k-dominant and Extended *k*-dominant Skyline Computation by Using Statistics

Md. Anisuzzaman Siddique
Graduate School of Engineering
Hiroshima University
Higashi-Hiroshima, Japan
anis_cst@yahoo.com

Yasuhiko Morimoto
Graduate School of Engineering
Hiroshima University
Higashi-Hiroshima, Japan
morimoto@mis.hiroshima-u.ac.jp

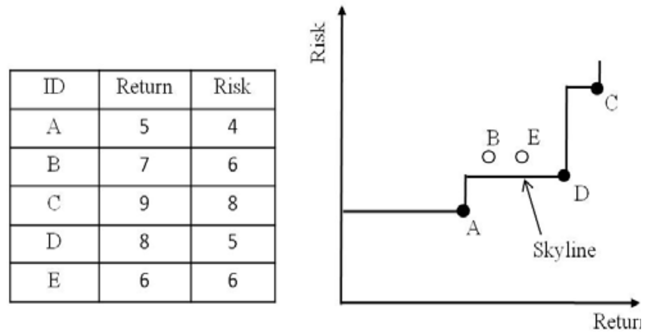
***Abstract*—Skyline queries have recently attracted a lot of attention for its intuitive query formulation. It can act as a filter to discard sub-optimal objects. However, a major drawback of skyline is that, in datasets with many dimensions, the number of skyline objects becomes large and no longer offer any interesting insights. To solve the problem, *k*-dominant skyline queries have been introduced, which can reduce the number of skyline objects by relaxing the definition of the dominance. However, sometimes, a *k*-dominant skyline query may retrieve too few objects to analyze. This paper addresses the problem of *k*-dominant skyline for high dimensional dataset. In addition, we extend the notion of *k*-domination by defining extended *k*-dominant skyline, which retrieves neither too many nor too few objects. We propose algorithms for *k*-dominant and extended *k*-dominant skyline computation. An extensive performance evaluation using both real and synthetic datasets demonstrated that our proposed methods are efficient and scalable.**

Keywords: Skyline, *k*-dominant Skyline, Extended *k*-dominant Skyline, Databases**.**

I. INTRODUCTION

Skyline queries are useful to multi-criteria decision making such as customer information systems, decision support, data visualization, and so forth. It represents the set of all solutions that are not worse than any objects. It can act as a filter to discard sub-optimal objects. The user can then interactively look at the (smaller) set of skyline objects and further select the ones that fit his/her needs.

Given an n-dimensional dataset 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. An investor chooses a stock from the list according to her/his preference. In this situation, her/his 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,



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

(a) Stocks (b) Skyline

Fig. 1. Skyline example

D and A are incomparable since a long-term investor may be willing to get lower return to ensure lower risk. A number of efficient algorithms for computing skyline objects have been reported in the literature [1], [2], [3], [4], [5].

It is always assumed that all of the attributes are involved in the skyline queries, that is, the dominating relationship is evaluated based on every dimensions 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 return and risk, 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 deal with this dimensionality curse, one possibility is to reduce the number of dimensions considered. However, which dimensions to retain is not easy to determine, and at the very least requires intimate knowledge of the application domain. To reduce the number of dimensions without any intimate knowledge of the application domain, Chan, *et al.* considered *k*-dominant skyline query [6]. They relaxed the definition of "dominated" so that an object is 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. In contrast, conventional skyline objects are n-dominant objects. However, sometimes, a k-dominant skyline query may retrieves too few objects to analyze. To solve this problem in this paper, we extend the notion of k-domination by defining extended k-dominant skyline, which retrieves neither too many nor too few objects.

TABLE I
SYMBOLIC DATASET

Object	D_1	D_2	D_3	D_4	D_5	D_6
O_1	7	7	4	5	4	2
O_2	7	6	5	6	3	2
O_3	2	2	3	4	5	6
O_4	3	3	4	5	1	2
O_5	4	4	1	2	6	3
O_6	2	5	2	4	3	4
O_7	3	2	2	4	6	5
O_8	1	1	6	3	1	1
O_9	5	6	4	3	3	1
O_{10}	2	6	4	5	2	3
O_{11}	1	2	6	5	4	4
O_{12}	3	6	5	2	4	5

