Efficient Computation of Skyline Query with Keyword-based User Preference

Deokmin Haam*, Myoung Ho Kim**
*,** Korea Advanced Institute of Science and Technology, Korea
E-mail: dmhaam@dbserver.kaist.ac.kr*, mhkim@dbserver.kaist.ac.kr**

1. Introduction

Given a d-dimensional dataset, skyline query finds a set of tuples that are not dominated by any other tuples in the dataset. A tuple *t* is said to dominate another tuple *t'* when *t* is better than or equal to *t'* on all dimensions and *t* is strictly better than *t'* on at least one dimension. Because skyline query finds all the tuples deserve consideration (i.e., tuples not dominated), it is useful in many applications such as multi-criteria decision making. Thus, the skyline computation and its variants have received attention in conventional databases [1][2][3][4].

Since skyline query retrieves generally good tuples, all users get the same results by the query. To provide personalized skyline query results, some works let users describe additional condition of the result. [2] computes skyline based on user preference on data dimensions. For instance, when a user buys a car, she may think "price" dimension is more important than "mileage" dimension. [3] computes skyline based on user preference on nominal attributes such as color. For instance, when a user buys a car, she may prefer "white" cars to "black" cars. [4] uses keywords as user preference of skyline queries. They assume that each tuple is additionally described by a textual attribute, and they find skyline tuples whose textual descriptions contain all query keywords given by the user. For instance, when a user buys a car, she may prefer cars with both lower mileage and lower price, and she considers only cars equipped with both "air bags" and "air conditioning".

In this paper, we propose a skyline query which uses query keywords as user preference. Query-by-Keywords is one of the most useful ways to describe users' needs, proven by the success of keyword search on the Web. We assume that each tuple in a dataset has *d* numeric type attributes and an additional textual information, and a user expresses her preference by using two sets of query keywords, K_N and K_O: she wants only the tuples including all $k_i \in K_N$, and she prefers tuples including more $k_j \in K_O$. This query is suitable to cases as follows: A user wants to buy a used car. She thinks "air conditioning", "sunroof", "heated seats", "navigation system", and "cruise system" are useful in the convenience features. She thinks that "air conditioning" is indispensable, but other features are not. She may prefer cars with both lower mileage and lower price, and also prefer cars having more of {"sunroof", "heated seats", "navigation system", "cruise system"}.

Intuitively, the skyline with K_N and K_O, called *skyline query with keyword-based user preference*, can be computed by a straightforward approach, which consists of three steps: i) retrieving tuples including all $k_i \in K_N$, ii) converting each retrieved tuple into *d+1*-dimensional tuple by combining its *d* numerical attributes and the number of $k_j \in K_O$ included in its textual information, and iii) computing skyline tuples among those *d+1*-dimensional tuples. Although this straightforward approach can find correct skyline tuples, this cannot utilize any index structure because the values in the $d+1^{th}$ dimension of the tuples are calculated when a query is issued, i.e., index structures considering the dimension cannot be constructed in advance.

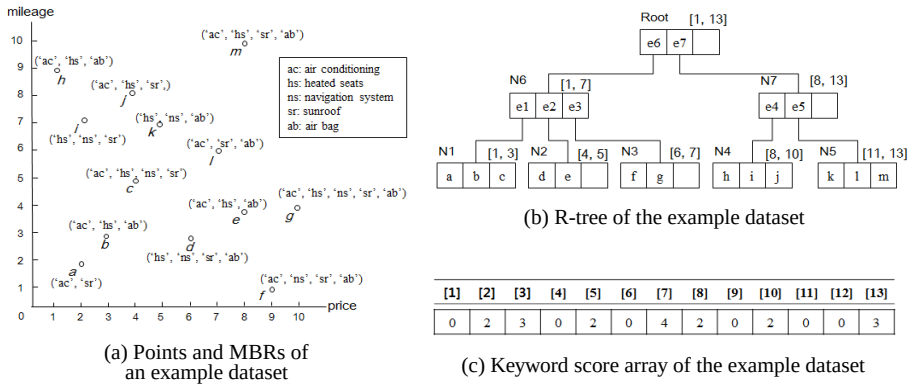
To compute the skyline query efficiently, we propose an algorithm, called *Keyword Preference Skyline computation algorithm* (KPS), utilizing R-tree based index structure used in [1] and bitmap indexes. The *d*-dimensional tuples are indexed by R-tree and their textual information fields are indexed by bitmap indexes. While traversing the R-tree according to a branch-and-bound approach (BBS) [1], our algorithm refers bitmap indexes to consider the $d+1^{th}$ dimension of the tuples. By combining those two index structures, our algorithm computes the skyline query with keyword-based user preference efficiently.

