Efficient k-dominant Skyline Computation for High Dimensional Space with Domination Power Index

Md. Anisuzzaman Siddique*, Yasuhiko Morimoto**
* Dept. of Computer Science and Eng.,University of Rajshahi, Rajshahi, Bangladesh
** Graduate School of Eng., Hiroshima University, Higashi-Hiroshima, Japan
Email: *anis_cst@yahoo.com, ** morimoto@mis.hiroshima-u.ac.jp

***Abstract*— Skyline queries have recently attracted a lot of attention for its intuitive query formulation. It can act as a filter to discard sub-optimal objects. However, a major drawback of skyline is that, in datasets with many dimensions, the number of skyline objects becomes large and no longer offer any interesting insights. To solve the problem, recently k-dominant skyline queries have been introduced, which can reduce the number of skyline objects by relaxing the definition of the dominance. This paper addresses the problem of k-dominant skyline objects for high dimensional dataset. We propose algorithms for k-dominant skyline computation. Our algorithms reduce the pairwise comparison between the k-dominant skyline objects and the dataset. Through extensive experiments with real and synthetic datasets, we confirm that our algorithms can efficiently compute k-dominant skyline queries.**

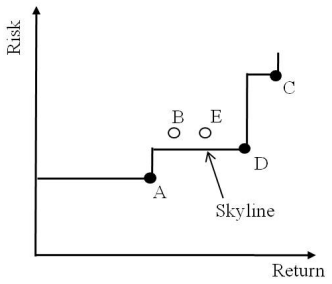
***Index Terms*— skyline, k-dominant skyline, domination power, dataset**

I. Introduction

The skyline query, which returns a set of objects not dominated by any other objects, has widely been applied in preference queries in relational datasets. Skyline query functions are important for several database applications, including customer information systems, decision support, data visualization, and so forth. Given an n-dimensional database DB, an object O_i is said to be in skyline of DB if there is no other object O_j $(i \neq j)$ in DB such that O_j is better than O_i in all dimensions. If there exist such O_j, then we say that O_i is dominated by O_j or O_j dominates O_i. Figure 1 shows an example of skyline. The table in the figure is a list of stocks, each of which contains two numerical attributes: $risk$ and $return$. Stock S_1 dominates stock S_2 if $S_1.risk \leq S_2.risk$, $S_1.return \geq S_2.return$, and strictly $S_1.risk < S_2.risk$ or $S_1.return > S_2.return$. The most interesting stocks are called skyline stocks which are not dominated by any other stock. From our example dataset an investor choice usually comes from the stocks in skyline, i.e., one of A, C, D (See figure 1(b)). Stock D has lower risk and higher return than B and E, meaning that D is better independently of the relative importance of the two attributes. On the other hand, D and A are incomparable since a long-term investor may be willing to obtain lower return to ensure lower risk. Hence, computing skyline from a dataset is helpful for users' decision-making. A

ID	Return	Risk
A	5	4
B	7	6
C	9	8
D	8	5
E	6	6

(a) Stocks



(b) Skyline

Figure 1. Skyline example

number of efficient algorithms for computing skyline objects have been reported in the literature [1]–[5], [11], [13], [14].

For skyline computation it is always assumed that the dominating relationship is evaluated based on every dimensions (attributes) of the dataset. However, a major drawback of skylines is that, in datasets with many dimensions, the number of skyline objects becomes large and no longer offer any interesting insights. The reason is that as the number of dimensions increases, for any object O_1, it is more likely there exists another objects O_2 where O_1 and O_2 are better than each other over different subsets of dimensions. If the investor, cared not just about risk and return, but also about the *price/earning (P/E) ratio* and *price-to-book ratio*, then most stocks may have to be included in the skyline answer since for each stock there may be no one stock that beats it on all criteria.

To reduce the size of result set and to find more important and meaningful objects, Chan, *et al.* considered k-dominant skyline query [6]. They relaxed the definition of "dominated" so that an object is more likely to be dominated by another. Given an n-dimensional dataset, an object O_i is said to k-dominates another object O_j $(i \neq j)$ if there are k $(k \leq n)$ dimensions in which O_i is better than or equal to O_j. A k-dominant skyline object is an object that is not k-dominated by any other objects.

Motivating Example. Assume we have a symbolic dataset as listed in Table I. In the table, each object is represented as a tuple containing six dimensions from D_1 to D_6. Without loss of generality, we assume smaller value

doi:10.4304/jcp.7.3.608-615

TABLE I.
SYMBOLIC DATASET

$Obj.$	D_1	D_2	D_3	D_4	D_5	D_6
O_1	7	3	5	4	4	3
O_2	3	4	4	5	1	3
O_3	4	3	2	3	5	4
O_4	5	3	5	4	1	2
O_5	1	4	1	1	2	4
O_6	5	3	4	5	1	5
O_7	1	2	5	3	1	2

is better in each dimension. Conventional skyline query for this dataset returns five objects: O_2, O_3, O_5, O_6, and O_7. Objects O_1 and O_4 are not in skyline because they are dominated by O_7. If we take a look at these skyline objects more closely, we can find that some objects are not significant in a sense. For example, compared with O_2, O_6 is survived only by its value of D_2. O_3 is in skyline because no other object fails to dominate it in all dimensions, even though it does not have any minimal feature value. In such a situation, the user naturally wants to eliminate the non-significant skyline objects by using selective criterion.

