Journal of Korea Multimedia Society Vol. 14, No. 12, December 2011(pp. 1591-1600)

A Filter Lining Scheme for Efficient Skyline Computation

Jihyun Kim†, Myung Kim††

ABSTRACT

The skyline of a multidimensional data set is the maximal subset whose elements are not dominated by other elements of the set. Skyline computation is considered to be very useful for a decision making system that deals with multidimensional data analyses. Recently, a great deal of interests has been shown to improve the performance of skyline computation algorithms. In order to speedup, the number of comparisons between data elements should be reduced. In this paper, we propose a filter lining scheme to accomplish such objectives. The scheme divides the multidimensional data space into angle-based partitions, and places a filter for each partition, and then connects them together in order to establish the final filter line. The filter line can be used to eliminate data, that are not part of the skyline, from the original data set in the preprocessing stage. The filter line is adaptively improved during the data scanning stage. In addition, skylines are computed for each remaining data partition, and are then merged to form the final skyline. Our scheme is an improvement of the previously reported simple preprocessing scheme using simple filters. The performance of the scheme is shown by experiments.

Key words: skyline computation, decision making system, filtering, multidimensional data processing

1. INTRODUCTION

The skyline of a multidimensional data set is defined as follows. Let A be a d dimensional data set, and let p be a data item in A. Suppose that p_i, $1 \le i \le d$, is the value of the i-th dimension of p, then p can be expressed by $(p_1, p_2, \dots, p_d)$ in the d dimensional data space. Now, consider the dominance relationship between any two points in the set. Let $p=(p_1, p_2, \dots, p_d)$ and $q=(q_1, q_2, \dots, q_d)$ be data items in A. If $p_i \le q_i$, for all i, $1 \le i \le d$, and if $p_j < q_j$, for at least one j, $1 \le j \le d$, then p is said to dominate q [1]. The skyline of A is defined to be the maximal subset whose elements are not dominated by any other elements of A [2]. For example, we are given a 2-dimensional data set {(1, 4), (2, 2), (3, 5), (5, 5), (4, 2), (3, 3), (4, 1), (5, 3)}, as in Figure 1. The skyline of the data set is {(1,4), (2, 2), (4, 1)}, since these three elements are not dominated by any elements of A. That is, {A, B, G} is the skyline of the given data set in the figure.

Skyline computation is one of the most useful operations for extracting from a large data set a small set of data that satisfies the user's requirements. For example, the points in Figure 1

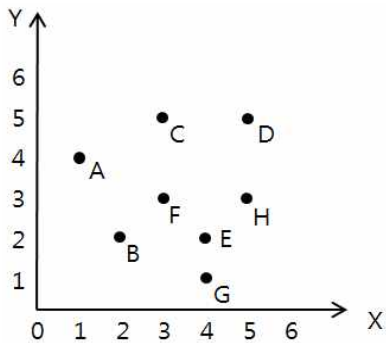


Fig. 1. A 2-dimensional data set and its skyline.

※ Corresponding Author : Myung Kim, Address : (120-750) Department of Computer Science and Engineering Ewha Womans University, 11-1 Daehyun-Dong Seodaemun-Ku, Seoul TEL : +82-2-3277-2594, FAX : +82-2-3277-2306, E-mail : mkim@ewha.ac.kr
Receipt date : Oct. 10, 2011, Revision date : Nov. 11, 2011
Approval date : Dec. 13, 2011
† Department of Computer Science and Engineering Ewha Womans University
(E-mail: jhrosa@ewhain.net)
†† Department of Computer Science and Engineering Ewha Womans University

represent restaurants. The X-axis shows the distance between the user and the corresponding restaurants, and the Y-axis shows the average price of the food served by the restaurants. Now assume that the user wants to find a restaurant that is closer and the food served by the restaurant is relatively cheap. The skyline $\{A, B, G\}$ satisfies such requirements. Thus, if the decision making system provides the user with the skyline of the restaurant set, he/she can then make the final decision using the given skyline [3].

