ISSA: Efficient Skyline Computation for Incomplete Data

Kaiqi Zhang(✉), Hong Gao, Hongzhi Wang, and Jianzhong Li

School of Computer Science and Technology, Harbin Institute of Technology, Harbin, China
{zhangkaiqi,honggao,wangzh,lijzh}@hit.edu.cn

Abstract. Over the past years, the skyline query has already caused wide attention in database community. For the skyline computation over incomplete data, the existing algorithms focus mainly on reducing the dominance tests among these points with the same bitmap representation by exploiting *Bucket* technique. While, the issue of exhaustive comparisons among those points in different buckets remains unsolved, which is the major cost. In this paper, we present a general framework COBO for skyline computation over incomplete data. And based on COBO, we develop an efficient algorithm ISSA in two phases: *pruning compared list* and *reducing expected comparison times*. We construct a compared list order according to ACD to diminish significantly the total comparisons among the points in different buckets. The experimental evaluation on synthetic and real data sets indicates that our algorithm outperforms existing state-of-the-art algorithm 1 to 2 orders of magnitude in comparisons.

1 Introduction

The skyline query coined out by Börzsönyi et al. in [1] is an important query in database community. Over the years, it has caused wide attention to researchers, especially in multi-criteria decision making field. For a data set S with multiple dimensions, the skyline query retrieves out the interesting points which are not dominated by others. Any two points p and q in S, p dominates q if p is smaller or equal to q in every dimension and smaller than q in at least one dimension.

The skyline query can diminish the amount of the concern points down to the skyline set and non-skyline points will never take up users' decision time. In the light of the importance of the skyline query, a series of efficient algorithms [1,3,9] for the skyline computation have been proposed. They are devised based on the *completeness assumption*, *i.e.*, all dimensions are available for every data point. While there are many real life scenarios do not hold this assumption. For instance, there is a movie rating data set (MovieLens [5]). Users usually rate a few of all the movies, which results in many *null* value for the movie that a user does not rate. In this incomplete data set, [6] gives the new definition about skyline, which makes all aforementioned traditional algorithms invalid. Also, proposes a *ISkyline* algorithm. It reduces the dominance tests among these points

H. Gao et al. (Eds.): DASFAA 2016 Workshops, LNCS 9645, pp. 321–328, 2016.
DOI: 10.1007/978-3-319-32055-7_26

with the same bitmap representation by exploiting *Bucket* technique. While, the issue of exhaustive comparisons among those points in different buckets remains unsolved, which is the major cost for computing skyline set over incomplete data.

In this paper, we first illustrate the challenge brought by *non-transitive* dominance relation for incomplete data set. Then, a general framework for skyline computation over incomplete data, named COBO, is presented to accommodate the relation. Later, based on COBO, we propose an efficient algorithm ISSA in two phases: *pruning compared list* and *reducing expected comparison times*. *Pruning compared list* could delete safely these points which are eliminated that have no effect on the result of skyline computation. And *reducing expected comparison times* constructs a compared list order according to ACD to strive to diminish the total expected comparisons among points in different buckets. We summarize our contributions as follows:

- We generate a general framework named COBO for skyline computation over incomplete data.
- We analyze that there are only two techniques to improve the general framework: *pruning compared list* and *reducing expected comparison times*.
- We propose a compared list order according to ACD to strive to diminish the total expected comparisons.
- We evaluate our presented algorithm with state-of-the-art algorithm on synthetic and real data sets.

2 Related Work

The skyline query is first pioneered in [1] in the database community, and afterward, many skyline algorithms have been presented, such as SFS [3], OSPS [9] and so on. In addition, many efforts have been paid to the variants of traditional skyline, for instance k-dominant skyline [2], probabilistic skyline [7], reverse skyline [4], and skyline cube [8].

However, for incomplete data set, there are quite few of literatures up to now. The issue is first introduced in [6] which gives the new definition about skyline in incomplete data set. Unfortunately, the transitive dominance relation cannot remain hold, *i.e*, dominance relation become *non-transitive*, which leads to that all the traditional skyline algorithms are not applicable. In the light of this, [6] presents an algorithm ISkyline for skyline computation over incomplete data. ISkyline stores automatically the points with the same bitmap representation into the identical bucket. It reduces the dominance tests among the points in the same bucket, in which the transitive dominance relation become valid. While, the issue of exhaustive comparisons among those points in different buckets remains unsolved, which is the major cost for computing skyline set over incomplete data.

3 Preliminaries

This section first introduces the description about incomplete data set. And then the definitions of dominance and skyline are both given referring to the prior work [6].