The k-dominant skyline query can control the selectivity by changing k. Consider the case where $k = 5$. Now the 5-dominant skyline query for this dataset returns two objects: O_5 and O_7. Objects O_1, O_2, O_3, O_4 and O_6 are not in 5-dominant skyline because they are 5-dominated by O_7. 4-dominant skyline query (i.e., $k = 4$) returns only one object, O_7. If we decrease the value of k by one, then 3-dominant skyline query returns empty.

The main contribution of this paper are as follows:

• We propose a new algorithm called Two Scan Algorithm with Domination Power Index (TSADPI), which offers a substantial improvement over k-dominant skyline computation.

• We conduct extensive performance evaluation using both real and synthetic datasets and compare our method with the Two-Scan Algorithm (TSA) technique [6], which is currently considered to be the most efficient k-dominant skyline method. Our evaluation shows that proposed method is significantly faster than TSA technique.

The rest of the paper are organized as follows: Section II the related work. Section III presents the notions and properties of k-dominant skyline computation. We provide detailed examples and analysis of the algorithm is examined in Section IV. We present experimental evaluations of the proposed algorithm in Section V under a variety of settings. Finally, the conclusion of the paper is explained with some directions for future work in Section VI.

II. RELATED WORK

Previous studies about skyline query processing are reviewed in this section. The related work on skyline query processing and k-dominant skyline query processing are discussed in Section II-A and II-B, respectively.

A. Skyline Query Processing

Borzsonyi, *et al.* first introduced the skyline operator over large datasets and proposed three algorithms: $Block\text{-}Nested\text{-}Loops(BNL)$, $Divide\text{-}and\text{-}Conquer(D\&C)$, and B-tree-based schemes [2]. BNL compares each object of the dataset with every other object, and reports it as a result only if any other object does not dominate it. A window W is allocated in main memory, and the input relation is sequentially scanned. In this way, a block of skyline objects is produced in every iteration. In case the window saturates, a temporary file is used to store objects that cannot be placed in W. This file is used as the input to the next pass. $D\&C$ divides the dataset into several partitions such that each partition can fit into memory. Skyline objects for each individual partition are then computed by a main-memory skyline algorithm. The final skyline is obtained by merging the skyline objects for each partition. Chomicki, *et al.* improved BNL by presorting and they proposed $Sort\text{-}Filter\text{-}Skyline(SFS)$ as a variant of BNL [4]. SFS requires the dataset to be pre-sorted according to some monotone scoring function. Since the order of the objects can guarantee that no object can dominate objects before it in the order, the comparisons of tuples are simplified.

Among index-based methods, Tan, *et al.* proposed two progressive skyline computing methods Bitmap and Index [7]. Both of them require preprocessing. In the Bitmap approach, every dimension value of an object is represented by a few bits. By applying bit-wise and operation on these vectors, a given object can be checked if it is in the skyline without referring to other objects. The index method organizes a set of n-dimensional objects into n lists such that an object O is assigned to list i if and only if its value at attribute i is the best among all attributes of O. Each list is indexed by a B-tree, and the skyline is computed by scanning the B-tree until an object that dominates the remaining entries in the B-trees is found. Kossmann, *et al.* observed that the skyline problem is closely related to the nearest neighbor (NN) search problem [3]. They proposed an algorithm that returns skyline objects progressively by applying nearest neighbor search on an R*-tree indexed dataset recursively. The current most efficient method is $Branch\text{-}and\text{-}Bound\ Skyline(BBS)$, proposed by Papadias, *et al.*, which is a progressive algorithm based on the best-first nearest neighbor (BF-NN) algorithm [5]. Instead of searching for nearest neighbor repeatedly, it directly prunes using the R*-tree structure. Balke, *et al.* show how to efficiently perform distributed skyline queries and thus essentially extend the expressiveness of querying current Web information systems [15]. Kapoor studies the problem of dynamically maintaining an effective data structure for an incremental skyline computation in a 2-dimensional space [16].

