Skyline Computation Using Adaptive Filters

Jihyun Kim, Myung Kim
Dept. of Computer Science & Engineering
Ewha Womans University
Seoul, Korea

***Abstract*— Skyline queries can be effectively used for recommendation systems. Recently, there has been a lot of research interest in improving the performance of the skyline computation. Some such efforts are to eliminate non-skyline records in the early stage of the skyline computation. In this paper, we propose an efficient skyline computation scheme that dynamically adjusts adaptive filters during the first scan of the data set in order to maximize the amount of non-skyline record elimination. The performance of the proposed scheme is evaluated by experiments.**

Keywords-skyline computation; filtering; multidimensional data analysis

I. INTRODUCTION

The skyline of a multidimensional data set is defined as follows. Let A be a d dimensional data set, and let $p = (p_1, p_2, \dots, p_d)$ and $q = (q_1, q_2, \dots, q_d)$ be data elements in A, where p_i and q_i, $1 \le i \le d$, are the values of dimension i of p and q, respectively. If $p_i \le q_i$, for all i, and $p_j < q_j$ for at least one j, $1 \le i, j \le d$, then p is said to dominate q [5]. The skyline of A is defined to be the maximal subset whose elements are not dominated by any other elements of A. For example, Fig. 1(a) shows a 2 dimensional data set consisting of 7 elements. Data elements A, C, and F are not dominated by other elements, thus they compose the skyline of the set. Fig. 1(b) shows two regions R_1 and R_2 that are related to C. Element C is included in the skyline, since there is no element in R_1. Note also that all the elements in R_2 are dominated by C.

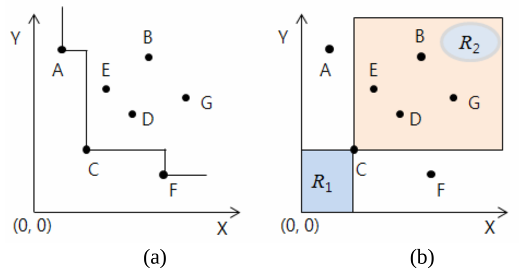

Figure 1. The Skyline of a sample data set. (a) Skyline, (b) Region dominated by a skyline element.

Skyline queries can be applied to many real world problems. For example, let us say that the points in Fig. 1(a) are restaurants. The X axis represents the distance between the user and the restaurants, and the Y axis represents the price of the food served by the restaurants. The skyline of the restaurants consists of the restaurants that are closer to the user as well as cheaper compared to other restaurants.

Skyline computation involves massive comparisons among the elements of the set. Various schemes for reducing the number of comparisons have been reported [1, 2, 3, 4, 5]. Some major schemes are filtering schemes [10, 11, 12], sort based algorithms [9, 10], and an algorithm using stop lines [6]. Filtering schemes are used to eliminate a large amount of non-skyline elements during the first scan of the data set so that other schemes can be applied later to a relatively small amount of remaining data set. Although these filters are replaced by better filters during the data scan, the performance of the filters can still be improved. In this paper, we propose an adaptive filtering scheme that dynamically and systematically adjusts the filters during the first scan of the data set. At the end of the first scan of the data set, the fence-like line (or space) constructed by connecting the filters become very close to the final skyline of the data set. The skyline can be computed from the remaining small data set using a previously reported efficient algorithm.

The paper is organized as follows. In Section 2, related works are overviewed. In Section 3, the proposed skyline computation scheme is presented. In Section 4, the performance of the proposed scheme is evaluated by experiments. In Section 5 we conclude our work.