Up to now, we have referred to many times of incomplete data, so what is it and how to represent it? In incomplete data set, there maybe exist missing value in any dimension for every data point. For each data point, we call the dimension with missing value as *incomplete* dimension, otherwise the dimension is *complete* dimension. For example, $p(1, -, 4)$ is an incomplete point in 3-dimensional data space, where '–' denotes the missing value. Also, the ith dimensional value of p is represented by p^i, we know that p^3=4. Here, we first explain some symbols used through the paper from now on. Incomplete data set S has d dimensions whose value is positive number. Specially, any data point in S has 1 to $d-1$ *incomplete* dimensions. Now, we introduce formally the definitions about skyline over incomplete data in the following:

Definition 1 (Dominance for Incomplete Data). *Given any two points p, $q \in S$, it is said that p dominates q, denoted by $p \succ q$, if and only if the following conditions both satisfy:*

(1) $\exists i \in [1, d]$, p^i and q^i both exist and $p^i < q^i$.
(2) $\forall i \in [1, d]$, p^i is missing or q^i is missing or $p^i \leq q^i$.

If $p \succ q$, we call that p is the *dominating point* of q and q is the *dominated point* of p.

Definition 2 (Skyline for Incomplete Data). *Using $SKY(S)$ represents the skyline set of S, where $SKY(S) = \{\forall p \in S \mid \nexists\, q \in S, q \succ p\}$.*

4 A General Framework for Skyline Computation over Incomplete Data

In this section, we first present a general framework named Compared One by One (COBO) for computing skyline result over incomplete data. Based on it, we propose an efficient algorithm ISSA in two phases: *pruning compared list* and *reducing expected comparison times*. And then we illuminate them in detail.

4.1 A General Framework

For the skyline computation over complete data, points hold the transitive dominance relation. For instance, there are three points $p1$ = (2, 3, 2), $p2$ = (3, 3, 4) and $p3$ = (4, 5, 4). $p1$ dominates $p2$ and $p2$ dominates $p3$, therefore $p1$ must dominate $p3$. All existing approaches are based on transitive dominance relation. While, the dominance relation is *non-transitive* [6] among incomplete points according to the Definition 1. An example as follows: $p1$ = (2, 3, –), $p2$ = (4, –, 1), $p3$ = (–, –, 2). Apparently, $p1$ dominates $p2$ and $p2$ dominates $p3$, while $p1$ cannot dominate $p3$. Non-transitive makes it extremely difficult that the skyline computation over incomplete data. Not like computing skyline result in complete data set, once one point is checked out to be non-skyline, it can be removed immediately. Because all points it dominates must be dominated

by the points which dominate it. So traditional skyline algorithms exploiting the property can only maintain a skyline window organized as sorted list [3] or *skyline* tree [9] to justify the subsequent points. Unfortunately, for incomplete data set, it is not safe to discard the points which, even though, are determined to be non-skyline. Any point, whether it is skyline or not, must be contained in the window to check other points.

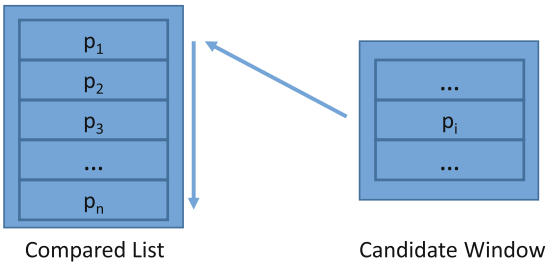


Fig. 1. A general framework: COBO

Based on above analysis, we present a general framework named Compared One by One (COBO) for computing skyline result over incomplete data as shown in Fig. 1. The framework is made of two parts, compared list (CL) and candidate window (CW). Compared list is mainly used to justify whether a point is skyline or not, and it generally contains all points of the data set because of the *non-transitive* dominance relation. For candidate window, it stores these points which maybe belong to the skyline result. Simply initialize candidate window by all points since any point has the possibility to be skyline. Now we will illustrate the process of skyline computation in the framework. After initialization of two parts, every point in candidate window do this operation: For any being checked point p_i in candidate window, generating a compared list order according to p_i. Then compare the points in this order, one by one, with p_i. Once one point dominating p_i occurs, eliminate p_i from candidate window. Finally, the remaining points in candidate window are the skyline result of the incomplete data set.

The basic process of the framework COBO is shown in Algorithm 1. While, its cost is significantly expensive. Now, based on COBO, we propose an efficient algorithm ISSA in two phases: *pruning compared list* and *reducing expected comparison times*.