B. k-dominant Skyline Query Processing

Chan, *et al.* introduce k-dominant skyline query [6]. They proposed three algorithms, namely, One-Scan Al-

gorithm (OSA), Two-Scan Algorithm (TSA), and Sorted Retrieval Algorithm (SRA). OSA uses the property that a k-dominant skyline object cannot be worse than any skyline object on more than k dimensions. This algorithm maintains the skyline objects in a buffer during the scan of the dataset and uses them to prune away objects that are k-dominated. TSA retrieves a candidate set of dominant skyline objects in the first scan by comparing every object with a set of candidates. The second scan verifies whether these objects are truly dominant skyline objects or not. 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, 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 [8].

Based on transitivity property of skyline objects most of the above algorithms sort the whole tuples (objects) with a monotonic scoring function sum. However, this assumption is not true for k-dominant skyline computation due to the well-known intransitivity of the k-dominance relation. Moreover, there is an open issue that the efficiency of the most efficient k-dominant skyline search algorithm TSA proposed in [6] crucially depends on the pruning capability of non-dominant skyline objects during the first scan. If the number of false positives produced by the first scan is small, then the performance of TSA will be good.

Recently, more aspects of skyline computation have been explored. Vlachou, *et al.* introduce the concept of extended skyline set, which contains all data elements that are necessary to answer a skyline query in any arbitrary subspace [9]. Fotiadou, *et al.* mention about the efficient computation of extended skylines using bitmaps in [10]. Chan, *et al.* introduce the concept of *skyline frequency* to facilitate skyline retrieval in high-dimensional spaces [11]. Tao, *et al.* discuss skyline queries in arbitrary subspaces [12]. There exist more work addressing *spatial skyline* [17], [18], skylines on partially-ordered attributes [13], *dada cube* for analysis of dominance relationships [19], *probabilistic skyline* [20], skyline search over small domains [14], *reverse skyline* [21], and extended k-dominant skyline [22].

III. PRELIMINARIES

In this section, we first set the context and state the assumptions that are adopted in this paper. We then discuss the k-dominant skyline problems. Assume there is an n-dimensional dataset DB and D_1, D_2, $\cdots$, D_n be the n attributes of DB. Let O_1, O_2, $\cdots$, O_r be r objects (tuples) of DB. We use $O_i.D_j$ to denote the j-th dimension value of O_i.

A. k-dominance

An object O_i is said to *dominate* another object O_j, which we denote as $O_i \leq O_j$, if $O_i.D_s \leq O_j.D_s$ for all dimensions ($s = 1, \cdots, n$) and $O_i.D_t < O_j.D_t$ for at least one dimension ($1 \leq t \leq n$). We call such O_i as *dominant object* and such O_j as *dominated object* between O_i and O_j.

On the other hand, an object O_i is said to *k-dominate* another object O_j, denoted as $O_i \leq_k O_j$, if $O_i.D_s \leq O_j.D_s$ in k dimensions among n dimensions and $O_i.D_t < O_j.D_t$ in one dimension among the k dimensions. We call such O_i as *k-dominant object* and such O_j as *k-dominated object* between O_i and O_j.

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

B. k-dominant Skyline

An object $O_i \in DB$ is said to be a *k-dominant skyline object* of DB if O_i is not k-dominated by any other object in DB. We denote a set of all k-dominant skyline objects in DB as $Sky_k(DB)$.

Theorem: *Every object that belongs to the k-dominant skyline also belongs to the skyline, i.e., $Sky_k(DB) \subseteq Sky_n(DB)$.*

Proof: Let $O_1 \in Sky_k(DB)$ and $O_1 \notin Sky_n(DB)$. It follows that there is an another object O_2 that n-dominates the objects O_1. Based on the definition of skyline $\forall D_k (k = 1, \cdots, n)$: $O_2.D_k \leq O_1.D_k$. Therefore, based on the k-dominant skyline definition we find that $O_1 \notin Sky_k(DB)$, which leads to a contradiction. $\diamondsuit$

IV. TWO SCAN ALGORITHM WITH DOMINATION POWER INDEX (TSADPI)

We use a filter based algorithm to compute $Sky_k(DB)$ efficiently. It has two parts: one is sorting by domination power and the other is k-dominant skyline calculation.

A. Sort by Domination Power Index

By using ordered objects we can eliminate some of non-skyline objects easily. To get the benefit of ordered objects Chan, *et al.* sort the whole objects with a monotonic scoring function sum in their OSA algorithm for k-dominant query [6]. However, this sort is not effective for k-dominant query computation, especially, when values of each attribute is not normalized. For example, assume $O_i = (1, 2, 3, 3, 3, 2)$ and $O_j = (7, 1, 3, 2, 3, 1)$ are two objects in 6-dimensional space. Although sum of O_i's values is smaller than that of O_j's, O_i does not 5-dominant of O_j. Instead, O_i is 5-dominated by O_j.

