K-Dominant Skyline Computation by using Sort-Filtering Method

Md. Anisuzzaman Siddique and Yasuhiko Morimoto

Hiroshima University,
1-7-1 Kagamiyama, Higashi-Hiroshima, 739-8521, Japan
{d074370@,morimoto@mis.}hiroshima-u.ac.jp,+81-82-424-5579

Abstract. Skyline queries are useful in many applications such as multi-criteria decision making, data mining, and user preference queries. A skyline query returns a set of interesting data objects that are not dominated in all dimensions by any other objects. For a high-dimensional database, sometimes it returns too many data objects to analyze intensively. To reduce the number of returned objects and to find more important and meaningful objects, we consider a problem of **k**-dominant skyline queries. Given an **n**-dimensional database, an object **p** is said to **k**-dominates another object **q** if there are (**k**≤**n**) dimensions in which **p** is better than or equal to **q**. A **k**-dominant skyline object is an object that is not **k**-dominated by any other objects. In contrast, conventional skyline objects are **n**-dominant objects. We propose an efficient method for computing **k**-dominant skyline queries. Intensive performance study using real and synthetic datasets demonstrated that our method is efficient and scalable.

Key words: k-Dominant Skyline, Domination Power, Sort-Filtering.

1 Introduction

Skyline queries have attracted considerable attention due to its importance in many applications such as multi-criteria decision making, data mining, and user preference queries [1]. Given a database X, an object p is said to be in skyline of X if there is no other object q in X such that q is better than p in all dimensions. If there exist such a q, then we say that p is dominated by q, or q dominates p. A number of efficient algorithms for computing all skyline objects have been reported in the literature [1–5].

There are two problems in conventional skyline queries: (i) As the number of dimensions increases, the number of skyline objects increases substantially because it becomes difficult to dominate other objects. (ii) Usually, users have to select some noteworthy objects from skyline objects. Sometimes, the users may have to select many objects. Sometimes, they have to select a few objects. Conventional skyline query cannot control such selectivity.

In this paper, we consider k-dominant skyline queries [6], which are considered to overcome the above problems.

1.1 Motivated Example

Assume a person want to purchase a notebook computer and is looking for suitable one. Assume there is a database containing eight notebooks as listed in Table 1. In the table, each notebook is represented as a tuple containing six attributes, CPU, RAM, HDD, HDD speed, Quality, and VRAM. Without loss of generality, we assume larger value is better in each attribute and all the attributes have equal importance.

Conventional skyline query for this database returns six notebooks: N_2, N_3, N_4, N_5, N_6 and N_7. N_1 and N_8 are not in skyline because both are dominated by N_2. If we look the six skyline notebooks, we can find that not all notebooks are significant in a sense. For example, N_3 is survived only by its value of "HDD Speed" and N_7 is survived only for "RAM" size. N_6 is skyline because no other notebook fails to dominate it in all dimensions, even though it does not have any maximal feature values. In such situation, the person naturally consider to eliminate the skyline notebooks by using stronger criterion.

Table 1. Database for Notebook PCs

NoteBook	CPU	RAM	HDD	HDD Speed	Q.	VRAM
N_1	3	3	5	6	6	8
N_2	9	4	9	7	7	9
N_3	8	4	7	9	2	7
N_4	5	6	8	9	5	9
N_5	9	7	9	6	2	4
N_6	6	6	6	5	3	5
N_7	5	7	3	8	4	6
N_8	4	4	8	6	6	4

Chan et al. considered k-dominant skyline query to handle the problem [6]. They relaxed the definition of "dominated" so that an object is likely to be dominated by another. Given a database X consists of n attributes, an object p is said to be in k-dominant skyline of X if there is no object q in X such that q is better than p in k ($\leq$n) dimensions. If there exist such a q, then we say that p is k-dominated by q or q k-dominates p.