II. RELATED WORKS

Recently, various techniques for computing the skyline of a large multidimensional data set have been proposed. The first skyline computation algorithm is introduced in [1]. As the skyline computation incurs a high computational cost, there have been research efforts to develop space partitioning-based skyline computation schemes. Three space partition-based skyline schemes, i.e., random partition scheme, grid-based partition scheme, dynamic partition scheme have been introduced in [5, 6, 7, 8]. The angle-based skyline computation scheme [7] further enhances these schemes [6]. It divides the data set into N partitions using angular coordinates. The angular coordinates relies on the hyper-spherical coordinates of the d dimensional coordinates. The scheme then computes the skyline for each partition. The results are then merged to finalize the global result set. The angle-based skyline computation scheme is especially used for parallel and distributed computing environment so that the load balancing is also an important issue to resolve.

Filtering techniques [3, 9, 10, 11, 12] are introduced to skyline computations in order to reduce the number of data comparisons. Filters are used [9, 10] to eliminate a large amount of data elements in the early stage of the skyline computation algorithms. LESS [3] uses filter windows in main memory, and eliminates many data items during the first scan of the initial data set. Although several filters are used, they are not guaranteed to cover a large range of areas in the data space. Filter lining scheme [12] places filters to cover a wide range of data space. Filters are places in angle-based partitioned areas one for each. They look like a fence close to the origin of the data space. However, the initial placement of their filters is not the best possible that can be obtained from the sample data set. Dynamic adaptation of the filter fence is not maximized either. In this paper, we enhance the filter lining scheme proposed in [12].

III. Skyline Computation using Adaptive Filters

Our skyline computation algorithm uses adaptive filters. The fundamental ideas for the filters will be given here, and the entire skyline computation algorithm will be given later. For convenience, we use a 2-dimensional data set as an example. Let us first take a look at the sample data set given in Fig. 1(b). Suppose that we randomly select a data element from the data set, and assume that the selected data element is *C*. Suppose also that we use *C* as the filter, and scan the entire data set. During the first scan of the data set, all the elements except elements *A* and *F* will be discarded. Thus, *C* can be considered to be a reasonable filter. Now instead of having *C* as the filter, suppose that *B* be used as the filter. In this case, no elements will be discarded in the data scan. Thus, we can see that filters should be placed close to the origin of the data space, and be widely spread over the data space.

In order to cover a wide range of the data space, we divide the 2-dimensional data space into equally spaced partitions as in Fig. 2(a). In the figure, the data space is divided into 4 partitions, P_1, P_2, P_3, and P_4. If we can place good quality filters to each partition, large amount of non-skyline elements will be discarded in the first scan of the data set. In the figure, F_i is such filter for partition P_i, $1 \le i \le 4$. Let us now consider how such filters can be selected from the data set. One way to do so is to randomly choose a predetermined amount of samples, and place them to each partition, and then determine one for each partition which is closer to the origin of the data space than others placed in the same partition.

Next, let us consider the situation depicted in Fig. 2(b). F_3 is dominated by F_2. In this case, although F_2 does not belong to partition P_3, F_2 can be a better filter for P_3. The filtering scheme should be intelligent enough not to miss such a chance. Let us now take a look at the situation shown in Fig. 2(c). There are two candidate filters, $F_{2,1}$ and $F_{2,2}$ for partition P_2. The distance from the origin of the data space and the two elements is the same. In this case, $F_{2,1}$ should be chosen as the filter for the partition because the region dominated by $F_{2,1}$ in the partition is larger than that by $F_{2,2}$. The proposed filtering scheme takes into consideration of such facts for the initial placement of the filters in each partition.