To prune unnecessary objects efficiently in the k-dominant skyline computation, we compute *domination power* of each object. Algorithm 1 represents the domination power calculation procedure. An object is said to have *δ-domination power* if there are δ minimal values in which it is better or equal to all other objects of DB. We sort objects in descending order by their values of domination power (δ). If more than one objects have the same domination power then sort those objects in ascending order of the sum value. This order reflects how likely to k-dominate other objects. Higher objects in the sorted

Algorithm 1: *Compute Domination Power, DP*

```
01. for each object O_i(i = 1 ··· r) do
02.     O_i.DP := 0 (initialize DP for each object)
03. for each attribute D_s(s = 1 ··· n) do
04.     minValue := O_1.D_s
05.     for each object O_i(i = 2 ··· r) do
06.         if (O_i.D_s < minValue) then
07.             minValue := O_i.D_s
08.     for each object O_i(i = 1 ··· r) do
09.         if (minValue == O_i.D_s) then
10.             O_i.DP := O_i.DP + 1 (Increment DP)
11. Sort dataset, DB, in descending order by DP and Sum
```

TABLE II.
SORTED DATASET

Obj.	D_1	D_2	D_3	D_4	D_5	D_6	DP	Sum
O_7	1	2	5	3	1	2	4	14
O_5	1	4	1	1	2	4	3	13
O_4	5	3	5	4	1	2	2	20
O_2	3	4	4	5	1	3	1	20
O_6	5	3	4	5	1	5	1	23
O_3	4	3	2	3	5	4	0	21
O_1	7	3	5	4	4	3	0	26

sequence are likely to dominate other objects. Thus this preprocessing helps to reduce the computational cost of k-dominant skyline. Experiments show that our estimation is robust over various distributions. Moreover, it also works well when data values are correlated, independent or anti-correlated.

Table II is the example of sorted dataset of Table I. In the sorted dataset, object O_7 has the highest domination power 4. Note that object O_7 dominates all objects lie below it in four attributes, D_1, D_2, D_5, and D_6.

B. k-dominant Skyline Calculation

To confirm whether an object O is k-dominant or not, we need to compare it against all skyline objects. This is because O can be k-dominated by some skyline objects even though O is not k-dominated by any of the k-dominant objects. To eliminate non-k-dominant skyline objects, one set of objects are maintained as $Sky_k(DB)$. Our algorithm needs two scans for k-dominant skyline computation. The dataset sorted by domination power can reduce the pairwise comparison between the objects of $Sky_k(DB)$ and DB. Algorithm 2 shows our Two Scan Algorithm with Domination Power Index approach.

Table III represents the first scan comparison between TSA and TSADPI method to compute 5-dominant skyline. Columns 1, 2, and 3 represent the sorted DB, $Sky_5(DB)$ and no. of pairwise comparisons respectively during the first scan of TSA and column 4, 5, and 6 are those of TSADPI. Initially $Sky_5(DB)$ is empty and O_5 is added in $Sky_5(DB)$ as a 5-dominant object without any comparison. Since O_5 and O_7 fails to become 5-dominant of each other, after one comparison O_7 is added in $Sky_5(DB)$. Next after two comparisons we can see that O_2 is not in $Sky_5(DB)$. In this way objects O_4, O_3, O_6, and O_1 need 2, 1, 2, and 2 comparisons respectively. At the end of the first scan $Sky_5(DB)$ = $\{O_5, O_7\}$. From Table III we can see that TSADPI method needs only 6 comparisons while TSA needs 10.

Algorithm 2: TSADPI (DB, k)

```
1. Sort DB by domination power and sum
2. Intialize Sky_k(DB) = ∅
3.   for each object O ∈ DB do
4.     Initialize isDominant = true
5.     For each object O' ∈ Sky_k(DB) do
6.       If (O' k-dominates O or O' n-dominates O) then
7.         isDominant = false
8.         break
9.       If (O k-dominates O' or O n-dominates O') then
10.        Remove O' from Sky_k(DB)
11.    If (isDominant) then
12.      Insert O into Sky_k(DB)
13.  For each object O ∈ DB do
14.    For each object O' ∈ Sky_k(DB), do
15.      If ((O k-dominates O' or O k-dominates O') and O' ≠ O) then
16.        Remove O' from Sky_k(DB)
17.  Return Sky_k(DB)
```

To eliminate the false positives produced by the first scan, a second scan is necessary. To determine whether an object $O \in Sky_k(DB)$ is indeed k-dominant, it is sufficient to compare O against DB - $Sky_k(DB)$. If the number of false positives produced by the first scan is small, the performance of the second scan as well as the overall Two-Scan approach will be better. After the completion of second scan both of the methods give the result $Sky_5(DB)$ = $\{O_7, O_5\}$. Like this example the result of TSA and TSADPI will be the same for any general dataset. This is because TSADPI follows the same TWO-Scan procedure like TSA. The first scan is for candidate k-dominant skyline computation and the second scan is to check the false positive cases. However, it could avoid successfully many unnecessary comparison between the candidate k-dominant skyline and the dataset.