In the example, if we consider 5-dominant skyline instead of conventional skyline, i.e., 6-dominant skyline, N_1, N_3, N_5, N_6, and N_8 are eliminated from 5-dominant skyline because they all are 5-dominated by N_2. N_7 fails to become 5-dominant skyline because it is 5-dominated by N_4.

If the person is satisfied with the selectivity of 5-dominant skyline, he/she can analyze the returned notebooks intensively. If he/she is still unsatisfied with the selectivity, he/she can compute further k-dominant skyline query with smaller k. Thus, k-dominant query solve the problems of conventional skyline query for high dimensional databases.

The contributions of this paper are as follows: 1) We have developped an efficient method for computing k-dominant skyline by using a *Sort-Filtering*

method that sorts objects by *domination power*. 2) We have performed intensive experiments on a variety of synthetic and real datasets to demonstrate that the proposed method is efficient and performs better than other existing methods.

2 *k*-Dominant Skyline

2.1 Preliminaries

Assume there is an n-dimensional database X containing m tuples. Let d_1,d_2,...,d_n be n attributes of X and let p_1,p_2,...,p_m be m tuples of X. We use $p_i.d_j$ to denote the j-th dimension value of p_i.

An object p_i is said to *dominate* another object q_j, denoted as $p_i \geq q_j$, if $p_i.d_k \geq q_j.d_k$ for all attributes d_k $(k = 1, \cdots, n)$ and $p_i.d_t > q_j.d_t$ for at least one dimension d_t $(1 \leq t \leq n)$. We call such p_i a *dominant object* and such q_j a *dominated object* between p_i and q_j. An object $p_i \in X$ is said to be a *skyline object* of X if p_i is not dominated by any other object in X.

An object p_i is said to *k-dominate* another object q_j, denoted as $p_i \geq_k q_j$, if $p_i.d_k \geq q_j.d_k$ in k attributes among n attributes and $p_i.d_t > q_j.d_t$ in an attribute d_t among the k attributes. We call such p_i as *k-dominant object* and such q_j as *k-dominated object* between p_i and q_j. An object p_i is said to be a *k-dominant skyline object* of X, if and only if there does not exist any object p_j $(j \neq i)$ in X that k-dominates p_i.

An object p_i is said to have *δ-domination power* if there are δ attributes in which p_i is better than or equal to all other objects of X.

2.2 A Priori Property

A k-dominant object has the following a priori property.

Theorem 1. *Any $(k-1)$-dominant object must be a k-dominant object for any k such that $1 < k \leq n$.*

Theorem 2. *Any k-dominated objects cannot be a $(k-1)$-dominant object for any k such that $1 < k \leq n$.*

Proof. Based on the definition, a $(k-1)$-dominant object p is not $(k-1)$-dominated by any other objects in X. It implies that p is not k-dominated by any other objects. Therefore, we can say p is k-dominant object. On the other hand, if an object q is k-dominated by another object, it must be $(k-1)$-dominated by the object. Therefore, q cannot be a $(k-1)$-dominant object.

The conventional skyline is the n-dominant skyline. If we decrease k of the k-dominant skyline, more objects are eliminated. For example, N_1 and N_8 of Table 1 are not in skyline because they are dominated (n-dominated) by N_2. So, they can't be a candidate of k-dominant skyline object for $k < n$. We can prune such non-skyline objects for further procedure of the k-dominant query. If we consider 5-dominant query, N_3, N_5, N_6, and N_7 are 5-dominated objects

in addition to the 6-dominated objects, N_1 and N_8. Therefore, we can prune those objects in 5-dominant query computation. Thus, by decreasing k, more dominated objects can be pruned away.

3 k-dominant Skyline Algorithm

In this section, we present an efficient method for computing k-dominant skyline objects from X. We used a Sort-Filtering method that consists of two parts: one is "domination power" calculation and sorting, and the other is k-dominant skyline objects checking.

