Skyline Queries and Pareto Optimality

Peng Peng and Raymond Chi-Wing Wong
Department of Computer Science and Engineering
Hong Kong University of Science and Technology
Clear Water Bay, Hong Kong
{ppeng, raywong}@cse.ust.hk

SYNONYMS

PARETO-OPTIMAL TUPLES

DEFINITION

Given two d-dimensional points p and q where d is a positive integer, p is said to *dominate* or *pareto-dominate* q, denoted by $p \prec q$, if p is better than or equal to q on all dimensions and p is better than q on at least one of the d dimensions. Given a set D of d-dimensional points and a point p in D, p is said to be a *skyline point* in D if p is not dominated by any other points in D. A *skyline query* is to find all skyline points in D.

Each dimension can be numeric or categorical. If a dimension is numeric, all values in this dimension are totally-ordered. For any two values in the dimension, one value is more preferable than the other value. One example of a numeric dimension is the price of a product where a smaller value is more preferable. Another example of a numeric dimension is the hotel class where a higher value is more preferable. If a dimension is categorical, the ordering on the values in this dimension is more complicated. One example is airline. Some users prefer one airline A to another airline B but do not have any inclination to prefer one airline to another airline (or vice versa). Besides, some other users prefer airline B to airline A but still do not have any inclination to prefer one airline to another airline (or vice versa). No matter whether each dimension is numeric or categorical, based on the preferences on all values in the dimension, the skyline query can determine all skyline points.

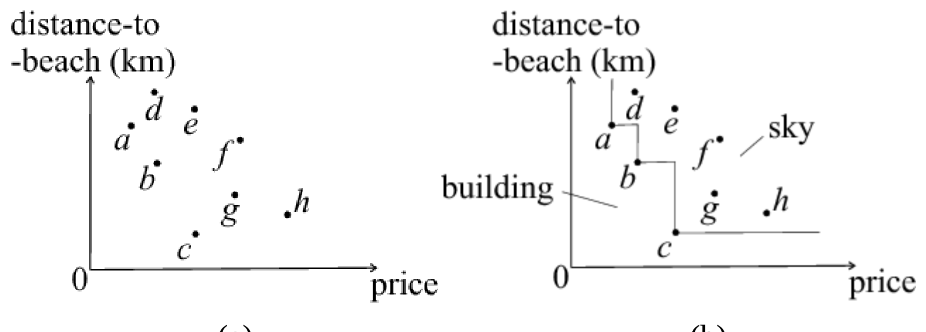

Figure 1. A running example

For example, consider that a user would like to find a hotel near to a beach by considering two criteria, namely the distance to a beach and the price. Figure 1(a) shows 8 hotels, namely *a, b, ..., h*. In this figure, hotel *a* dominates hotel *d* because hotel *a* is closer to a beach than hotel *d* and it is cheaper than hotel d. But, hotel *a* does not dominate hotel *f* because hotel *f* is closer to a beach than hotel *a*. Besides, hotel *a* is not dominated by any hotels in this example. Hotel *a* is a skyline point. In this example, the set of all skyline points is {*a*, *b*, *c*}.

HISTORICAL BACKGROUND

The skyline query was studied for traditional processing system in 1975 [1]. The term "skyline points" found in the skyline query is originally named as the term "pareto-optimal" commonly used in Economics and Engineering due to the Italian economist *Vilfredo Pareto*. Since Borzsony et. al. [2] revisited it in 2001 and named the term as "skyline points", the skyline query has been studied comprehensively in the literature of databases and the main focus is to study how to compute skyline points efficiently on very large relational databases [3, 4].The reason why the term is named as "skyline points" is that in a 2-dimensional space where smaller values are more preferable in each dimension, the visualization of the points together with the skyline points (or pareto-optimal) resembles to the real-life skyline view in a city. To illustrate, consider the same running example. Figure 1(b) is the same as Figure 1(a) but it includes two lines for each skyline point (i.e., *a*, *b* and *c*) which can separate the space into two partitions. One partition is the one near to the origin and the other partition is the one containing all non-skyline points. The former partition can be regarded as a number of buildings with different heights in the city and the latter partition can be regarded as a sky containing some stars each representing a non-skyline point. The lines introduced in the figure can be regarded as a skyline in the city.