V. PERFORMANCE EVALUATION

We conduct a series of experiments to evaluate the effectiveness and efficiency of the proposed method using both real and synthetic datasets. In this paper, we compare our proposed approach for k-dominant skyline against TSA, which is the most efficient k-dominant skyline search algorithm proposed in Ref. 6). The efficiency of the TSA approach is dependent on how effective are the k-dominant objects pruning non-dominant skyline points during the first scan. If the number of false positives produced by the first scan is small, then the performance of the second scan and hence the overall approach will be good.

The proposed algorithm works particularly well if the k-dominant skyline is small. In the best case, the algorithm terminates in one iterations; thus, the best case complexity is of the order of $O(n)$; n being the total number of objects in dataset. In the worst case, the complexity is of the order of $O(n^2)$. On the other hand, TSA takes O(n^2) to compute k-dominant skyline objects. Though in the worst case the time complexity of our

TABLE III.
FIRST SCAN COMPARISON BETWEEN TSA AND TSADPI

TSA			TSADPI		
DB sorted by sum	$Sky_k(DB)$	Co	DB sorted by DP	$Sky_k(DB)$	Co
$O_5, O_7, O_2, O_4, O_3, O_6, O_1$	$\emptyset$	0	$O_7, O_5, O_4, O_2, O_6, O_3, O_1$	$\emptyset$	0
$O_7, O_2, O_4, O_3, O_6, O_1$	O_5	0	$O_5, O_4, O_2, O_6, O_3, O_1$	O_7	0
O_2, O_4, O_3, O_6, O_1	O_5, O_7	1	O_4, O_2, O_6, O_3, O_1	O_7, O_5	1
O_4, O_3, O_6, O_1	O_5, O_7	2	O_2, O_6, O_3, O_1	O_7, O_5	1
O_3, O_6, O_1	O_5, O_7	2	O_6, O_3, O_1	O_7, O_5	1
O_6, O_1	O_5, O_7	1	O_3, O_1	O_7, O_5	1
O_1	O_5, O_7	2	O_1	O_7, O_5	1
$\emptyset$	O_5, O_7	2	$\emptyset$	O_7, O_5	1

TABLE IV.
COMPARISONS REDUCTION

Data Size(k)	Anti-Correlated			Correlated			Independent		
	#Comp. by TSA(k)	#Comp. by TSADPI(k)	RR (%)	#Comp. by TSA(k)	#Comp. by TSADPI(k)	RR (%)	#Comp. by TSA(k)	#Comp. by TSADPI(k)	RR (%)
100	143	86	40	11	7	34	113	71	37
200	287	175	39	23	13	43	239	167	31
300	413	265	36	31	17	45	334	239	29
400	580	358	38	45	27	40	406	304	25
500	691	428	38	62	36	42	552	385	30

proposed method is substantially the same as TSA, we can drastically reduce comparisons for k-dominant skyline computation by the domination power. TSADPI has the tendency to fit into the best case complexity. Results of all experiments support our claim that we can reduce the number of comparisons drastically. Domination power computation does not incur any additional cost in compare with TSA. This is because TSA also sorted the whole dataset in ascending order. However, for all experiments we have included the time cost of domination power.

A. Performance on Synthetic Datasets

As benchmark synthetic datasets, we use the datasets proposed in Ref. 2). Objects are generated using one of the following three value distributions:

Anti-Correlated: an anti-correlated dataset represents an environment in which, if an object has a small coordinate on some dimension, it tends to have a large coordinate on at least another dimension. As a result, the total number of non-dominating objects of an anti-correlated dataset is typically quite large.

Correlated: a correlated dataset represents an environment in which objects with large coordinates in one dimension are also have large coordinates in the other dimensions. In a correlated dataset, few objects dominate many other objects.

Independent: for this type of dataset, all attribute values are generated independently using uniform distribution. Under this distribution, the total number of non-dominating objects is between that of the correlated and the anti-correlated datasets.

Details of the three distributions can be found in Ref. 2). To study the potential advantages of δ-domination power on sort by sum, we evaluate comparisons of TSADPI against TSA and compute the reduction rate. The reduction rate is defined as

$$ReductionRate, (RR) = (1-\frac{Comp.byTSADPI}{Comp.byTSA})\times 100; \quad (1)$$

where Comp. by TSADPI and Comp. by TSA are the summation of all pairwise comparisons to compute k-dominant skyline by TSADPI and TSA respectively. We set n to 7, k to 6, and vary data cardinality from 100k to 500k. From Table IV, we notice that the number of comparisons of TSADPI is smaller than that of TSA and the reduction rate varies from 25% to 45%.

In the following sections, we examined the effect of dimensionality and cardinality.