An object O_i is said to extended k-dominates another object O_j $(i \neq j)$ if there are k $(k \leq n)$ dimensions in which O_i is better than O_j. An extended k-dominant skyline object is an object that is not extended 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 attributes or dimensions from D_1 to D_6. Without loss of generality, we assume smaller value is better in each dimension. Conventional skyline query for this dataset returns objects: from O_3 to O_{10} and O_{12}. Objects O_1 and O_2 are not in skyline because they are dominated by O_4. Similarly, object O_{11} is not in skyline because it is dominated by O_8. Thus, skyline query for this dataset returns nine out of twelve objects (too large). The k-dominant skyline query can control the selectivity by changing k. Consider the case where $k = 5$, then the 5-dominant skyline query for this dataset returns objects O_8 and O_5. Other objects are not in 5-dominant skyline because they are 5-dominated by O_8. Similarly, 4-dominant skyline query (i.e., $k = 4$) returns only one object, O_8 is in 4-dominant skyline. If we decrease the value of k by one, then the 3-dominant skyline will retrieve empty result. Thus we see that k-dominant skyline query for this dataset returns at most two out of twelve objects (too few). Again, if we apply proposed extended 5-dominant skyline query for this dataset, then it will retrieves O_4, O_5, O_8, and O_9. That means extended k-dominant skyline query for this dataset retrieves four objects out of twelve (neither too large nor too few).

In this paper, we propose efficient algorithms to compute k-dominant as well as extended k-dominant skyline. The remainder of this paper is organized as follows. Section II discusses related work. Section III presents the notions and properties of k-dominant skyline computation and its extension. We provide detailed examples and analysis of our algorithms in Section IV. We experimentally evaluate proposed algorithms in Section V under a variety of settings. Finally, Section VI concludes the paper.

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 introduce 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, 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 [8]. Kapoor studies the problem of dynamically maintaining an effective data structure for an incremental skyline computation in a 2-dimensional space [9]. Tao and Papadias studied sliding window skylines, focusing on data streaming environments [10]. Huang, *et al.* studied continuous skyline queries for dynamic datasets [11].

B. k-dominant Skyline Query Processing

Chan, *et al.* introduce k-dominant skyline query [6]. They proposed three algorithms, namely, One-Scan Algorithm (OSA), Two-Scan Algorithm (TSA), and Sorted Retrieval Algorithm (SRA). OSA uses the property that a k-dominant skyline objects 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 [12].

Another study on computing k-dominant skyline is *k-Z Search* proposed by Lee, *et al.* [13]. They introduced a concept called filter-and-reexamine approach. In the filtering phase, it removes all k-dominant objects and retain possible skyline candidates, which may contain false hits. In the re-examination phase, all candidates are reexamined to eliminate false hits.

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 [14]. Fotiadou, *et al.* mention about the efficient computation of extended skylines using bitmaps in [15]. Chan, *et al.* introduce the concept of *skyline frequency* to facilitate skyline retrieval in high-dimensional spaces [16]. Tao, *et al.* discuss skyline queries in arbitrary subspaces [17]. There exist more work addressing *spatial skyline* [18], [19], skylines on partially-ordered attributes [20], *dada cube* for analysis of dominance relationships [21], *probabilistic skyline* [22], skyline search over small domains [23], and *reverse skyline* [24].

III. PRELIMINARIES

This section discusses the k-dominant skyline problems and associated properties. 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 of DB. We use $O_i.D_j$ to denote the j-th dimension value of O_i.

k-dominance

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

By contrast, an object O_i is said to *k-dominate* another object O_j, denoted as $O_i \preceq_k O_j$, if $O_i.D_s \leq O_j.D_s$ 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.

k-dominant Skyline

An object $O_i \in DB$ is said to be a *skyline object* of DB if O_i is not dominated by any other object in DB. Similarly, 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 $Sky_k(DB)$. Note that objects that have k-domination power must be k-dominant skyline objects but not vice versa.

Extended k-dominant Skyline

An object O_i is said to *extended k-dominate* another object O_j, denoted as $O_i <_{ext\text{-}k} O_j$, if $O_i.D_s < O_j.D_s$ in k dimensions among n dimensions. We call such O_i as *extended k-dominant object* and such O_j as *extended k-dominated object* between O_i and O_j.

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

Theorem 1: *Every object that belongs to the k-dominant skyline also belongs to the extended k-dominant skyline, i.e.,* $Sky_k(DB) \subseteq Sky_{ext-k}(DB)$.

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

The following theorem 2 [6] shows that there may not be any k-dominant skyline object in a dataset for any $k < n$.