Skyline is one of the most useful queries in multi-criteria decision analysis. It does not require users to give any utility function which a well-known database query called *top-k query* needs. It is shown that for any linear utility function, the point with the greatest utility value based on this function can be found in the set of skyline points. Thus, skyline points can be regarded as candidates of the traditional top-1 query. However, since there is no need to specify any utility function in a skyline query, there can be a vast number of skyline points in a dataset.

SCIENTIFIC FUNDAMENTALS

Most existing studies about skyline queries are to find an efficient way of computing skyline points.

The earlier studies about skyline queries focused on skyline queries on datasets with numeric attributes only. The following studies additionally took categorical attributes into consideration for skyline queries.

First, we consider the datasets with numeric attributes only. In the following, we assume that a smaller value is preferable in each numeric attribute.

A naive method of computing skyline points for these datasets is the *block nested loop (BNL)* algorithm which is to compare each point in the dataset with all other points in the dataset to see whether it is dominated by these other points and return all points which are not dominated by other points. However, this method is not efficient since it needs to re-read the dataset multiple times in order to find all skyline points.

One improved skyline algorithm [1] was based on a divide-and-conquer strategy. The idea is to firstly divide the entire set of points into a number of partitions, compute all skyline points based on each separate partition, and then find all skyline points based on all partitions by filtering out all points found in each separate partition which are dominated by other points found in other partitions. Although this method can speed up to find skyline points, it is not *progressive*. A method is said to be progressive if the method returns a subset of the answer before the algorithm ends and keeps updating the answer to the final answer until the algorithm terminates.

Another improved skyline algorithm is the sort first skyline (SFS) algorithm [3] which is progressive. In this algorithm, each point in the dataset is assigned a score which is computed based on the monotonic function on its dimensional values. This algorithm makes use of an interesting property that if a point p dominates another point p', the score of p must be smaller than that of p'. Based on this property, an efficient SFS algorithm was designed. Initially, the algorithm initializes a variable S, which stores the current skyline set, to an empty set. Then, it sorts all points in ascending order of their scores. Next, it iteratively processes each point in this ordering by comparing it with all points in S to check whether it is dominated by any point in S. If no, it is inserted into S. The final set S corresponds to the final skyline point set.

There are many other methods for skyline queries like Bitmap [5], Index [5], Nearest Neighbor [6] and Bound-and-Bound (BBS) [4]. The most influential one is the BBS method which is progressive and IO-optimal. In this algorithm, all points in the dataset are indexed by an R*-tree. Besides, in this algorithm, a min-heap is maintained to keep a list of nodes in the R*-tree where the key of each node is the minimum distance between a reference point (usually, the origin) and this node. Since each node in an R*-tree could be represented by a minimum bounding rectangle (MBR), the algorithm can make use of the geometry property of this MBR for pruning some unnecessary nodes during the execution of the algorithm, resulting in an efficient algorithm.

Next, we consider the datasets including categorial attributes in addition to numeric attributes. Some existing studies [7] focused on finding skyline points when values in each categorical attribute follow a partial ordering for any user. Some other studies [8] focused on finding skyline points when values in each categorical attribute follow different partial orderings for different users.

Some existing studies [9, 10], developed based on skyline queries, studied how to reduce the output size of a skyline query. One kind of queries is called a *k-representative skyline* query [9, 10]. A k-representative skyline query [9, 10] outputs a set of *k* "representative" points from a set of the skyline points. Different papers have different definitions about the concept of "representative" points. [9] proposed to find *k* points which dominate the greatest number of points as "representative" points. [10] proposed to find *k* points such that each of these points is far away from the other points.

KEY APPLICATIONS

Skyline queries are usually used for multi-criteria decision analysis where the utility function of a user is unknown. A famous example of skyline queries in the literature of databases is the hotel booking application which is shown in our running example that a traveler plans to book a hotel near to a beach for vacation. All hotels shown in the answer of the skyline query are all possible candidates for a user with any utility function.