Effect of Dimensionality. For this experiment, we vary dataset dimensionality n ranges from 10 to 20 and k from 6 to 19. Figure 3(a), (b), and (c) represents the effect of dimensionality. For all distributions the response time of the proposed method is better than TSA approach and it increases if the data dimensionality n increases.

Effect of Cardinality. For this experiment, we vary dataset cardinality ranges from 100k to 500k and set the values of n to 15 and k to 13. Figure 2(a), (b), and (c) shows that when the size of the dataset increases from

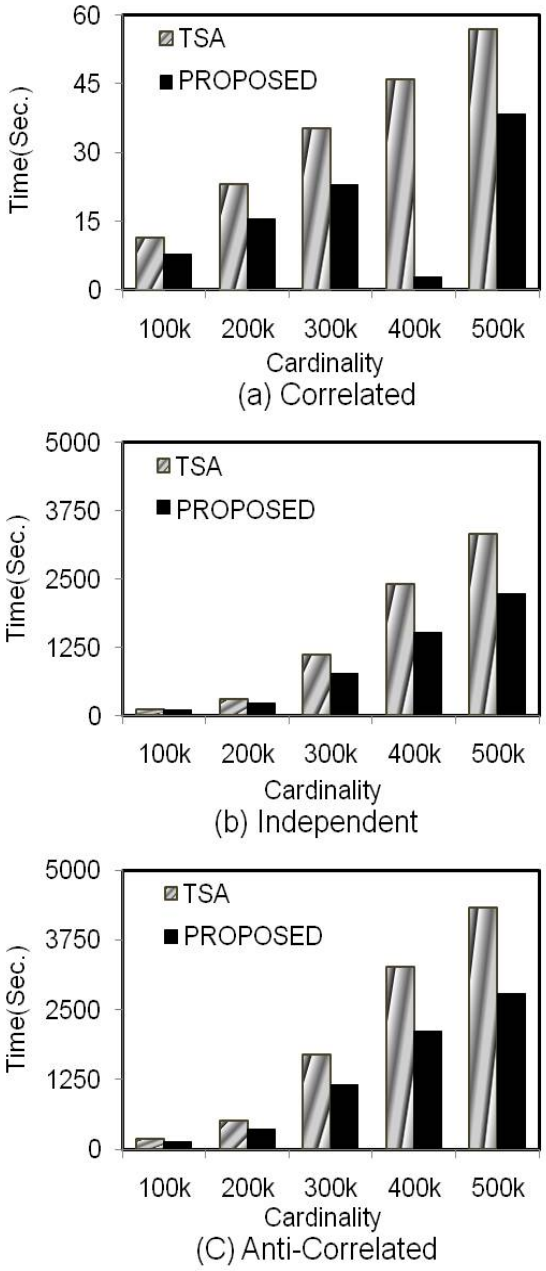


Figure 2. k-dom. skyline computation for different datasize

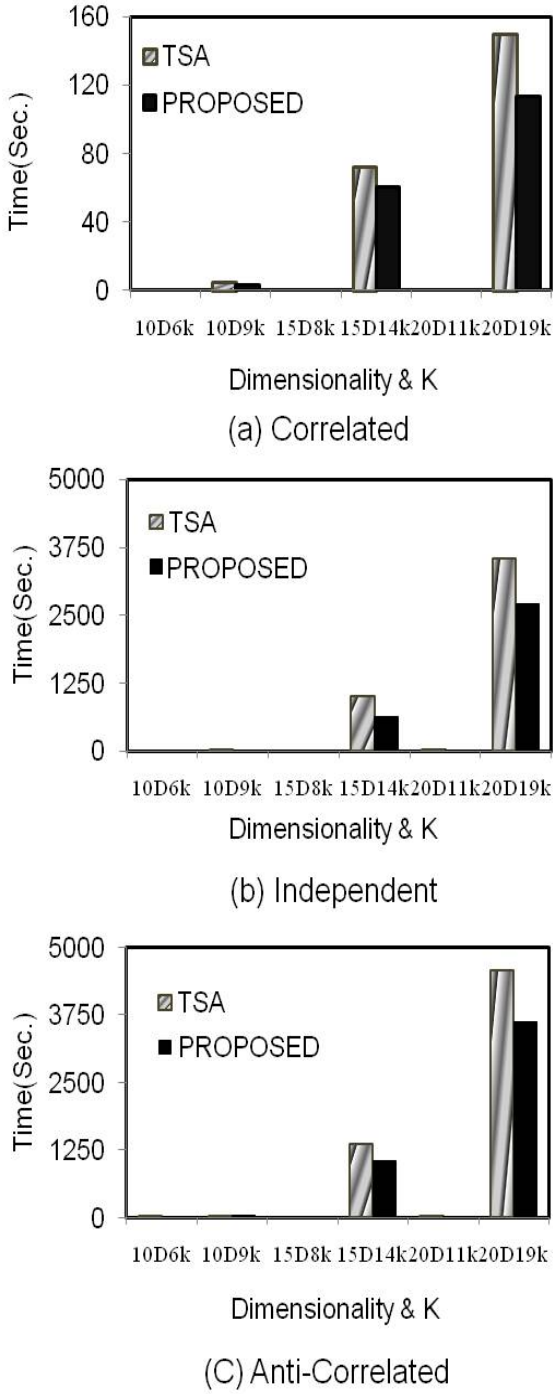


Figure 3. k-dom. skyline computation for different dimension

100k to 500k, the computation time of both algorithms maintain a positive correlation. Notice that our TSADP performs better than TSA.