Theorem 2: *For any* $k < n$ $(k \geq 2)$ *and a n-dimensional space, there exist a dataset* DB^I *with size* $|DB^I| \geq n$ *such that* $Sky_k(DB^I) = \emptyset$.

Table II shows an example dataset that exhibits the cyclic dominance relationship when $k = 3$. Specifically, we have O_i 3-dominates O_{i+1}, $\forall i \in [1, 3]$, and O_4 in turn 3-dominates O_1. In any cyclic dominance relationship, if two objects have same domain value in any attribute. Then according to extended k-dominant skyline definition, the cyclic dominance does not no longer exist. From Table II we see that object O_4 fails to become the extended k-dominant of object O_1. Thus, the extended 3-dominant query on Table II will retrieve O_1 as a result.

IV. ALGORITHMS

In this section, we present our algorithms for computing k-dominant and extended k-dominant skyline in n-dimensional dataset DB. We use filter based technique to compute $Sky_k(DB)$ and $Sky_{ext-k}(DB)$, efficiently. For both types is discusses in Section IV-A. We discuss about k-dominant skyline and extended k-dominant skyline computation in Section IV-B and Section IV-C, respectively.

A. Domination Power Calculation

Chan, *et al.* sort the whole tuples (objects) with a monotonic scoring function sum in their OSA algorithm for k-dominant query [6]. By using the ordered tuples, we can eliminate some of non-skyline objects easily. However, this ordered tuples 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.

TABLE II
CYCLIC DOMINANCE RELATIONSHIP

Object	D_1	D_2	D_3	D_4
O_1	6	6	6	6
O_2	2	7	7	7
O_3	3	2	8	8
O_4	4	3	6	9

TABLE III
SORTED DATASET

Object	D_1	D_2	D_3	D_4	D_5	D_6	*DP*	*Sum*
O_8	1	1	6	3	1	1	4	13
O_5	4	4	1	2	6	3	2	20
O_4	3	3	4	5	1	2	1	18
O_9	5	6	4	3	3	1	1	22
O_{11}	1	2	6	5	4	4	1	22
O_{12}	3	6	5	2	4	5	1	25
O_6	2	5	2	4	3	4	0	20
O_3	2	2	3	4	5	6	0	22
O_{10}	2	6	4	5	2	3	0	22
O_7	3	2	2	4	6	5	0	23
O_1	7	7	4	5	4	2	0	29
O_2	7	6	5	6	3	2	0	29

In order to prune unnecessary objects efficiently in the k-dominant skyline computation, we consider a new type statistics of each object which we called *domination power*. 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 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. However, sometimes, lower object can k-dominate higher object. Experiments show that our estimation is robust over various distributions. Moreover, it also works well when data values are correlated, independent or anti-correlated. Algorithm 1 represents the domination power statistics calculation procedure.

Table III is the example of sorted dataset DB. In the sorted dataset, object O_8 has the highest domination power 4. Note that object O_8 dominates all objects lie below it in four attributes D_1, D_2, D_5, and D_6.

B. k-dominant Skyline Algorithm

To determine whether an object O is k-dominant or not, we need to compare it against k-dominant skyline objects as well as non-k-dominant skyline objects. This is because O can be k-dominated by any non-k-dominant 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 $Sky_k(DB)$. Similar as TSA we also need two scan for k-dominant skyline computation. But the dataset sorted by domination power can reduce many pairwise comparisons between the objects of $Sky_k(DB)$ and DB. Algorithm 2 shows our proposed algorithm for k-dominant skyline computation.

Algorithm 1: *Compute Domination Power, DP*

```
1. For each attribute $D_i (i = 1$ to $n)$ do
2.   Initialize minValue = 0
3.   For each object $O_j (j = 1$ to $r)$ do
4.     Initialize DP of object $O_j$, $O_j$.DP = 0
5.     If (minValue < $O_j.D_i$) then
6.       minValue = minValue
7.     Else
8.       minValue = $O_j.D_i$
9.   For each object $O_j (j = 1$ to $r)$ do
10.    If (minValue == $O_j.D_i$) then
11.      Increase DP of $O_j$, $O_j$.DP by 1
12.  Sort dataset, $DB$ in ascending order by DP
13.        If (more than one objects have same DP) then
14.        Sort those objects in ascending order of SUM
13.  Return the sorted $DB$
```

During the k-dominant evaluation initially $Sky_5(DB)$ is empty and O_8 is added in $Sky_5(DB)$ as a 5-dominant object without any comparison. Since O_8 and O_5 fails to become 5-dominant of each other, after one comparison O_5 is added in $Sky_5(DB)$. Next, after compare with O_8 we see that O_4 is not in $Sky_5(DB)$. In this way objects O_9, O_{11}, O_{12}, O_6, O_3, O_{10}, O_7, O_1, and O_2 are not in $Sky_5(DB)$. At the end of the first scan $Sky_5(DB)=\{O_8, O_5\}$. After second scan we confirm that there exist no false positive object. Next, using $Sky_5(DB)$ objects, we get $Sky_4(DB)=\{O_8\}$.