2. Keyword Preference Skyline Computation Algorithm

Since tuples in a dataset have *d* numeric type attributes, we can treat them as points in a *d*-dimensional vector space. Thus, we can index them by using a multidimensional index structure such as R-tree. Figure 1 (a) shows an example dataset consisting of used cars. The cars have price and mileage values, and they are represented by points in a 2-dimensional space. Figure 1 (b) is the R-tree index which is constructed over the cars. After the R-tree is constructed, we give a number to each entry in leaf nodes. The number is the position of the entry in the R-tree. For instance, the car 'a' is the first entry in the R-tree and 'm' is the thirteenth entry. Then, we give range information, which forms [*x*, *y*] in Figure 1 (b), to each non-leaf node. This range information means that the MBR (i.e., non-leaf node) contains x^{th} entry, $x+1^{th}$ entry, …, and y^{th} entry of the R-tree. This range information will be used for comparing the $d+1^{th}$ dimension (i.e., keyword scores of tuples based on K_N and K_O) of the tuples. Also, we index textual information of the cars by using bitmap indexes. Words in brackets (e.g., 'ac', 'sr', etc.) in Figure 1 (a) are convenience features equipped with the cars, i.e., textual information of the cars. For each convenience feature, we make a bitmap whose i^{th} bit is 1 if the i^{th} tuple has the feature (i.e., keyword); 0 otherwise.

By using those two index structures, KPS responses skyline query with keyword-based user preference. When a skyline query is issued with K_N and K_O, the bitmaps of $k_i \in K_N$ and $k_j \in K_O$ are retrieved. Then, K_N array (K_N array = $\wedge_{ki \in KN}$ bitmap$_i$) and K_A array (KA array[i] = $\Sigma_{kj \in KO}$ bitmap$_j$[i], for *i=1, 2, .., M*) are made, where bitmap$_i$ is a bitmap of a keyword k_i and *M* is the number of tuples in the dataset. By these two arrays, *keyword score array* (KSA), which contains keyword scores of tuples based on K_N and K_O, is constructed as follows: KSA[i] = K_N array[i] + K_N array[i]

× K_A array[i], for *i=1, 2, .., M*. Figure 1 (c) shows the keyword score array of the used cars in Figure 1 (a), when K_N={'air conditioning', 'heated seats'} and K_O={'navigation system', 'sunroof', 'air bag'}.



(a) Points and MBRs of an example dataset

(b) R-tree of the example dataset

(c) Keyword score array of the example dataset

Figure 1. An example of a dataset and index structures

After constructing KSA, KPS traverses R-tree based on a branch and bound approach [1] to find skyline tuples. It starts from the root node of the R-tree, and it is inserted into a heap that adjusts order of visited nodes. In the heap nodes are sorted according to their *mindist* from the origin of the *d*-dimensional data space, and the nodes closer to the origin are visited earlier. When a node whose range is [*x, y*] is visited, its keyword score is computed by finding the maximum value in KSA[*x*], KSA[*x+1*], …, and KSA[*y*]. This can be computed quickly by sequential scan of KSA. If the keyword score of the node is equal to 0, then the node is directly pruned because no tuple in the node includes all the $k_i \in K_N$. If the node (whose keyword score is larger than 0) is not dominated by any current skyline tuples in those d+1 dimensions, the node is removed from the heap and its child nodes are inserted into the heap. Otherwise, the node is pruned. When the visited node is a tuple, if it is not dominated by any current skyline tuples, it is removed from the heap and inserted into a set of current skyline tuples. This process is repeated until the heap is empty. Given an example dataset in Figure 1 (a), and K_N={'air conditioning', 'heated seats'} and K_O={'navigation system', 'sunroof', 'air bag'}, the skyline tuples are {*b, c, h, g*}. This result is different from i) {a, f, h}, which is a set of skylines when only price and mileage are considered, ii) {*b, h*}, which is a set of skylines when only tuples including all $k_i \in K_N$ are considered, and iii) ∅, which is a set of skylines when only tuples including all $k_i \in K_N \cup K_O$ are considered.

3. Conclusion

In this paper, we proposed the skyline query with keyword-based user preference and an efficient algorithm for computing the skyline query. To the best of our knowledge, this work is the first attempt to consider the number of matched query keywords as a degree of user preference for skyline computation. KPS, the proposed algorithm, retrieves R-tree based on a branch and bound approach based on the R-tree index structure constructed on tuples in a given dataset. While traversing the R-tree, KPS can prune non-skyline nodes based on their coordinates represented by MBR and their keyword scores computed from bitmap indexes of keywords. The performance study through experiments in various parameters and datasets are in progress. By the experiments we can show that the proposed algorithm can compute skyline query with keyword-based user preference efficiently in many different kinds of environments.

4. Acknowledge

This research was supported by the MSIP(Ministry of Science, ICT and Future Planning) of Korea under the ITRC support program(NIPA-2013-H0301-13-4009), and the National Research Foundation of Korea grant funded by the Korea government(MEST) (No. 2012R1A2A2A01046694).

5. References

[1] Papadias, D., Tao, Y., Fu, G., & Seeger, B. (2003, June). "An optimal and progressive algorithm for skyline queries.", In Proceedings of the 2003 ACM SIGMOD international conference on Management of data (pp. 467-478). ACM.
[2] Lee, Jongwuk, Gae-won You, and Seung-won Hwang. "Personalized top-k skyline queries in high-dimensional space.", Information Systems 34.1 (2009): 45-61.
[3] Wong, R. W., Pei, J., Fu, A. C., & Wang, K. (2009). "Online skyline analysis with dynamic preferences on nominal attributes.", Knowledge and Data Engineering, IEEE Transactions on, 21(1), 35-49.
[4] Choi, H., Jung, H., Lee, K. Y., & Chung, Y. D. (2013). "Skyline queries on keyword-matched data.", Information Sciences, 232, 449-463.