3.1 Domination Power Calculation

Objects whose sum of all their dimension values is large are likely to dominate other objects, while objects whose sum is small are likely to be dominated. Therefore, we sort the whole tuples in X in descending order of the sum of all their dimension values. This preprocess, sorting by sum, has been proposed by Chomicki et al. [4]. By this preprocessing, we can eliminate some of non-skyline objects easily. Chan et al. used the popular preprocessing in their OSA algorithm for k-dominant query [6]. But this preprocess is not effective for k-dominant query computation especially when values of each attribute is not normalized. For example, assume p(9,1,2) and q(3,2,3) are two objects in 3D space. Although the object p has greater sum than object q but p fails to become 2-dominant of q. Here, object p is 2-dominated by object q.

Therefore, in order to prune unnecessary objects efficiently in the k-dominant skyline computation, we compute *domination power* of each object, i.e., how many maximal values it has within all of dimensions. Then, we sort objects in descending order by domination power. If more than one objects have same domination power then we sort those objects in descending order of the sum value.

Without apply any sorting Table 2 represent the domination power and sum of the each Notebook PC's of Table 1. Table 3 is the example of sorted database (sort Notebook PC's in descending order by corresponding domination power and for same domination power sort in descending order of the sum value) of the Notebook PC's database of Table 1. In the sorted table, N_2 has the highest dominant power 4 and N_8, N_1 and N_6 have no dominant power. Note that N_2 dominates all notebooks lie below it in four attributes, CPU, HDD, Quality, and VRAM.

Let X' be the sorted database of X. X' has the following property.

Theorem 3. *Let p_i be the i-th object in the ordered object sequence of X'. If p_i has δ-domination power, it δ-dominates p_j for j such that $j > i$ in the sequence of X'.*

Proof. An object p_i with δ-domination power is δ-dominant object to any other object in X'. Since X' is sorted by domination power in descending order, an

Table 2. Domination Power Calculation

Notebook	CPU	RAM	HDD	HDD S.	Q.	VRAM	Domination Power	Sum
N_1	3	3	5	6	6	8	0	31
N_2	9	4	9	7	7	9	4	45
N_3	8	4	7	9	2	7	1	37
N_4	5	6	8	9	5	9	2	42
N_5	9	7	9	6	2	4	3	37
N_6	6	6	6	5	3	5	0	31
N_7	5	7	3	8	4	6	1	33
N_8	4	4	8	6	6	4	0	32

Table 3. Sorted Dataset

Notebook	CPU	RAM	HDD	HDD S.	Q.	VRAM	Domination Power	Sum
N_2	9	4	9	7	7	9	4	45
N_5	9	7	9	6	2	4	3	37
N_4	5	6	8	9	5	9	2	42
N_3	8	4	7	9	2	7	1	37
N_7	5	7	3	8	4	6	1	33
N_8	4	4	8	6	6	4	0	32
N_1	3	3	5	6	6	8	0	31
N_6	6	6	6	5	3	5	0	31

object with δ-domination power always comes before objects whose domination power is less than δ. Therefore, from the definition we can say p_i is a δ-dominant for any other object p_j since $j > i$.

By using this property, an object with k-domination power k-dominates all other following less domination power objects. Moreover, when more than one object has same domination power then our proposed method sort those objects in descending order of the sum values. Therefore higher objects in the sorted sequence are likely to dominate other objects. This sort filtering preprocessing helps to reduce the computational cost of k-dominant skyline.

The sorted sequence roughly reflects the importance of objects and our method can progressively output the k-dominant objects based on the sequence. It helps users' to make their decision more practical.

3.2 k-dominant Checking

By using X', we progressively output k-dominant skyline objects as follows. We scan X' to compare each object $p \in X'$ against the first object q. In the scan procedure, objects that are k-dominated by the first object are removed from X'. During the procedure, if the first object q is k-dominated by any other objects $p \in X'$, we remove q from X' and stop the scanning procedure. If q is not removed in the scanning procedure, then output q as a k-dominant object and remove q from X'. We repeat this scanning procedure until X' becomes empty.