Let us now examine the process of computing the skyline of a multidimensional data set. For convenience, we use the terms 'a multidimensional data set' and 'a point set in a multidimensional space' interchangeably. Note that the skyline of a multidimensional point set consists of the points that are proven not to be dominated by other points in the given set. Thus, skyline computation mainly consists of comparisons between points in order to determine the dominance relationships of the points. Recently, various schemes for reducing the number of comparisons between points are reported [1-6]. Major schemes include filtering schemes [5, 6], sort based algorithms [1-6], and an algorithm using stop lines [5]. Filtering schemes first select a point p that is considered to be close to the origin of the multidimensional space, and set it to be the filter. The input point set is then scanned, and the points that are dominated by the filter are eliminated. The remaining point set is used for extracting the skyline. Sort based algorithms first sort the input point set so that the number of comparisons among the points are reduced. This scheme can also be used after the filtering scheme is applied to the original set. The algorithm using stop lines computes and updates the stop line while reading the sorted input points. During the input scanning stage, once meeting a point that is beyond the stop line, the algorithm halts since there is no skyline point in the remaining input set. Among these schemes, the filtering scheme is one that can be applied to the input set in the preprocessing stage. The time for skyline computation can be greatly reduced if a big portion of the input set is eliminated in this stage.

In this paper, we propose a filter lining scheme that uses an angle-based space partitioning method, and that has most of the advantages of the previously described three schemes. Our filter lining scheme first divides the multidimensional data space by equally spaced angles from the origin of the space. It then takes samples from the original data set, and chooses the best filter for each partition. The filters are then compared to each other, and are replaced by the filter that is expected to eliminate more data points in the partition. Remaining filters form the initial filter line. The algorithm then scans the original data set, and eliminates all the data points that are dominated by the filter line. The filter line is adoptively updated during the input scanning stage.

The contribution of our work includes the following. First, we propose a filter lining scheme that can be used in the preprocessing stage of skyline computation algorithms. The filter line covers a wide range of areas in the multidimensional space, thus filtering effect can be maximized. Furthermore, the filter line moves toward the origin incrementally while scanning the input data. Thus, at the end of the input scanning stage, the filter line approaches the final skyline. Second, the effect of using stop lines appear in our filter lining scheme. Third, due to the angle based partitioning, the skyline points for each partition are very likely to be included in the skyline for the entire data set. The paper is organized as follows. In Section 2, related work is given. In Section 3, the proposed filter lining scheme is presented. In Section 4, performance evaluations by experiments are given. In Section 5 we conclude the work.

2. RELATED WORK

Various schemes for computing the skyline of

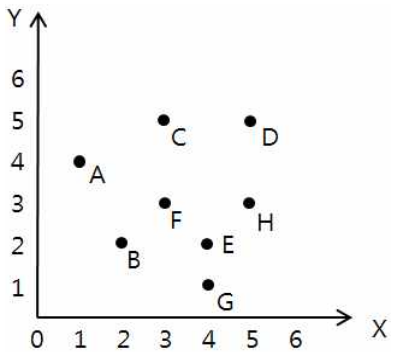
a large multidimensional data set are mainly proposed for the last decade [1-10]. The first reported skyline computation algorithm is given in [2]. It assumes that the multidimensional data set is small enough to be stored in main memory. However, today's application data sets are more likely to be very big, and are more likely to be stored in hard disks. Research results in [1-6] assume such conditions. [4] assumes that the data set is stored in hard disk, and it divides the multidimensional data space into a grid, and uses a divide and conquer algorithm for computing the skyline. It also places windows in main memory and stores in the windows the points that are strong candidates for the final skyline.

The research work [1, 5, 6] present schemes for reducing the number of data comparisons. The work [5, 6] use external sorting on the original data set. In the first scan of the sorting, they use filters to eliminate the data elements that cannot belong to the final skyline. Algorithm *LESS* in [6] is an improvement of algorithm *SFS* in [5]. *LESS* uses filter windows in main memory and stores in them filters that are close to the origin of the space. These filters are used for eliminating data during the first scan of the input data. However, in case the filters are clustered in some specific area, the filtering might not perform well.