Recently, some research studies focus on how to utilize skyline points for decision-making to serve the end purpose of business people. One example is how to find competitive products [11] and profitable products [12] among all possible products in the pool for decision-making.

FUTURE DIRECTIONS*

One future direction of the skyline queries is based on some variants of the concept of "dominance", which can help users formulate their preferences in a richer manner. One representative example is *subspace dominance* [4].

Another future direction of skyline queries is to study different types of representative skylines, attempting to reduce the size of output points by utilizing more information which could be easily obtained from users.

The other future direction of skyline queries is to study the skyline queries on different types of databases such as dynamic databases [4], distributed databases [13] and uncertain databases [14]. More discussions on the future work of the pareto optimal and skyline queries can be found in [15].

DATA SETS*

nba dataset: http://www.basketballreference.com color dataset:
http://kdd.ics.uci.edu
household dataset: http://www.ipums.org

CROSS REFERENCES

TOP-K QUERIES
UNCERTAIN DATABASE
DYNAMIC DATABASE

References

[1] H.-T. Kung, F. Luccio, and F. P. Preparata, “On finding the maxima of a set of vectors,” *J. acm*, vol. 22, no. 4, pp. 469–476, 1975.

[2] S. Borzsony, D. Kossmann, and K. Stocker, “The skyline operator,” in *Data Engineering, 2001. Proceedings. 17th International Conference on. IEEE, 2001*, pp. 421–430.

[3] J. Chomicki, P. Godfrey, J. Gryz, and D. Liang, “Skyline with pre-sorting,” in *ICDE*, vol. 3, 2003, pp. 717–719.

[4] D. Papadias, Y. Tao, G. Fu, and B. Seeger, “An optimal and progressive algorithm for skyline queries,” in *Proceedings of the 2003 ACM SIGMOD international conference on Management of data. ACM, 2003*, pp. 467–478.

[5] K. Tan, P. Eng, and B. Ooi, “Efficient progressive skyline computation,” in *VLDB*, 2001.

[6] D. Kossmann, F. Ramsak, and S. Rost, “Shooting stars in the sky: An online algorithm for skyline queries,” in *VLDB*, 2002.

[7] C.-Y. Chan, P.-K. Eng, and K.-L. Tan, “Stratified computation of skylines with partially-ordered domains,” in *SIGMOD*, 2005.

[8] R. C.-W. Wong, J. Pei, A. W.-C. Fu, and K. Wang, “Mining favorable facets,” in *SIGKDD*, 2007.

[9] X. Lin, Y. Yuan, Q. Zhang, and Y. Zhang, “Selecting stars: The k most representative skyline operator,” in *Data Engineering, 2007. ICDE 2007. IEEE 23rd International Conference on*. IEEE, 2007, pp. 86–95.

[10] Y. Tao, L. Ding, X. Lin, and J. Pei, "Distance-based representative skyline," in *Data Engineering, 2009. ICDE'09. IEEE 25th Interna-tional Conference on*. IEEE, 2009, pp. 892–903.

[11] Q. Wan, R. C.-W. Wong, I. F. Ilyas, M. T. Ozsu, and Y. Peng, "Creating competitive products," in *the 35th International Conference on Very Large Data Bases (VLDB'09)*, 2009.

[12] Q. Wan, R. C.-W. Wong, and Y. Peng, "Finding top-k profitable products," in *the 27th International Conference on Data Engineering (ICDE'11)*, 2011.

[13] W.-T. Balke, U. Güntzer, and J. X. Zheng, "Efficient distributed skylining for web information systems," in *Advances in Database Technology-EDBT 2004*. Springer, 2004, pp. 256–273.

[14] I. Bartolini, P. Ciaccia, and M. Patella, "The skyline of a probabilistic relation," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 25, no. 7, pp. 1656–1669, 2013.

[15] J. Chomicki, P. Ciaccia, and N. Meneghetti, "Skyline queries, front and back," *ACM SIGMOD Record*, vol. 42, no. 3, pp. 6–18, 2013.