B. Performance on Real Datasets

To evaluate the performance for real dataset, we study two different real datasets. The first dataset is NBA statistics. It is extracted from "`www.nba.com`". The dataset contains 17k 13-dimensional data objects, which correspond to the statistics of an NBA players' performance in 13 aspects (such as points scored, rebounds, assists, etc.). The second dataset is FUEL dataset and extracted from "`www.fueleconomy.gov`". FUEL dataset is 24k 6-dimensional objects, in which each object stands for the performance of a vehicle (such as mileage per gallon of gasoline in city and highway, etc.). Using both datasets we conduct the following experiment.

Experiments on Real Dataset for k-dominant Skyline. We performed two experiments on NBA dataset. In the first experiment, we study the effect of dimensionality when n varies from 5 to 13 and k from 4 to 12. Figure 4(a) shows the result. NBA dataset exhibits similar result to synthetic dataset, if the number of dimension increases the performance of both algorithms becomes slower. Figure 4(a) represents that proposed method is faster than TSA.

For FUEL dataset, we performed similar experiment

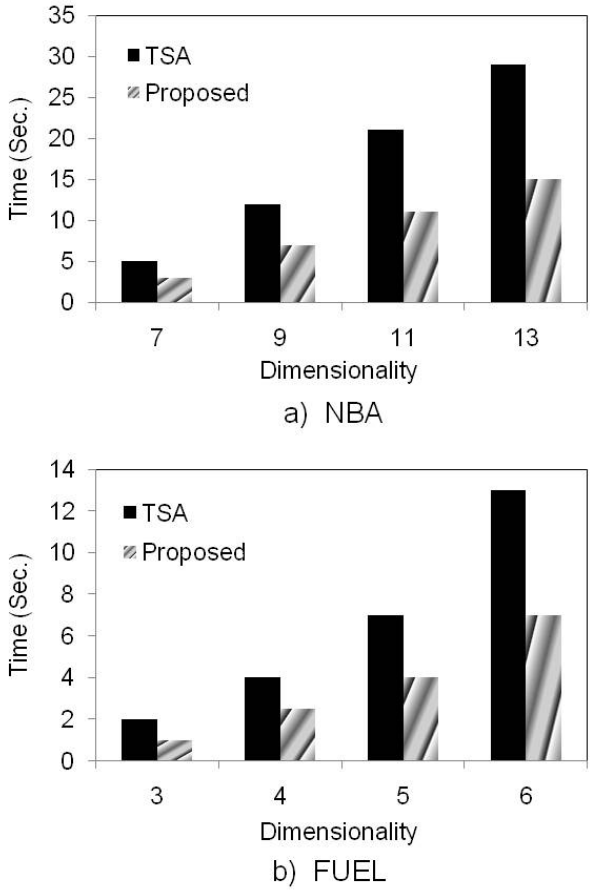


Figure 4. k-dominant experiments on NBA and FUEL dataset

like NBA dataset. For this experiment, n varies from 3 to 6 and k varies from 2 to 5. Result is shown in Figure 4(b). For this experiment with FUEL dataset, we obtain similar result like NBA dataset that represents the scalability of the proposed method.

VI. Conclusion

In this paper, we consider k-dominant skyline query problem and present a method for computing the query result for a particular k at any time. By applying domination power strategy, we reduce huge number of comparisons between the k-dominant skyline objects and the dataset. Our comprehensive performance study using real and synthetic datasets demonstrate that the proposed method is very efficient and scalable.

However, proposed method is efficient to compute k-dominant skyline only for static datasets. It may not be efficient for frequently updated datasets. The study to facilitate incremental updates dataset is let for future work.

Acknowledgment

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

References

[1] T. Xia, D. Zhang, and Y. Tao, "On Skylining with Flexible Dominance Relation," in Proceedings of ICDE. Cancun, pp. 1397-1399, April 2008.

[2] S. Borzsonyi, D. Kossmann, and K. Stocker, "The skyline operator," in Proceedings of ICDE. Germany, pp. 421-430, April 2001.

[3] D. Kossmann, F. Ramsak, and S. Rost, "Shooting stars in the sky: An online algorithm for skyline queries," in Proceedings of VLDB. China, pp. 275-286, August 2002.