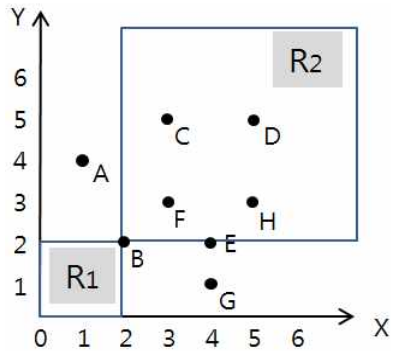
Skyline computation algorithms for client-server environments or parallel & distributed environments have also been proposed [1,10]. *SaLSa* algorithm in [2] assumes the client-server environment, and attempts to reduces the amount of data transfer to the clients. In order to do so, the original data in the server are sorted in advance. The client receives the sorted data from the server one by one, and computes skyline points incrementally. The client also initializes and updates the stop line while receiving the input data from the server. The client halts receiving data from the server if the current input is beyond the stop line, meaning that there are no further skyline points. The work in [10] proposes a skyline computation algorithm for parallel and distributed environment. It uses angle-based partitioning method in order to distribute the multidimensional data to each processor evenly. That is, the partitioning is used for work load balancing. The work in [7-10] propose a skyline computation algorithm that returns skyline points incrementally during the process of skyline computation.

3. PROPOSED FILTER LINING SCHEME

We now present our filter lining scheme. For convenience, we use a 2-dimensional data set as an example. First of all, we explore the dominance relationship between the points. Figure 2(a) shows 7 points in a 2-dimensional space. Among the points, $\{A, B, G\}$ are the points that belong to the skyline. Region R_2 of Figure 2(b) represents the



(a) A 2-dimensional data set



(b) The data region dominated by B

Fig. 2. The dominance relationship among multi-dimensional data points.

region of points that are dominated by B. In other words, all the coordinates of the points in R_2 are larger than those of B, respectively. Also, in order for B to belong to the skyline, there should not be any point in region R_1. It is because the coordinates of the points in R_1 are smaller than those of B, respectively. If B belongs to the filter line that are proposed in our work, $\{C, D, E, F, H\}$ will be eliminated during the filtering stage since they belong to R_2. Our objectives are to find filters as close to the origin as possible so that the region R_2 for the filters become as large as possible.

We now explain the proposed filter lining scheme. First of all, using an angle-based partitioning method, the multidimensional data space is divided into m equally spaced partitions. In Figure 3, $m=4$, and we have 4 partitions, P_0, P_1, P_2, P_3. Next step is to determine the initial filter F_i for partition P_i, $0 \le i \le m$. In order to find proper filters, we collect samples from the input data set. Among the samples, $s_1, s_2, \cdots, s_x$, that belong to P_i, if s_j is the sample that is the closest to the origin and its R_2 (in Figure 3) is the largest, then s_j is chosen to be the initial filter F_i for partition P_i. Note that for each sample, its partition number can be calculated in $O(1)$ time using the following equations (1) and (2). Here, (x,y) is the coordinates of the sample, 'degree of the point' is the (radian) angle of the sample (x,y) from the origin and X-axis, and 'number of partitions' is m.

$$\text{degree of the point} = \tan^{-1}\frac{y}{x} \times \frac{180}{\pi} \quad (1)$$

$$\text{partition ID} = \text{degree of the point} \ / \ \text{number of partitions} \quad (2)$$

Next step is to connect the filter $(F_0 \sim F_{m-1})$ for each partition to establish the initial filter line. In Figure 3, the line connecting $F_0 \sim F_3$ is the initial filter line, and we regard the area right to the filter line as region R_2 (Figure 2). Now, let us pay attention to the filter line for partition P_2 in Figure 3. It is affected by the coordinates of F_1, F_2, and F_3. Thus by using the filter line, each partition is considered to have three filters. Next, we take a look at the relationships between the initial filters. Note that F_i is not dominated by any other samples that belong to partition P_i. However, there might be F_j, $j \neq i$, $0 \le j \le m-1$, that dominates F_i. If such F_j is closer to the origin, then F_j is better than F_i for partition P_i. Thus, F_i is replaced by F_j. Now, the initial filter line is completely established. [Algorithm 1] is a formal description of what we explained so far.

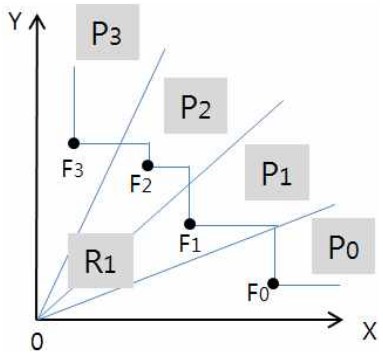


Fig. 3. Filter Lining ($m=4$).