Applying the k-dominant check with $k = 5$ for Table 3, we note that in the first scan N_5, N_3, N_8, N_1 and N_6 are 5-dominated by the first object, N_2. Therefore, those dominated objects are removed from X'. On the other hand, N_2 is not 5-dominated by any other notebooks. So, after the first scan, our method outputs N_2 as a 5-dominant skyline object and then remove N_2 from X'. In the second scan, N_4 becomes the first object and it 5-dominates N_7. So, we remove N_7 and outputs N_4 as a 5-dominant skyline object. Similarly, if we apply $k = 4$ for the same database X', then our Sort-Filtering method returns only N_2 as a 4-dominant skyline object.

4 Related Works

Chan et al. introduce k-dominant skyline query [6]. They proposed three algorithms to compute the k-dominant skyline query. The first algorithm, One-Scan Algorithm (OSA), uses the property that a k-dominant skyline objects cannot be worse than any skyline on more than k dimensions. This algorithm maintains the skyline objects in a buffer during a scan of the dataset and uses them to prune away objects that are k-dominated. As the whole set of skyline objects can be large, the authors proposed the Two-Scan Algorithm (TSA). In the first scan, a candidate set of dominant skyline objects is retrieved by comparing every object with a set of candidates. The second scan verifies whether these objects are truly dominant skyline objects. This method turns out to be much more efficient than the one-scan method. A theoretical analysis is provided to show the reason for its superiority. The third algorithm, Sorted Retrieval Algorithm (SRA), is motivated by the rank aggregation algorithm proposed by Fagin et al., which pre-sorts data objects separately according to each dimension and then merges these ranked lists [7].

As the authors mentioned in the OSA, skyline objects need to be maintained to compute the k-dominant skyline objects. Since the set of skyline objects could be much larger than the set of k-dominant skyline objects, maintaining skyline can incur large space and computation overhead.

Compared with their works, the proposed method can find k-dominant skyline objects without maintaining skyline. Therefore, there is no possibility for space and computational overhead. In addition, TSA algorithm scans whole data twice. In first scan, it generates candidate set of dominant skyline objects by comparing every object with a set of candidates and in second scan it verifies whether these objects are truly dominant skyline objects, while the proposed method can compute dominant skyline directly and does not suffer for false positive elimination procedure. As for SRA, the performance is uncertain because it depends crucially on the choice of proper dimension. Section 5 demonstrates that the performance of our algorithm is better than all of the three algorithms proposed in [6].

Algorithm called CoSMuQ also computes k-dominant skyline [8]. It divides the space in pairs of attributes and maintains a grid for each pair of dimensions. Each grid maintains its skyline tuples. Finally, the k-dominant skyline is

obtained by the union of the skylines of these grids. This method has two severe problems, in high dimensional case. It needs to maintain huge number of grids. For example, if dimension size is equal to 15, then this algorithm needs to maintain 105 grids. In addition to the space complexity problem, CoSMuQ always needs to maintained 2-dominant skyline to compute k-dominant skyline. Compared with theirs, the proposed method does not suffer from such kinds grid as well as 2-dominant skyline maintaining problems.

5 Performance Evaluation

We have conducted a series of experiments to evaluate the performance of our Sort-Filtering method. We also compare the performance with all the algorithms proposed by Chan et al. which are One-Scan Algorithm (OSA), Two-Scan Algorithm (TSA) and Sorted Retrieval Algorithm (SRA) [6]. To make the comparison fair, we have include all the preprocessing cost, i.e., cost of domination power computation and sorting.

5.1 DataSets

We use both synthetic datasets and real datasets in the experiments. The generation of the synthetic datasets is controlled by three parameters, dimension number n, data size $Size$ and distribution $Dist$. There are three optional distributions in the synthetic data sets: Correlated, Independent and Anti-Correlated. Table 4 shows the number of the k-dominant skyline objects on 15-dimensional data set with 100k objects on different distributions and different constraint parameter k.