4.2 Phase I: Pruning Compared List

The first phase is pruning compared list, *i.e.*, remove all these points which are eliminated that have no effect on the result of skyline computation. While, what the points can be removed safely in the compared list? This issue has already been figured out in [6] by introducing the *bucket* technique. These points with the same bitmap representation are stored automatically into the identical bucket. The transitivity of dominance relation becomes valid among the points in the

Algorithm 1. COBO(S)

Input: A d-dimensional positive numerical incomplete data set S
Output: The skyline result of data set S

```
1: Candidate Window candidate = S
2: Compared List CL = S
3: for ∀p ∈ candidate do
4:    determine the order of CL as CL_p according to p.
5:    for ∀q ∈ CL_p do
6:       if q dominates p then
7:          delete p from candidate;
8:          break;
9: return candidate
```

same bucket. Therefore we can straightforwardly discard the dominated points in each bucket.

For example, $p1 = (2, 3, –)$, $p2 = (3, 4, –)$, $p3 = (–, 5, 1)$. The points $p1$ and $p2$ are in the same bucket 110 and $p1$ dominates $p2$. Then we can remove $p2$ safely, the remaining point $p3$ dominated by $p2$ must be dominated by $p1$.

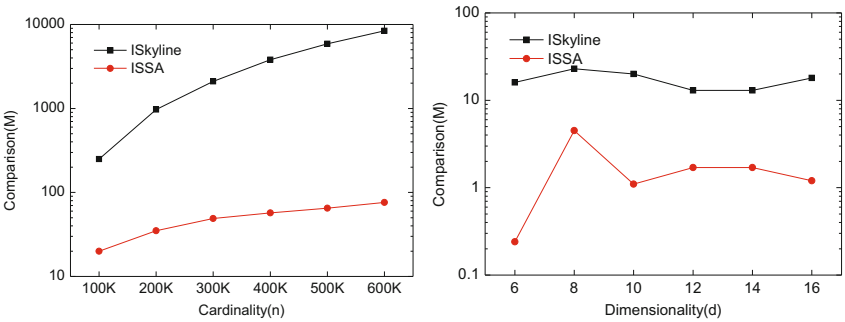
Theorem 1. *Bucket technique can maximize to delete these points which are eliminated that have no effect on the result of skyline computation.*

Proof. Suppose that there is one point p which can be discarded safely and it does not belong to these points that are eliminated by bucket technique. It is obvious to infer that there must exist a point q which satisfy following conditions if p can be safely removed: q is not worse than p in every dimension and q and p have the same representation. Then q dominates p and they are in the identical bucket. So point p must be pruned by bucket technique. This contradicts with origin assumptions. □

Bucket technique prunes as many safe points as possible. Then, use the remaining points to initialize CL and CW as shown in Algorithm 1 in line 1–2. Although the cost is still extremely expensive, there is no better way since the dominance relation is *non-transitive* in incomplete data.

4.3 Phase II: Reducing Expected Comparison Times

Only adopting phase I makes it still inefficient, which is mainly due to the point in candidate list stopping being checked until find its dominating point in compared list. So we should strive to find out the dominating point of the being checked candidate point as soon as possible. As the skyline points, they must be compared with all the points in compare list. While for the other points, the ideal way is to only need one comparison, *i.e.*, compared with one of their dominating points. Unfortunately, if we know which points are their dominating points, the skyline result will be given immediately.



(a) varying cardinality (b) varying dimensionality

Fig. 2. Performance on synthetic data set

Now, the most important task is to pay low cost to construct a compared list order, in which all points can quickly find their dominating points. So as to achieve the goal of reducing expected comparison times. Under UI condition [3], especially in incomplete data set, for those points with the same size of *complete* dimensions, we observe that the point with lower sum have greater probability of being not dominated by any point than that with higher sum. To extend it to general incomplete situation, we sort all points in compared list according to their average value in all *complete* dimensions (ACD).

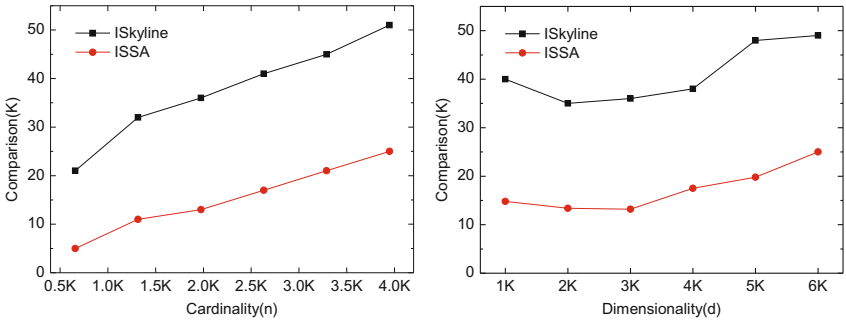
For instance, $p1 = (2, 7, –)$, $p2 = (4, –, 3)$ and $p3 = (–, 3, 2)$ in compared list. The order is $p1$, $p2$, $p3$ before sorting. Now we sort the list according to ACD, and result in the sorted list $p3$, $p2$, $p1$. We call the list sorted compared list. Then, all the points in candidate window are checked by the order of sorted compared list.