[Algorithm 1] Initial filter lining for each partition

[Assumption] partitions $P_0, P_2, \cdots, P_{m-1}$, sample size $s = c \times m$ (c: constant)

[step 1] Collect s samples from the input data set, and compute the partition number for each sample.

[step 2] Among the samples for P_i, $0 \le i \le m-1$, choose one that is closer to the origin than other samples, and its R_2 is the largest. Let it be F_i.

[step 3] For each partition P_i, $0 \le i \le m-1$, if F_i is dominated by other filters, choose one (F_j) that is closer to the origin, and its R_2 is the largest. Replace F_i by F_j.

After the initial filter line is established, the input data is scanned to eliminate the data that belong

to region R_2 of the filter line. During this stage, filters are adaptively updated and the filter line gradually approaches the final skyline set. The filter line gets very close to the skyline at the end of the data scanning. Input data scanning stage can be described as in [Algorithm 2]. First, a data item, say a, is read, and find the partition number using equations (1) and (2). Suppose that a is in partition P_i. If a is dominated by filters F_{i-1}, F_i, F_{i+1}, it is eliminated. Otherwise, a remains in partition P_i and a is then examined to see if it can be a better filter for partition P_i. If a is proven to be a better filter, then F_i is replaced by a. With such a filter updating scheme, filters gradually approach the origin of the space as well as the skyline of the input data. In addition, F_i might be dominated by other filters, F_j, $j \neq i$, $0 \leq j \leq m-1$. In this case, F_i is replaced by F_j periodically. Note that the comparisons between filters are conducted very frequently, it could be some overhead, thus such comparisons should be done with some intervals.

[Algorithm 2] Data Scan and Filter Line Update

Scan the input multidimensional data set and do the following step.

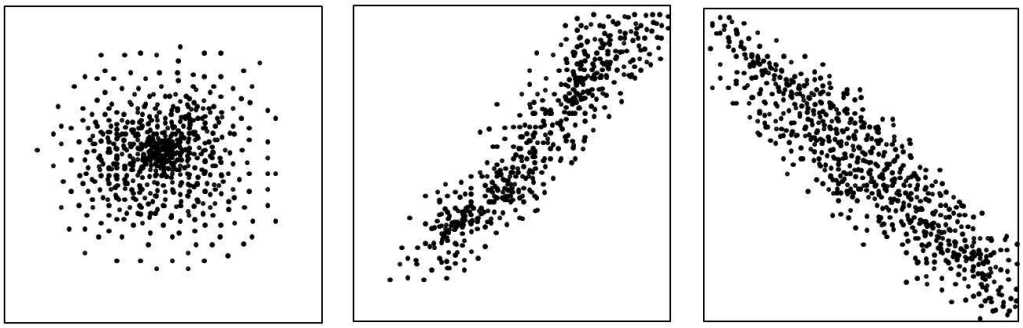
[step 1] For each data item a, find the corresponding partition P_i using the equations (1) and (2). Compare a with F_{i-1}, F_i, F_{i+1}. If a is dominated by any of the three filters, it is eliminated. Otherwise, a is compared with F_i to see if a is a better filter than F_i. In such a case, F_i is replaced by a. [step 3] of Algorithm 1 is also done periodically to improve the filter line.

After the filtering stage, the remaining data set is used for computing the skyline of the original multidimensional data set. It can be done as follows. After [Algorithm 2] is finished, the remaining data set is divided into partitions, $P_0, P_2, \cdots, P_{m-1}$. Each partition is sorted in increasing order of the distance from the origin of the data space. And the skyline is computed for each partition, independently. Let SKY_i be the skyline for partition P_i. SKY_i, $0 \leq i \leq m-1$, are compared to each other and are merged to get the skyline for the entire data set. Note that when SKY_i and SKY_{i+1} are merged, the points in SKY_i are only compared with the points in SKY_{i+1}, and the points only need to be compared with the points that are closer to the origin. It is because a point cannot dominate other point that is closer to the origin. In addition, the shape of the data space division looks like a fan blade shape. Thus, the points in each SKY_i are more likely to belong to the skyline for the entire data set. It means that our partitioning do not incur much overhead nor redundant computations.