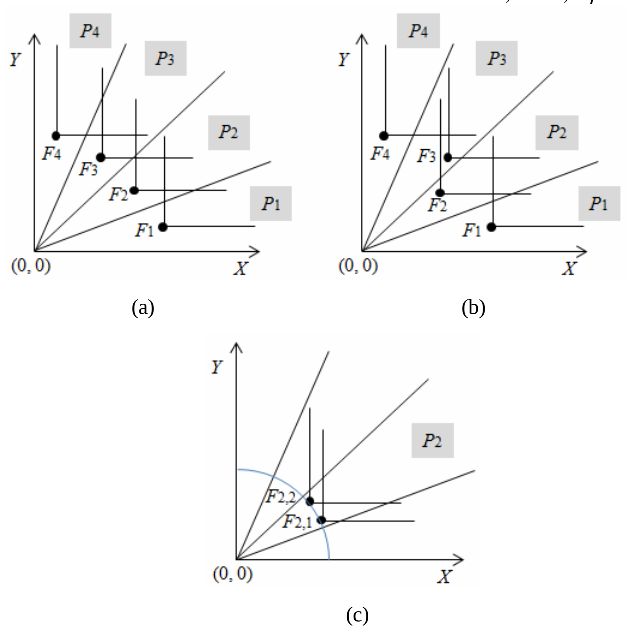

Figure 2. Adaptive filters. (a) Filters for the partitions, (b) Relationships among filters, (c) Regions covered by filters.

A. The initial placement of filters

Initially, the proposed algorithm places filters as follows. First, a predetermined amount of data elements is selected randomly from the input data set, and let the set be *S*. *S* is then sorted in increasing order of the distance between the origin and the data elements. The skyline of *S* is computed next. Let *SS* be the skyline of *S*. Let *m* be the number of partitions, and F_1, F_2, … , F_m be the filters for partitions P_1, P_2, . . . , P_m, respectively. All such filters are initialized with the first element of *SS* which is closest to the origin of the data space. The elements *s* in *SS* are then compared in order with each F_i. If *s* is determined to be a better filter for P_i as in Fig. 2(b) and Fig. 2(c), F_i is updated with the new value, which is *s*. At the end of the filter update stage, the initial filter for each partition will be guaranteed to be the best filter among the sample set *S*.

B. Filter updates in the data scanning stage

The next step of the skyline computation is to scan the data set. For each data element read, *p*, its partition number is calculated. Let us say that *p* belongs to partition P_i. Element *p* is then compared with F_{i-1}, F_i, and F_{i+1}. If *p* is dominated by one of these filters, it is eliminated. On the other hand, if *p* dominates any such filters, the filters are replaced with *p*. Fig. 2(a) shows a case when three adjacent filters work better than only one filter for the partition. Note that filters are updated by some elements in the corresponding partition as well as some elements that belong to the adjacent partitions. This way, filters are upgraded faster. However, there might be two filters, F_i and F_j, $i \neq j$, that do not belong to adjacent partitions, but one dominates the other. In order to get rid of such bad filters, all the filters are compared periodically. This way, at the near end of the data scanning stage, all the filters approach the final skyline of the data set. Note that after finishing the data scanning, all the filters belong to the skyline.

C. Skyline Computation Algorithm

We now present our skyline computation algorithm in detail for a d dimensional data set. Let us assume that each dimension is divided into m partitions, thus there are m^{d-1} partitions in the entire data space. Partitions are indexed as $P(i_1, i_2, \dots, i_{d-1})$ and their corresponding filters are indexed as $F(i_1, i_2, \dots, i_{d-1})$. The size of the sample set, S, is n_S. The algorithm is formally described as follows.

Skyline Computation Algorithm

[Step 1] [*Data Sampling*]
Choose n_S random samples from the given data set, and let S be the sample data set.

[Step 2] [*Skyline Computation for S*]
For each sample s, compute its distance from the origin of the data space, and let it be G_s. Sort S in increasing order of G_s . Compute the skyline for the sample data set, S, and let it be SS.

[Step 3] [*Initial placement of filters*]
Let SS_0 be the first element of SS. Initialize all the filters, $F(i_1, i_2, \dots, i_{d-1})$, with SS_0. For each element s in SS, it is compared with each filter $F(i_1, i_2, \dots, i_{d-1})$. If s is better than the current filter for the corresponding partition, the current filter is replaced with the value of s. At the end of step 3, the values of all the filters are the best possible filters for the corresponding partitions that can ever be selected from the sample set S.