[4] J. Chomicki, P. Godfrey, J. Gryz, and D. Liang, "Skyline with Presorting," in Proceedings of ICDE. India, pp. 717-719, March 2003.

[5] D. Papadias, Y. Tao, G. Fu, and B. Seeger, "Progressive skyline computation in database systems," ACM Transactions on Database Systems, vol. 30(1), pp. 41-82, March 2005.

[6] C. Y. Chan, H. V. Jagadish, K-L. Tan, A-K. H. Tung, and Z. Zhang, "Finding k-Dominant Skyline in High Dimensional Space," in Proceedings of ACM SIGMOD. USA, pp. 503-514, June 2006.

[7] K.-L. Tan, P.-K. Eng, and B. C. Ooi, "Efficient Progressive Skyline Computation," in Proceedings of VLDB. Italy, pp. 301-310, September 2001.

[8] R. Fagin, A. Lotem, and M. Naor, "Optimal aggregation algorithms for middleware," in Journal of Computer and System Sciences. USA, vol. 66, pp. 614-656, April 2003.

[9] A. Vlachou, C. Doulkeridis, Y. Kotidis, and M. Vazirgiannis, "SKYPEER: Efficient Subspace Skyline Computation over Distributed Data," in Proceedings of ICDE. Turkey, pp. 416-425, April 2007.

[10] K. Fotiadou, and E. Pitoura, "BITPEER: Continuous Subspace Skyline Computation with Distributed Bitmap Indexes," in Proceedings of DaMaP. USA, pp. 35-42, March 2008.

[11] C. Y. Chan, H. V. Jagadish, K-L. Tan, A-K. H. Tung, and Z. Zhang, "On High Dimensional Skylines," in Proceedings of EDBT. Germany, pp. 478-495, March 2006.

[12] Y. Tao, X. Xiao, and J. Pei, "Subsky: Efficient Computation of Skylines in Subspaces," in Proceedings of ICDE. USA, pp. 65-65, April 2006.

[13] C.-Y. Chan, P.-K. Eng, and K.-L. Tan, "Stratified Computation of Skylines with Partially-Ordered Domains," in Proceedings of ACM SIGMOD. USA, pp. 203-214, June 2005.

[14] M. Morse, J. M. Patel, and H. V. Jagadish, "Efficient Skyline Computation over Low-Cardinality Do-

mains," in Proceedings of VLDB. Austria, pp. 267-278, September 2007.

[15] W. T. Balke, U. Guntzer, and J. X. Zheng, "Efficient distributed skylining for web information systems," in Proceedings of EDBT. Greece, pp. 256-273, March 2004.

[16] S. Kapoor, "Dynamic Maintenance of Maxima of 2-d Point Sets", in SIAM Journal on Computing, vol. 29(6), pp. 1858-1877, April 2000.

[17] K. Deng, X. Zhou, and H. T. Shen, "Multi-source Skyline Query Processing in Road Networks," in Proceedings of ICDE. Turkey, pp. 796-805, April 2007.

[18] M. Sharifzadeh and C. Shahabi, "The Spatial Skyline Query," in Proceedings of VLDB. Korea, pp. 751-762, September 2006.

[19] C. Li, B. C. Ooi, A-K. H. Tung, and S. Wang, "DADA: A Data Cube for Dominant Relationship Analysis," in Proceedings of ACM SIGMOD. USA, pp. 659-670, June 2006.

[20] J. Pei, B. Jiang, X. Lin, and Y. Yuan, "Probabilistic Skylines on Uncertain Data," in Proceedings of VLDB. Austria, pp. 15-26, September 2007.

[21] E. Dellis and B. Seeger, "Efficient Computation of Reverse Skyline Queries," in Proceedings of VLDB. Austria, pp. 291-302, September 2007.

[22] M. A. Siddique and Y. Morimoto, "Extended k-dominant Skyline in High Dimensional Space," in Proceedings of ICISA. Korea, Vol. 2, pp. 520-527, April 2010.

Md. Anisuzzaman Siddique is an Associate Professor at University of Rajshahi (RU), Bangladesh. He received Ph.D. degree from Hiroshima University, Japan. He completed the B.Sc. and M.Sc. degrees in Computer Science and Technology from RU, Bangladesh in 2000 and 2002, respectively. Since 2002-present he is a faculty member in RU. His research interests include skyline evaluation, data mining, and privacy preserving information retrieval.

Yasuhiko Morimoto is an Associate Professor at Hiroshima University, Japan. He received B.E., M.E., and Ph.D. from Hiroshima University in 1989, 1991, and 2002, respectively. From 1991 to 2002, he had been with IBM Tokyo Research Laboratory where he worked for data mining project and multimedia database project. Since 2002, he has been with Hiroshima University. His current research interests include data mining, machine learning, geographic information system, and privacy preserving information retrieval.