Algorithm 2: k-dominant Skyline

```
1. Sort DB by domination power and sum
2. Initialize $Sky_k(DB) = 0$
3.   For each object $O \in DB$ do
4.     Initialize isDominant = $true$
5.     For each object $O^I \in Sky_k(DB)$ do
6.       If ($O^I \leq_n O$ or $O^I \leq_k O$) then
7.         isDominant = $false$
8.         break
9.       If ($O \leq_n O^I$ or $O \leq_k O^I$) then
10.        Remove $O^I$ from $Sky_k(DB)$
11.    If ($isDominant$) then
12.      insert $O$ into $Sky_k(DB)$
13.  For each object $O \in DB$ do
14.    For each object $O^I \in Sky_k(DB)$, do
15.      If (($O \leq_n O^I$ or $O \leq_k O^I$) and $O^I = O$) then
16.        Remove $O^I$ from $Sky_k(DB)$
17.  Return $Sky_k(DB)$
```

C. Extended k-dominant Skyline Algorithm

This section introduces how to calculate extended k-dominant skyline of sorted dataset DB. The algorithm (shown in algorithm 3) is based on the following key properties.

Lemma 1: *Consider an object $O \in DB$ that is not an extended k-dominant skyline object. Then it is possible for O not to be extended k-dominated by any extended k-dominant skyline object.*

Our algorithm takes as input a n-dimensional dataset DB and a parameter k, and outputs the set of extended k-dominant skyline objects in DB. To compute extended k-dominant skyline (i.e., $Sky_{ext-k}(DB)$) proposed method scan the dataset DB twice. In the first scan of DB (steps 1 to 12), a set of candidate extended k-dominant skyline objects, $Sky_{ext-k}(DB)$ is computed progressively by comparing each object $O \in DB$ against the computed objects in $Sky_{ext-k}(DB)$. If an object is extended k-dominated, then it is removed from $Sky_{ext-k}(DB)$. During this scan we ignore non-skyline objects that are all dominated (i.e., n-dominated) by $Sky_{ext-k}(DB)$ objects. This is because according to theorem 1 non-skyline object cannot become an extended k-dominant skyline. Although it has same domain value in some attributes compare with other objects in $Sky_{ext-k}(DB)$. Note that false positives can exist in $Sky_{ext-k}(DB)$ due to property in lemma 1.

To eliminate the false positives produced by the first scan, a second scan of DB (step 13 to 17) is necessary. During the second scan we can exclude all $Sky_{ext-k}(DB)$ as well as non-skyline objects for further extended k-dominant checking. The efficiency of our approach depends on the pruning capability of non-skyline objects 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.

Algorithm 3: Extended k-dominant Skyline

```
1. Sort DB by domination power and sum
2. Initialize $Sky_{ext-k}(DB) = 0$
3.   For each object $O \in DB$ do
4.     Initialize isDominant = $true$
5.     For each object $O^I \in Sky_{ext-k}(DB)$ do
6.       If ($O^I \leq_n O$ or $O^I <_k O$) then
7.         isDominant = $false$
8.         break
9.       If ($O \leq_n O^I$ or $O <_k O^I$) then
10.        Remove $O^I$ from $Sky_{ext-k}(DB)$
11.    If ($isDominant$) then
12.      Insert $O$ into $Sky_{ext-k}(DB)$
13.  For each object $O \in DB$ do
14.    For each object $O^I \in Sky_{ext-k}(DB)$, do
15.      If (($O \leq_n O^I$ or $O <_k O^I$) and $O^I = O$) then
16.  Remove
17. Return $Sky_{ext-k}(DB)$
```

Assume $k = 5$, then applying the proposed algorithm on

Table III, we note that at the end of first scan objects O_8, O_5, O_4, O_9, and O_{12} will be inserted into $Sky_{ext-5}(DB)$. This example demonstrates the effective pruning ability of the extended k-dominant skyline objects in eliminating non-extended k-dominant skyline objects. However, there exists a false positive object, O_{12}. During the second scan O_{12} is extended 5-dominated by O_6. That means after the completion of second scan our method give $Sky_{ext-5}(DB)$ = $\{O_8, O_5, O_4, O_9\}$ as a result. Next, using $Sky_{ext-5}(DB)$ objects, we get $Sky_{ext-4}(DB)$ = $\{O_8\}$.