4. EXPERIMENTS AND PERFORMANCE EVALUATION

Our experiments are conducted on a PC equipped with 2.40GHz Intel i5 CPU and 4.0 GB main memory. The programs are written in C and are executed in .NET environment. The objectives of the experiments are as follows. (1) Validate the effectiveness of the proposed scheme by measuring the time for filtering and skyline computation for data with various distributions and sizes. (2) Check the effectiveness of the data elimination using the proposed filter lining. (3) Compare the sum of the sizes of the skylines for all the partitions with that of the skyline for the entire input data set, to see the amount of the redundant computation. (4) Verify the superiority of the proposed scheme by varying the number of partitions.

The experiments are conducted using 2~3 dimensional data sets. First, we show the experimental results for 2-dimensional data sets. Data sets with various sizes and distributions are artificially synthesized and generated. The sizes of the data sets are 1MB, 5MB, 10MB, respectively. The data distributions used for experiments are ①



(a) Gaussian Distribution (b) Correlated Distribution (c) Anti-correlated distribution

Fig. 4. Data distributions used for experiments.

Gaussian distribution, ② correlated distribution, and ③ anti-correlated distribution, as shown in Figure 4. Circles with low density are used for Gaussian distribution. For correlated and anti-correlated data distributions, we use ovals by adjusting the ratio of the length of the horizontal axis to the length of the vertical axis. For correlated data distribution, the ovals are rotated 45° counterclockwise. For anti-correlated data distribution, the ovals are rotated 135° counterclockwise. Data sets are stored in main memory and the number of partitions is 9, meaning that the angle distance between partitions is 10° each. The number of partitions also vary to 18 (5° each), 9 (10° each), 4 (23° each). Sample size is 10,000. Random number generation is used for test data generation.

For Tables 1~3, T_1, T_2, and T_3 represent Gaussian Distribution, correlated distribution, and anti-correlated distribution, respectively. Table 1 shows the execution time for the data sets whose sizes are 10MB, 5MB, and 1MB, respectively. It shows the time taken by the filtering stage along with the time taken by the rest of the skyline computation. The latter is shown in parentheses. The time for filtering depends only on the size of the original input data sets since filtering stage consists of only one scan of the input data set. It rarely depends on the distribution of the data set. In addition, filtering only takes a very small amount of time. In the experiments, it takes only 1.14~1.18 second for data sets with 10MB. It shows that our filtering is very efficient.

Table 2 shows the percentage of the data remaining after the filtering is processed. The rate is independent of the data size, but dependent on the data distribution. For the data sets with Gaussian distribution, T_1, only 3.94%~3.99% are remaining after the filtering. That means 96% of the data items are eliminated. For the correlated distribution, we use an oval whose x-axis ratio to y-axis is 15:4. This data set is sparse and the oval is 45° rotated counterclockwise. For such data distribution, filter lining is very effective since the shape of the filter line is similar to that of the data sets. We can see that only 0.23% of the data remains after the filtering. On the other hand, for the data sets with anti-correlated distribution, which is 135° rotated counterclockwise, 19% of the data

Table 1. Time for filtering and the rest of the algorithm (Unit: sec)

Distribution \ Size	10MB	5MB	1MB
T1	1.141 (0.039)	0.564 (0.140)	0.115 (0.022)
T2	1.118 (0.014)	0.558 (0.007)	0.114 (0.001)
T3	1.178 (1.346)	0.600 (0.648)	0.119 (0.112)

Table 2. The effect of the data elimination of the filtering scheme

Distribution \ Size	10MB	5MB	1MB
T1	3.99%	3.94%	3.97%
T2	0.23%	0.23%	0.22%
T3	19.2%	19.0%	18.9%

remains after the filtering. Although the ratio for this case is rather larger than that for the other two distributions, it is considered to be very effective.

Table 3 gives experimental results showing the efficiency of the angle based partitioning, and the divide & conquer scheme of skyline computation. The proposed skyline computation algorithm first eliminates data using the filter line, and then computes the skyline from the remaining data set. Skyline points are computed for each partition independently, and they are merged to form the final set of the skyline for the entire data set. In this experiment, we compare the size (SKY_F) of the skyline for the entire data set with the size (SKY_{EA}) of the sum of the skyline points for each partition. For example, for the data set T_1 of size 10MB, SKY_{EA} is 1,156, while SKY_F is 1,152. We can see that only 4 points are not included in the final set of the skyline. For other data sets, the maximum differences between SKY_F and SKY_{EA} is less than 10. This experiment shows that the proposed partitioning and divide & conquer algorithm is outstanding.