5 Experimental Evaluation

In this section, we evaluate our algorithm by experiments. We compare our algorithm ISSA with state-of-the-art approach ISkyline [6] in dimensionality and cardinality in synthetic and real data sets. All algorithms are performed on a Microsoft Windows 7 computer with an Intel Core i7-4790 CPU at 3.6 GHz and 8 GB memory.

5.1 The Performance on Synthetic Data Sets

This section describes the performance of our algorithm ISSA and state-of-the-art approach ISkyline [6] on synthetic data sets. We generate several incomplete synthetic data sets with dimensions and incompleteness ratio ranging from 6 to 16 and 25 % to 50 %, respectively. First, we produce the complete data set under the UI condition in [3], and the domain of complete dimensions is [0,1000]. Then, randomly remove some dimensions to make them incompleteness according to



(a) varying cardinality (b) varying dimensionality

Fig. 3. Performance on real data set

identical ratio for every data point. These synthetic incomplete data sets are generated as above methods.

Figure 2(a) shows the total comparisons of them over cardinality variation from 100K to 600K, where the dimensionality is 12 all the time and the incompleteness ratio is 25 %. Apparently, ISSA outperforms ISkyline 1 to 2 orders of magnitude in comparisons. Figure 2(b) describes the performance of algorithms by varying dimensionality from 6 to 16. The cardinality and incompleteness ratio of the data sets are all 200K and 50 %, respectively. As Fig. 2(b) reports, ISSA is always faster than ISkyline.

5.2 The Performance on Real Data Sets

This section describes the performance of ISSA and ISkyline on real data set MovieLens [5]. It is one of data sets in [5] and contains 1 million ratings. Actually, 95.75 % is incomplete for the data set which includes 3952 points(movies) and each of them has 6040 dimensions (users). We conduct experiments by varying cardinality and dimensionality. In Fig. 3(a), the cardinality is ranging from 658 to 3952 where the dimension is 6040. In Fig. 3(b), the dimension is ranging from 1 K to 6 K where the cardinality is 3952. It is clear that ISSA is always several times faster than ISkyline as shown in Fig. 3.

6 Conclusions

In this paper, we presented a general framework COBO for skyline computation over incomplete data. And based on COBO, we developed an efficient algorithm ISSA in two phases: *pruning compared list* and *reducing expected comparison times*. *Pruning compared list* could delete safely these points which are eliminated that have no effect on the result of skyline computation. And *reducing expected comparison times* constructs a compared list order according to ACD to strive to diminish the total expected comparisons among points in different

buckets. Experimental results on synthetic and real data sets indicated that our algorithm outperforms existing state-of-the-art algorithm 1 to 2 orders of magnitude in comparisons.

References

1. Börzsönyi, S., Kossmann, D., Stocker, K.: The skyline operator. In: Proceedings of the 17th International Conference on Data Engineering, Heidelberg, Germany, pp. 421–430, 2–6 April 2001
2. Chan, C.Y., Jagadish, H.V., Tan, K., Tung, A.K.H., Zhang, Z.: Finding k-dominant skylines in high dimensional space. In: Proceedings of the ACM SIGMOD International Conference on Management of Data, Chicago, Illinois, USA, pp. 503–514, 27–29 June 2006
3. Chomicki, J., Godfrey, P., Gryz, J., Liang, D.: Skyline with presorting. In: Proceedings of the 19th International Conference on Data Engineering, Bangalore, India, pp. 717–719, 5–8 March 2003
4. Dellis, E., Seeger, B.: Efficient computation of reverse skyline queries. In: Proceedings of the 33rd International Conference on Very Large Data Bases, University of Vienna, Austria, pp. 291–302, 23–27 September 2007
5. http://movielens.umn.edu
6. Khalefa, M.E., Mokbel, M.F., Levandoski, J.J.: Skyline query processing for incomplete data. In: Proceedings of the 24th International Conference on Data Engineering, ICDE 2008, Cancún, México, pp. 556–565, 7–12 April 2008
7. Pei, J., Jiang, B., Lin, X., Yuan, Y.: Probabilistic skylines on uncertain data. In: Proceedings of the 33rd International Conference on Very Large Data Bases, University of Vienna, Austria, pp. 15–26, 23–27 September 2007
8. Yuan, Y., Lin, X., Liu, Q., Wang, W., Yu, J.X., Zhang, Q.: Efficient computation of the skyline cube. In: Proceedings of the 31st International Conference on Very Large Data Bases, Trondheim, Norway, pp. 241–252, 30 August–2 September 2005
9. Zhang, S., Mamoulis, N., Cheung, D.W.: Scalable skyline computation using object-based space partitioning. In: Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD, Providence, Rhode Island, USA, pp. 483–494, 29 June–2 July 2009