[Step 4] [*Data scan and local filter upgrade*]
Scan the data set. For each data item, p, compute its partition number, and let it be $P(i_1, i_2, \dots, i_{d-1})$. Compare p with $F(i_1, i_2, \dots, i_{d-1})$ and neighboring filters for each dimension. If p is dominated by any of the $2(d-1) + 1$ filters, it is eliminated. On the other hand, it there is any filter among the neighboring filters that is dominated by p, then p becomes a new filter for the corresponding partition. Do the above work for all the elements of the input data set.

[Step 5] [*Global upgrade of the filters*]
After predetermined number of data scans, all the filters are compared as in [Step 3], in order to adjust the filters globally. Data scanning is then resumed.

[Step 6] [*Skyline Computation for each partition*]
Sort the remaining elements for each partition, and then compute the skyline for each partition independently.

[Step 7] [*Skyline Computation for the entire data set*]
Merge the skylines for each partition, and compute the skyline for the entire data set.

IV. PERFORMANCE EVALUATION

We conducted some experiments on a PC equipped with 2.40GHz Intel *i*5 CPU and 4.0 GB main memory. The programs are written in C, and are run in .NET environment. The data sets with various sizes and distributions are artificially synthesized and generated. We use Gaussian distribution, Correlated distribution, and Anti-correlated distribution with sizes of 1M, 5M, and 10M data elements, where M represents million. Data sets are initially stored in main memory.

Table 1 shows the time taken by the proposed skyline computation algorithm. The data sets used for experiments are 2 dimensional data sets, and the number of partitions in the data space is 9, meaning that partitions are placed in every 10°. The size of the sample set is 10,000. The time taken by filtering is 1.00 sec for 10M, 0.46 sec for 5M, 0.10 sec for 1M data sets, respectively. Thus, we can see that the time taken by computing the skyline with the remaining data is quite negligible for the data sets with Gaussian distribution or Correlated distribution. However for the data sets with Anti-correlated data distribution, the execution time is nearly double the time taken by the data sets of the other two distributions. It is because almost 14% of the data set remains after the filtering, as in column 4, Table II. Overall, the skylines are computes very efficiently.

TABLE I. TIME TAKEN BY THE SKYLINE COMPUTATION ALGORITHM.

Type / Size	Data Distributions		
	Gaussian D	*Correlated D*	*Anti-Correlated D*
10M	1.05sec	1.10sec	2.37sec
5M	0.77sec	0.46sec	1.15sec
1M	0.11sec	0.10sec	0.20sec

Table 2 shows the percentage of the remaining data after the filtering is finished. The rate does not depend on the sizes of the data sets, but depends on the data distributions only. For the data sets with Gaussian distribution, nearly 96% of the data set is eliminated by the adaptive filters. For the correlated data sets, almost all the data elements are eliminated at the filtering stage. However, 14% of the data set remains for the Anti-correlated data sets. This is why for such data sets, the entire time taken by the algorithm doubles that for the other two data distributions as in the 4th column of Table 1.

We conducted experiments with 3 dimensional data sets. The test results are shown in Table 3.The objective of the experiments is to check to see how much overhead is incurred for high dimensional data sets. The number of partitions varies from 7 x 7 to 9 x 9. The size of the data set is 10M, and the data distribution is Gaussian. As in [step 4] of the skyline computation algorithm, each data element is compared with at most $2(d-1) + 1$ filters. As explained in [12], partition number calculation for each element is done in $O(1)$ time. Thus, time taken by the case with 9 x 9 partitions can be smaller than the cases with smaller number of partitions. We can also see that the filtering effect is better with more partitions. Overall, the proposed skyline computation algorithm is very efficient.

TABLE II. THE EFFECT OF THE DATA ELIMINATION.

Type / Size	Data Distributions		
	Gaussian D	*Correlated D*	*Anti-Correlated D*
10M	3.75%	0.20%	14.0%
5M	3.77%	0.20%	13.8%
1M	3.73%	0.20%	14.0%

TABLE III. FILTERING EFFECT AND OVERHEAD OF A 3 DIMENSIONAL DATA SET.