When k is close to the dimension size, the number of k-dominant skyline objects in the anti-correlated dataset becomes much larger than that of the independent and correlated datasets. However, when k is small, the correlated dataset can still have some dominant skyline objects, while no dominant skyline objects can be found on the other two distributions.

Table 4. Number of k-Dominant Skyline Objects

k	Correlated	Independent	Anti-Correlated
8	1	0	0
9	1	0	2
10	1	0	5
11	6	16	33
12	17	178	500
13	84	2180	5670
14	433	16143	29828

We also examine the performance for a real dataset. We used the NBA statistics. This dataset contains 17000 players season records on 17 attributes from

the first season of NBA in 1945. Every records contain the statistical value of a player's performance in one season, such as game played(GP), field goal made(FGM), total assists(AST) and so on. One player may have several records if he played in NBA for more than one season.

5.2 Performance

We evaluated the computational cost of our Sort-Filtering method and compared the result with all three algorithms proposed by Chan et al. with similar parameter setting. The proposed method is implemented in Java. All experiments were conducted on a PC with an Intel Pentium 3GHz CPU and 2G main memory, which is running on Microsoft Windows XP operating systems. Figure 1, 2, and 3 are the time to compute k-dominant skyline for synthetic datasets.

Figure 1 examines the effect of the k value. We observe that our method is more efficient than other three methods on all distributions when k varies from 8 to 14. Because, maintaining the set of skyline objects for OSA incurs large space and computation overhead. To compute k-dominant skyline, TSA suffered for false candidates elimination. Again, the performance of SRA is not better than Sort-Filtering because this approach crucially depends on the choice of proper dimension.

Figure 2 examines the effect of the dimension value. When k is small, TSA, RSA and Sort-Filtering are much faster than OSA on all three distributions. With increasing dimensionality TSA is several times slower than the other three algorithms. As shown in the figure, our Sort-Filtering is more scalable on high dimensional data sets.

In order to evaluate the effect of cardinality on the performance of the four algorithms, we use datasets with cardinality 50k, 100k, 150k and 200k. In this experiment, we fixed the size of n to 15 and k to 11. Figure 3 shows that when the size of the data set increases from 50k to 200k, the computation time of the four algorithms maintain a positive correlation. Notice that our Sort-Filtering performs best while OSA is the most worst.

In Figure 4, we show the experimental result on the NBA data set. When varying the constraint parameter k, TSA and our Sort-Filtering are the efficient algorithm when $k < 14$, but RSA is worst among the four algorithms when $k > 15$. Sort-Filtering and SRA are faster than other two when k is large.

6 Conclusion

We consider k-dominant skyline query problem and present a Sort-Filtering method. We demonstrate that our method is easy to compute and can be used for high dimensional large datasets. Performance evaluations show the superiority of the proposed method against the OSA, TSA and SRA algorithms.

k-Dominant skyline reduces the number of interesting objects returned by skyline. But sometimes the number of k-dominant skyline objects can still be large when k is not sufficiently small. If k is too small, no (or few) k-dominant