TABLE IV
NO. OF RETURNED OBJECT

Size	Anti-Correlated			Correlated			Independent		
	Skyline Objects	k-dom. Objects	Ext. k-dom. Objects	Skyline Objects	k-dom. Objects	Ext. k-dom. Objects	Skyline Objects	k-dom. Objects	Ext. k-dom. Objects
100k	57686	16745	22896	7857	1254	2594	36651	10728	17411
200k	108150	33178	42698	15578	2542	4708	70516	23287	30540
300k	158570	47693	62500	23415	3849	5257	110782	32872	45156
400k	211558	69613	85392	30513	5047	8758	143165	48827	65045
500k	258507	87432	105454	38312	6265	10123	180156	58253	72854

V. PERFORM ANCE EVALUATION

We conduct a series of experiments to evaluate the effectiveness and efficiency of proposed methods. 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). On the other hand in lack of techniques dealing directly with the problem of extended k-dominant skyline, we cannot compare our proposed method against other methods. However, we conduct simulation experiments on a PC running on MS Windows XP professional. The PC has an Intel(R) Core2 Duo 2GHz CPU and 3GB main memory. All experiments were coded in Java J2SE V6.0. Each experiment is repeated five times and the average result is considered for performance evaluation.

A. Performance on Synthetic Datasets

As benchmark 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 coordinate in one dimension are also have large coordinate 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.

The generation of the synthetic datasets is controlled by three parameters, n, "Size", and "Dist", where n is the number of attributes, "Size" is the total number of objects in the dataset, and "Dist" can be the any of the three distributions.

Table IV shows the returned objects comparison between skyline, k-dominant skyline, and extended k-dominant skyline. For this experiment, we vary data cardinality from 100k to 500k, set n to 13, and k to 12. We use SFS method proposed in Ref. 4) to compute skyline objects and TSA method proposed in Ref. 6) to compute k-dominant skyline computation. Table IV shows that number of skyline objects for all distribution is much larger than that in the k-dominant skyline and extended k-dominant skyline. However, the returned objects set size of the extended k-dominant skyline is between that of the skyline and the k-dominant skyline.

In the following sections, we will examined the effect of cardinality and dimensionality. In each experiment, we evaluate total time to compute extended k-dominant skyline. Similar to most of the related work in the literature, we employ the *elapsed time* as the performance metric.

Effect of Cardinality

For k-dominant skyline experiment, we vary dataset cardinality ranges from 100k to 500k, 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 100k to 500k, the computation time of both algorithms maintain a positive correlation. Notice that our proposed method performs better than TSA.

For extended k-dominant skyline experiment, we vary dataset cardinality ranges from 100k to 500k, set the value of n to 13, and k to 12. Figure 3(a), (b), and (c) shows the time to compute extended k-dominant skyline. For all distributions, the time of proposed method is increases if the data cardinality increases. The result shows that it takes highest time for anti-correlated datasets then for independent datasets. This is because for anti-correlated distribution, if an object has a small coordinate on some dimension, it tends to have a large coordinate on other dimensions. As a result, the total number of non-dominating objects set size is large.

