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.

<table>
<thead>
<tr>
<th>ID</th>
<th>Return</th>
<th>Risk</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>B</td>
<td>7</td>
<td>6</td>
</tr>
<tr>
<td>C</td>
<td>9</td>
<td>8</td>
</tr>
<tr>
<td>D</td>
<td>8</td>
<td>5</td>
</tr>
<tr>
<td>E</td>
<td>6</td>
<td>6</td>
</tr>
</tbody>
</table>

Figure 1. Skyline example

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 $O_j$ is better than or equal to $O_i$. 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 number of efficient algorithms for computing skyline objects have been reported in the literature [1]–[5], [11], [13], [14].
is better in each dimension. Conventional skyline query for this dataset returns five objects: \(O_2, O_3, O_5, O_6, \text{ and } O_7\). 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 \(D_2\). \(O_4\) 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: \(O_2\) and \(O_7\). Objects \(O_3, O_5, O_6, \text{ and } O_4\) are not in \(5\)-dominant skyline because they are \(5\)-dominated by \(O_2\). \(4\)-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

Borczsonyi, et al. first introduced the skyline operator over large datasets and proposed three algorithms: Block-Nested-Loops (BNL), Divide-and-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-Filter-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-and-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-

<table>
<thead>
<tr>
<th>Obj</th>
<th>(D_1)</th>
<th>(D_2)</th>
<th>(D_3)</th>
<th>(D_4)</th>
<th>(D_5)</th>
<th>(D_6)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(O_1)</td>
<td>7</td>
<td>3</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>(O_2)</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>(O_3)</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>(O_4)</td>
<td>5</td>
<td>3</td>
<td>5</td>
<td>4</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>(O_5)</td>
<td>1</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>(O_6)</td>
<td>5</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>(O_7)</td>
<td>1</td>
<td>2</td>
<td>5</td>
<td>3</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>
Algorithm (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 \( D_1, D_2, \ldots, D_n \) be the \( n \) attributes of \( DB \). Let \( O_1, O_2, \ldots, O_r \) 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 \( O_i \preceq O_j \), if \( O_i.D_s \preceq O_j.D_s \) for all dimensions \( s = 1, \ldots, n \) 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 \( O_i \preceq_k O_j \), if \( O_i.D_s \preceq_k O_j.D_s \) 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 \( \delta \)-domination power if there are \( \delta \) 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., \( Sky_k(DB) \subseteq Sky(DB) \).

**Proof:** Let \( O_i \in Sky_k(DB) \) and \( O_i \notin Sky(DB) \).

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

IV. TWO SCAN ALGORITHM WITH DOMINATION POWER INDEX (TSA\( DPI \))

We use a filter based algorithm to compute \( Sk(y_k(DB)) \) 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 \( 5 \)-dominate 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 \( \delta \)-domination power if there are \( \delta \) 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 \( \delta \). 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
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 \( DB, Sky_k(DB) \) and no. of pairwise comparisons respectively during the first scan of TSA and column 4, 5, and 6 are those of TSADPI. Initially \( Sky_k(DB) \) is empty and \( O_7 \) is added in \( Sky_k(DB) \) as a 5-dominant object without any comparison. Since \( O_5 \) and \( O_7 \) fails to become 5-dominant each other, after one comparison \( O_7 \) is added in \( Sky_k(DB) \). Next after two comparisons we can see that \( O_2 \) is not in \( Sky_k(DB) \). 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 \( Sky_k(DB) = \{O_5, O_7\} \). From Table III we can see that TSADPI method needs only 6 comparisons while TSA needs 10.

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 \( Sky_k(DB) = \{O_7, O_5\} \). 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.

### Algorithm 1: Compute Domination Power, DP

1. for each object \( O_i (i = 1 \ldots n) \) do
2. \( O, DP := 0 \) (initialize DP for each object)
3. for each attribute \( D_k (s = 1 \ldots n) \) do
4. \( minValue := O_i.D_k \)
5. for each object \( O_i (i = 2 \ldots r) \) do
6. if \((O_i.D_k < minValue)\) then
7. \( minValue := O_i.D_k \)
8. for each object \( O_i (i = 1 \ldots r) \) do
9. if \((minValue == O_i.D_k)\) then
10. \( O_i.DP := \) \( O_i.D_k \)
11. Sort dataset, \( DB \), in descending order by DP and Sum

### Algorithm 2: TSADPI (DB, k)

1. Sort DB by domination power and sum
2. Initialize \( Sky_k(DB) = \emptyset \)
3. for each object \( O \in DB \) do
4. Compute DP
5. \( isDominant = true \)
6. for each object \( O' \in Sky_k(DB) \) do
7. if \((O' \text{ -dominates } O \text{ or } O \text{ -dominates } O')\) then
8. \( isDominant = false \)
9. break
10. Remove \( O' \) from \( Sky_k(DB) \)
11. if \((isDominant)\) then
12. Insert \( O \) into \( Sky_k(DB) \)
13. for each object \( O \in DB \) do
14. if \((O \text{ -dominates } O' \text{ or } O' \text{ -dominates } O')\) and \( O' \neq O \) then
15. Remove \( O' \) from \( Sky_k(DB) \)
16. Return \( Sky_k(DB) \)

© 2012 ACADEMY PUBLISHER
Correlated: a 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 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 $\delta$-domination power on sort by sum, we evaluate comparisons of TSADPI against TSA and compute the reduction rate. The reduction rate is defined as

$$\text{Reduction Rate (RR)} = (1 - \frac{\text{Comp. by TSADPI}}{\text{Comp. by TSA}}) \times 100$$

where Comp. by TSADPI and Comp. by TSA are the summation of all pairwise comparisons to compute $\delta$-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

<table>
<thead>
<tr>
<th>Data Size(k)</th>
<th>Anti-Correlated</th>
<th>Correlated</th>
<th>Independent</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>#Comp. by TSA(k)</td>
<td>#Comp. by TSADPI(k)</td>
<td>RR (%)</td>
</tr>
<tr>
<td>100</td>
<td>143</td>
<td>86</td>
<td>40</td>
</tr>
<tr>
<td>200</td>
<td>287</td>
<td>175</td>
<td>39</td>
</tr>
<tr>
<td>300</td>
<td>413</td>
<td>265</td>
<td>36</td>
</tr>
<tr>
<td>400</td>
<td>580</td>
<td>358</td>
<td>38</td>
</tr>
<tr>
<td>500</td>
<td>691</td>
<td>428</td>
<td>38</td>
</tr>
</tbody>
</table>
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...
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


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


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.