Number of partitions	Rate of the remaining data	Time	Size of the skyline (count)
9 x 9	7.12%	2.83sec	18,163
8 x 8	7.85%	2.96sec	18,179
7 x 7	8.21%	2.95sec	17,513

V. CONCLUSIONS

We presented a skyline computation scheme with adaptive filters, and evaluated the performance by experiments. The initial placement of the filters to each partition is the best possible from the sample data set. During the first data scanning phase, each data element is compared with the current filter of its own partition as well as the filters for the neighboring partitions. This way, data elements are eliminated as many as possible, as well as, filters have more chances to get updated. The cost for computing the partition number of a data element is $O(1)$. Thus the time taken by computing the partition number does not depend on the dimensionality of the data set. However, at most $2(d-1) + 1$ data comparisons will be needed for d dimensional data sets. Overall, the proposed algorithm works very well.

REFERENCES

[1] S. Borzsonyi, D. Kossmann, and K. Stocker, “The skyline operator,” In *ICDE* 2001, pp. 421–430, Heidelberg, Germany, Apr. 2001.

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

[3] K.-L. Tan, P.-K. Eng, and B. C. Ooi, “Efficient progressive skyline computation,” In *VLDB* 2001, pp. 301–310, Roma, Italy, Sep. 2001.

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

[5] N. Z. Adan Cosgaya-Lozano, Andrew Rau-Chaplin. “Parallel computation of skyline queries,” In *International Symposium on High Performance Computing Systems and Applications(HPCS),* 2007.

[6] P. Wu, C. Zhang, and Y. Feng, “Parallelizing skyline queries for scalable distribution,” In *EDBT*, pp.112–130, 2005.

[7] A. Vlachou, C. Doulkeridis , Y. Kotidis, “Angle-based space partitioning for efficient parallel skyline computation,” In *ACM SIGMOD* 2008, pp. 227–238, Vancouver, Canada, Jun. 2008.

[8] Un-gyu Baek, Sukhyun Ahn, Seung-won Hwang, “Dynamic partitioning for parallel skyline computation,” In *EDB*, 2009.

[9] J. Chomicki, P. Godfrey, J. Gryz, D. Liang, “Skyline with presorting,” In *ICDE* 2003, pp. 717–719, India, Mar. 2003.

[10] I. Bartolini, P. Ciaccia, M. Patella, “SaLSa: Computing the skyline without scanning the whole sky,” In *CIKM* 2006, pp. 405-414, Arlington, Virginia, USA, Nov. 2006.

[11] P. Godfrey, R. Shipley, and J. Gryz, “Maximal vector computation in large data sets,” In *VLDB* 2005, pp. 229–240, Trondheim, Norway, Aug. 2005.

[12] Jihyun Kim, Myung Kim, “A filter lining scheme for efficient skyline computation,” *Journal of Korea Multimedia Society*, vol. 14, no. 2, pp.1581–1600, Dec. 2011.

AUTHORS PROFILE

Jihyun Kim received her BS degree in Computer Science at Sangmyung University in 1995, and MS degree in Computer Science and Engineering at Ewha Womans University in 2007. She is currently a PhD candidate in Dept. of Computer Science and Engineering at Ewha Womans University. Her research interests include Moving object databases, Skyline Computations, Data analysis, Stream data analysis, Ontology, and Information Retrieval.

Myung Kim received her BS degree in Mathematics, Ewha Womans University in 1981. She received her MS degree in Computer Science at University of Minnesota, USA, in 1990. She received her PhD in Computer Science, University of California, Santa Barbara, 1993. She is currently a faculty member of the Department of Computer Science and Engineering, Ewha Womans University. Her research interests include Data analysis, Real time recommendation Systems, Information Retrieval, and High performance computing.