Table 4 shows the filtering effect for various number of partitions. In this experiment, anti-correlated data distribution is used, and the size of the data is fixed to 10MB. We only vary the number of partitions. We place 4, 9, 18 partitions on the 2-dimensional data space, respectively. For each case, the angle width of each partition is 23°, 10°, 5°, respectively. When the number of partitions is 4, the rate of the remaining data is 55.2%. Only a half of the data items are eliminated. When the number of partitions is 18, the rate of the remaining data is around 19.2%. We notice that as many as 80% of the data items are eliminated. The time for computing the skyline after filtering stage depends on the size of the remaining data sets. For partitions of size 4, 9, 18, the time for skyline computation is 6.5 sec, 2.9 sec, and 1.3 sec, respectively. That is, the filtering effect becomes bigger if the size of the partitions gets smaller.

We conducted experiments using 3-dimensional data sets, and the results are shown in Table 5. The objective of the experiments is to see how bad the overhead of the algorithm for high dimensional data is, and to see the relationships between the

Table 3. Comparison between SKY_F and SKY_{EA} (Unit : Count)

Distribution \ Size	10MB	5MB	1MB
T1	1,156 (1,152)	943 (939)	519 (516)
T2	78 (70)	77 (69)	63 (55)
T3	1,386 (1,379)	1,357 (1,350)	1,034 (1,021)

Table 4. Filtering effect for various number of partitions (Data distribution : T_3, Data size : 10MB)

(time unit: sec)

Items \ Number of partitions	4 (23^o)	9 (10^o)	18 (5^o)
Ratio of remaining data after filtering	55.2%	33.1%	19.2%
Skyline computation time	6.589 sec	2.932 sec	1.346 sec
Size of the skyline	1,382	1,380	1,379

number of partitions and the filtering effect. The 3-dimensional data set used for the experiment follows Gaussian distribution. The data set is generated similarly with 2-dimensional data sets. Let D_1, D_2, and D_3 represent the three dimensions respectively. Plane $D_1 \times D_2$ and plane $D_1 \times D_3$ are divided into 1, 2, ... , 9 partitions respectively. The 3-dimensional partitions are formed by intersecting these 2 dimensional partitions. That is, the number of partitions vary from 1×1 to 9×9, as in Figure 5. The size of the input data is 10MB. Note that the filtering stage consists of three steps. First, the input data set is scanned. Second, partition number is determined for each data item. Third, each data item is compared with the filter for the partition. Among the three steps, the second step is where some overhead might incur. However, it only takes $O(1)$ time to determine partition numbers. We use equations (1) and (2) to determine partition number P_1 on plane $D_1 \times D_2$. Partition number P_2 on plane $D_1 \times D_3$ is determined similarly. Partition number for the input data is set to be (P_1, P_2) . When the dimension of the data set gets increased by 1, what is needed is to compute equations (1) and (2) one more time. As shown in the 4th column of Table 5, partition number determining process does not incur overhead at all. Table 5 shows the experimental results by varying the number of partitions from 1×1 to 9×9. When the number of partitions is 9×9, only 7.39% of the data remain after the filtering stage. When the number of partitions is 1×1, as large as 33% of the data remain after the filtering stage. In addition, the difference between SKY_F and SKY_{EA} is very small, meaning that skyline points for each partition are very likely to be included in the final set of skyline points. It shows that the proposed scheme is very effective. The time taken by the stage of scanning the input and filtering out dominated data items is 1.5~1.6 seconds. However, for the case of 9×9 partitions, skyline computation takes 1.83 seconds. On the other hand, for the case of 1×1 partitions, skyline computation takes as large as 58.79 seconds.