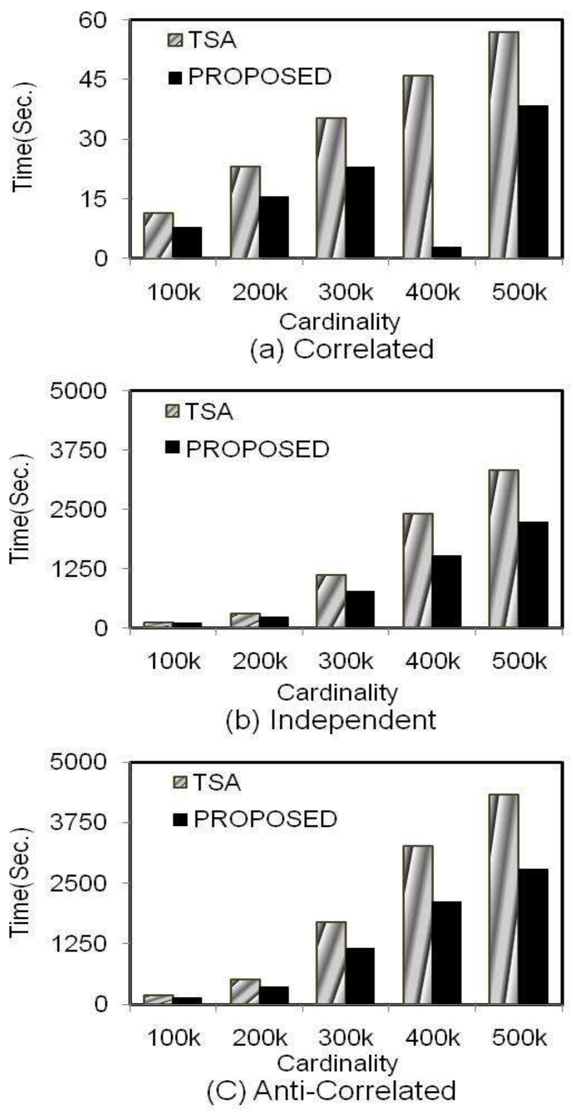


Fig. 2. *k*-dom. skyline computation for different datasize

Effect of Dimensionality

For *k*-dominant skyline experiment, we vary dataset dimensionality *n* ranges from 10 to 20 and *k* from 6 to 19. Figure 4(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.

Again for extended *k*-dominant skyline experiment, we vary dataset dimensionality *n* ranges from 7 to 13 and *k* from 6 to 12. Figure 5(a), (b), and (c) represents the effect of dimensionality. For all distributions, the response time of the proposed method is increases if the data dimensionality *n* increases. This is because by increasing the number of dimensions, the probability that an object dominates another one is reduced significantly.

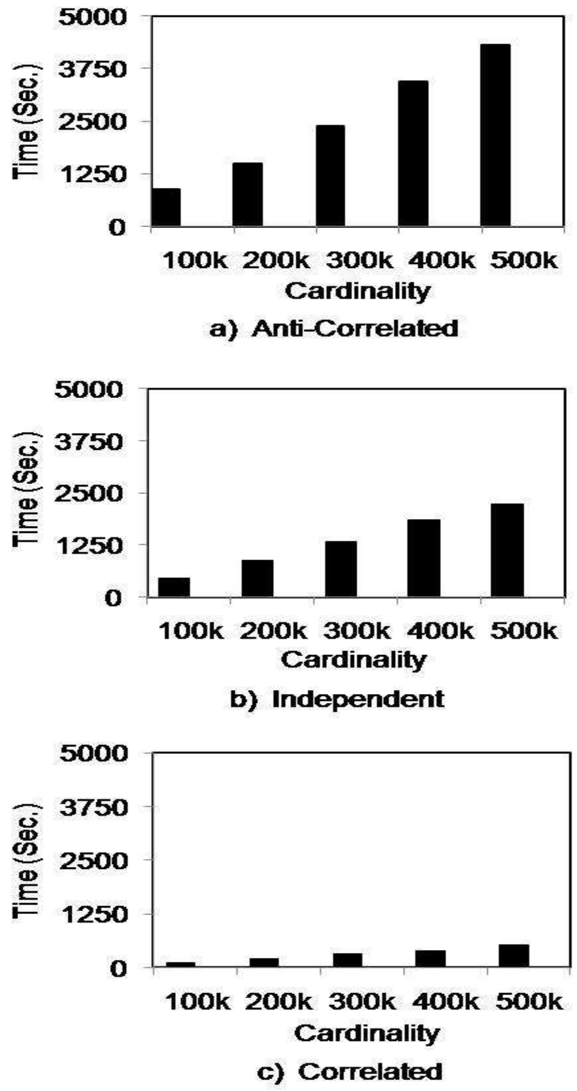


Fig. 3. Ext. *k*-dom. skyline computation for different datasize

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.) and domain have range [0, 4000]. The dataset approximates a correlated data distribution. 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.). For this dataset attribute domain range is [8, 89]. Using both datasets we conduct the following experiment.

