

[15] D. Kossmann, F. Ramsak, and S. Rost. Shooting stars in the sky: an online algorithm for skyline queries. In *Proceedings of the 28th International Conference on Very Large Data Bases*, pages 275–286. VLDB Endowment, 2002.

[16] X. Lin, Y. Yuan, Q. Zhang, and Y. Zhang. Selecting stars: The k most representative skyline operator. In

Proceedings of the 23rd International Conference on Data Engineering, pages 86–95, 2007.

[17] X. Miao, Y. Gao, L. Chen, G. Chen, Q. Li, and T. Jiang. On efficient k-skyband query processing over incomplete data. In *Proceedings of the International Conference on Database Systems for Advanced Applications*, volume 7825, pages 424–439. Springer, Berlin Heidelberg, 2013.

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

[19] J. Pei, B. Jiang, X. Lin, and Y. Yuan. Probabilistic skylines on uncertain data. In *Proceedings of the 33rd International Conference on Very Large Data Bases*, pages 15–26. VLDB Endowment, 2007.

[20] J. Pei, W. Jin, M. Ester, and Y. Tao. Catching the best views of skyline: a semantic approach based on decisive subspaces. In *Proceedings of the 31st International Conference on Very Large Data Bases*, pages 253–264. VLDB Endowment, 2005.

[21] M. Sharifzadeh and C. Shahabi. The spatial skyline queries. In *Proceedings of the 32nd International Conference on Very Large Data Bases*, pages 751–762. VLDB Endowment, 2006.

[22] K.-L. Tan, P.-K. Eng, and B. C. Ooi. Efficient progressive skyline computation. In *Proceedings of the 27th International Conference on Very Large Data Bases*, pages 301–310, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

[23] Y. Tao, L. Ding, X. Lin, and J. Pei. Distance-based representative skyline. In *Proceedings of the 25th International Conference on Data Engineering*, pages 892–903, Washington, DC, USA, 2009. IEEE Computer Society.

[24] Y. Wang, X. Li, X. Li, and Y. Wang. A survey of queries over uncertain data. *Knowledge and Information Systems*, (10115):1–46, 2013.

[25] T. Xia, D. Zhang, and Y. Tao. On skylining with flexible dominance relation. In *Proceedings of the 24th International Conference on Data Engineering*, pages 1397–1399, Washington, DC, USA, 2008. IEEE Computer Society.

[26] J. Yang, G. P. Fung, W. Lu, X. Zhou, H. Chen, and X. Du. Finding superior skyline points for multidimensional recommendation applications. *World Wide Web*, 15(1):33–60, 2012.

[27] Y. Yuan, X. Lin, Q. Liu, W. Wang, J. X. Yu, and Q. Zhang. Efficient computation of the skyline cube. In *Proceedings of the 31st International Conference on Very Large Data Bases*, pages 241–252. VLDB Endowment, 2005.

[28] Z. Zhang, X. Guo, H. Lu, A. K. H. Tung, and N. Wang. Discovering strong skyline points in high dimensional spaces. In *Proceedings of the 14th International Conference on Information and Knowledge Management*, pages 247–248, New York, NY, USA, 2005. ACM.

[29] Z. Zhang, H. Lu, B. C. Ooi, and A. K. Tung. Understanding the meaning of a shifted sky: a general framework on extending skyline query. *The VLDB Journal*, 19(2):181–201, 2010.