Let us now compare our filter lining scheme with previously reported skyline computation algorithms. The previously reported filtering schemes can be considered to be the last row of Table 5, since it has only one filter along with one partition. Skyline computation algorithms without being pre-processed using filtering schemes mostly sort the input data elements. When the size of the data set is n, the time taken by the skyline computation is at least $O(n \log n)$. In Figure 5, the last row shows that the time taken by the skyline computation is as large as 58.79 second when 33% of the input data are not filtered. If filtering is not used in the preprocessing stage, the time taken by

Table 5. Filtering effect and overhead for 3-dimensional data sets (time unit : sec)

Rate of the remaining data	SKY_{EA}	SKY_F	Time for filtering	Time after filtering	Number of partitions
7.39%	18,663	18,165	1.625	1.831	9×9
8.19%	18,665	18,181	1.623	2.037	8×8
8.87%	17,913	17,515	1.587	2.117	7×7
10.99%	18,557	18,199	1.601	2.734	6×6
13.10%	18,570	18,296	1.582	3.620	5×5
16.50%	18,446	18,241	1.610	4.977	4×4
20.17%	18,444	18,254	1.598	9.377	3×3
31.514%	18,301	18,252	1.619	17.849	2×2
33.092%	18,218	18,217	1.513	58.789	1×1

the sorting the entire input data set is too large to be considered to be reasonable. Thus, in our experiments, instead of comparing our scheme with such previous schemes, we rather attempt to show the effectiveness of the proposed scheme in various environments.

5. CONCLUSION

We propose a filter lining scheme that can be used in the preprocessing stage of skyline computations. The initial filter line is established as wide in the data space as possible so that as many data items as possible can be eliminated. The filter line gets incrementally updated during the data scanning stage. At the end of the filtering stage, the filter line gets very close to the skyline of the input data set. In addition, during the filter line establishing stage, the input data set is divided into angle based partitions. Skyline is then computed for each partition. Note that the skyline computed for each partition is very likely to be part of the skyline for the entire data set. Thus, the proposed filter lining scheme greatly reduces the number of comparisons between data elements. The effectiveness of the proposed scheme is shown by experiments.

REFERENCES

[1] 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.

[2] I. Bartolini, P. Ciaccia, and M. Patella, "SaLSa: Computing the Skyline Without Scanning the Whole Sky," In *CIKM 2006*, pp. 405-414, Arlington, Virginia, USA, 2006.

[3] Sung Koo, Lee, "A Recommendation System Based on Customer Preference Analysis and Filter Management," *Journal of Korea Multimedia Society*, Vol.7, No.4, pp. 592-600, 2004.

[4] S. Borzsonyi, D. Kossmann, and K. Stocker, "The Skyline Operator," In *ICDE 2001*, pp. 421~430, Heidelberg, Germany, 2001.

[5] J. Chomicki, P. Godfrey, J. Gryz, and D. Liang, "Skyline with Presorting," In *ICDE 2003*, pp. 717-719, India, 2003.

[6] P. Godfrey, R. Shipley, and J. Gryz, "Maximal Vector Computation in Large Data Sets," In *VLDB 2005*, pp. 229-240, Trondheim, Norway, 2005.

[7] D. Papadias, Y. Tao, G. Fu, and B. Seeger, "Progressive Skyline Computation in Database Systems," *ACM Transactions on Database Systems,* Vol.30, No.1, pp. 41-82, 2005.

[8] A. Siddique and Y. Morimoto, "K-Dominant Skyline Computation by Using Sort-Filtering Method," In *PAKDD 2009*, pp. 839-848, 2009.

[9] K.-L. Tan, P.-K. Eng, and B. C. Ooi, "Efficient Progressive Skyline Computation," In *VLDB 2001*, pp. 301-310, Roma, Italy, 2001.

[10] A. Vlachou, C. Doulkeridis, and Y. Kotidis, "Angle-based space partitioning for efficient parallel skyline computation," In *ACM SIGMOD 2008*, pp. 227-238, Vancouver, Canada, 2008.

Jihyun Kim

1995: BS in Computer Science, Sangmyung University
2007: MS in Computer Science and Engineering, Ewha Womans University
2011: Ph.D Candidate, Dept. of Computer Science and Engineering, Ewha Womans University
Research Interests : Moving Object Database, Skyline Computation, Data Analysis, Stream data, Ontology, Information Retrieval

Myung Kim

1981 : BS in Mathematics, Ewha Womans University
1983 : MS in Computer Science, Seoul National University
1993 : Ph.D. in Computer Science, University of California, Santa Barbara, USA
1993～1995: PostDoc, University of California, Santa Barbara, USA
1995～Present : Professor, Dept. of Computer Science and Engineering, Ewha Womans University
Research Interests : Data Analysis, Realtime Recommendation System, Information Retrieval, High Performance Computing