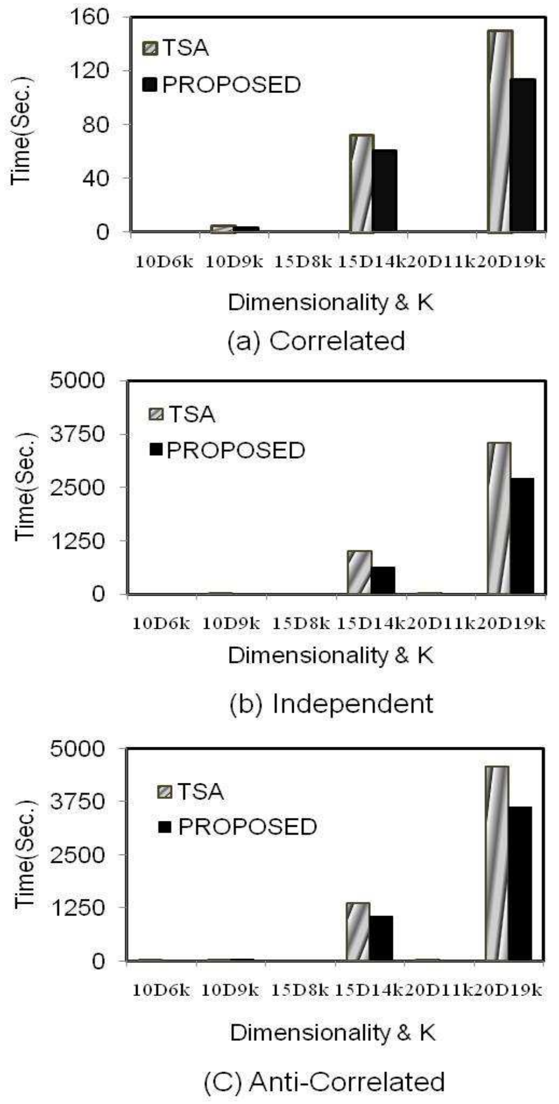


Fig. 4. *k*-dom. skyline computation for different dimension

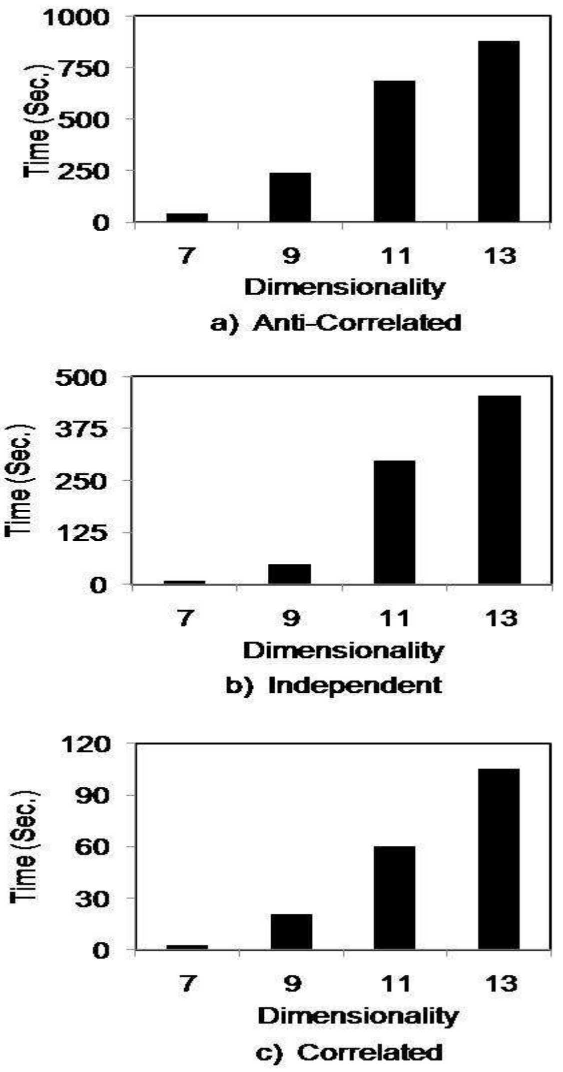


Fig. 5. Ext. *k*-dom. skyline computation for different dimension

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 6(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 6(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 6(b). For this experiment with FUEL dataset, we obtain similar result like NBA dataset that represents the scalability of the proposed method.

Experiments on Real Dataset for Extended k-dominant Skyline

We performed an experiment on NBA dataset. In this experiment, we study the effect of dimensionality when *n* varies from 7 to 13 and *k* from 6 to 12. Figure 7(a) shows the result. NBA dataset exhibits similar result to synthetic dataset, if the number of dimension increases the performance of proposed algorithm becomes slower.