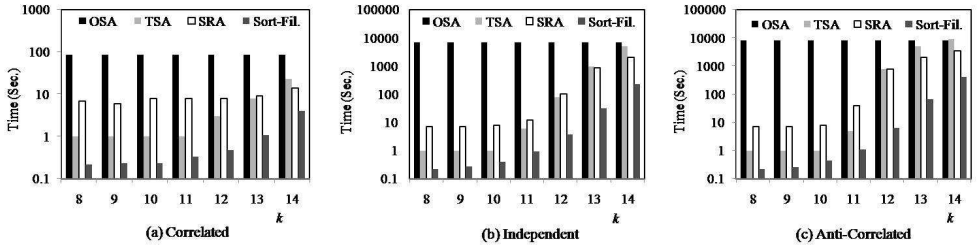

Fig. 1. k-Dominant Skyline Test on Varying k

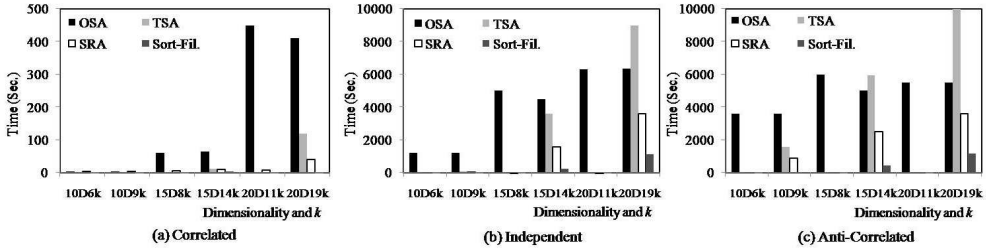

Fig. 2. k-Dominant Skyline Test on Varying *dimension*

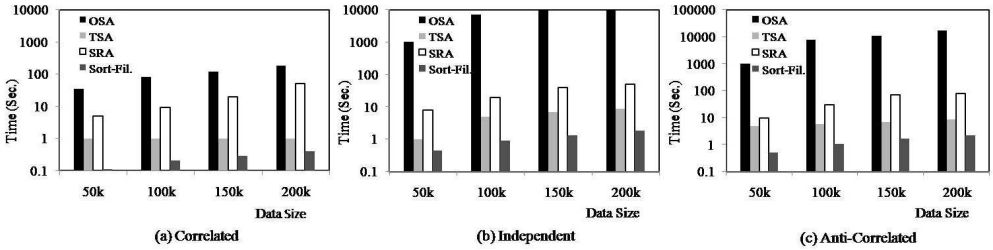

Fig. 3. k-Dominant Skyline Test on Varying *datasize*

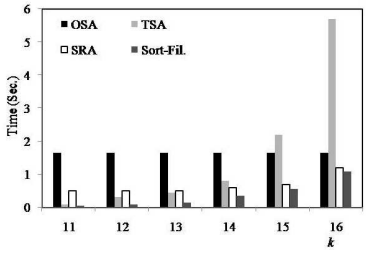

Fig. 4. k-Dominant Skyline Test on NBA Dataset Varying k

skyline objects are returned. Though our efficient computation allows us to compute k-dominant objects for various k, proper guide for choosing the right value of k is an open problem.

Acknowledgements

This work was supported by KAKENHI (19500123) and Md. Anisuzzaman Siddique was supported by the scholarship of MEXT Japan.

References

1. Xia, T., Zhang, D., Tao, Y.: On Skylining with Flexible Dominance Relation. In: Proceedings of ICDE, pp. 1397–1399, Mexico (2008)
2. Borzsonyi, S., Kossmann, D., Stocker, K.: The Skyline Operator. In: Proceedings of ICDE, pp. 421–430, Germany (2001)
3. Kossmann, D., Ramsak, F., Rost, S.: Shooting Stars in the Sky: An Online Algorithm for Skyline Queries. In: Proceedings of VLDB, pp. 275–286, China (2002)
4. Chomicki, J., Godfrey, P., Gryz, J., Liang, D.: Skyline with Presorting. In: Proceedings of ICDE, pp. 717–719, India (2003)
5. Papadias, D., Y., Tao, G., Fu, Seeger, B.: Progressive Skyline Computation in Database Systems. ACM Transactions on Database Systems. 30(1), pp. 41–82 (2005)
6. Chan, C.Y., Jagadish, H.V., Tan, K-L., Tung, A.K.H., Zhang, Z.: Finding k-Dominant Skyline in High Dimensional Space. In: Proceedings of ACM SIGMOD, pp. 503–514, USA (2006)
7. Fagin, R., Lotem, A. Naor, M.: Optimal Aggregation Algorithms for Middleware. ACM PODS, pp. 102–113, USA (2001)
8. Kontaki, M., Papadopoulos, A.N., Manolopoulos, Y.: Continuous k-Dominant Skyline Computation on Multidimensional Data Streams. In: Proceedings of ACM SAC, pp. 16–20, Brazil (2008)