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 7(b). For this experiment with FUEL dataset, we obtain similar result like NBA dataset that represents the scalability of the extended *k*-dominant skyline computation technique.

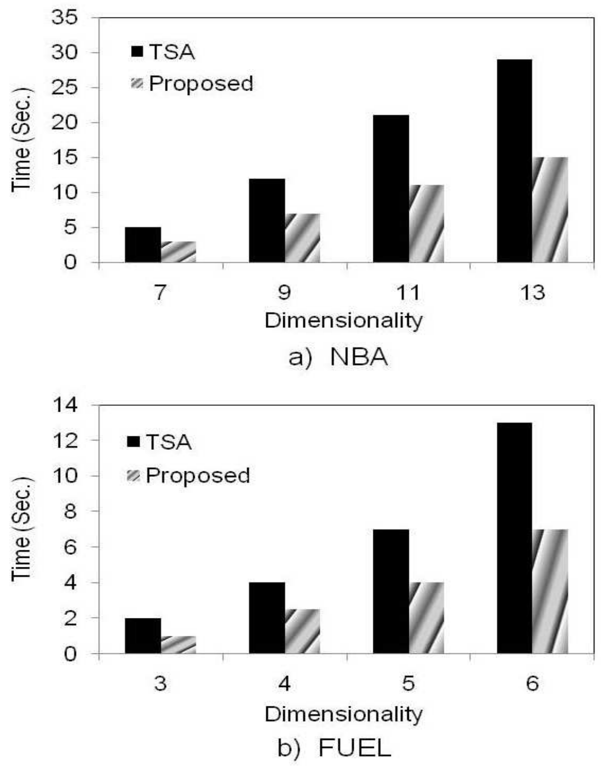


Fig. 6. *k*-dom. experiments on NBA and FUEL dataset

VI. CONCLUSION

In this paper, we consider *k*-dominant skyline query problem and present a method for computing the query result. By applying domination power statistics, we can compute the *k*-dominant skyline query result efficiently. Using real and synthetic datasets, we demonstrate the efficiency and scalability of our proposed method. However, a skyline has a side effect of retrieving too many objects and a *k*-dominant skyline query retrieves too few objects to analyze. To solve this problem, we consider extended *k*-dominant skyline query problem and present a method for computing the query result. Using real and synthetic datasets, we demonstrate the efficiency and scalability of our proposed method.

However, proposed methods performance are efficient to compute *k*-dominant and extended *k*-dominant skyline, but those methods are designed only for static datasets. They may not efficient for frequently updated datasets. Future works need to study techniques to facilitate incremental updates. Develop algorithms on *k*-dominant as well as extended *k*-dominant skyline integrating with ranking the usefulness of query results would be desirable.

ACKNOW LEDGEMENTS

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

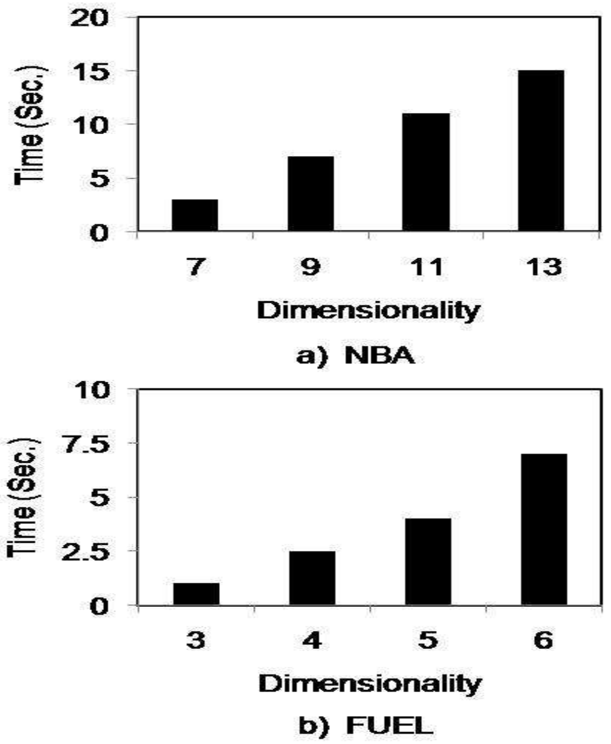


Fig. 7. Ext. *k*-dom. experiments on NBA and FUEL dataset

REFERENCES

[1] T. Xia, D. Zhang, and Y. Tao, “On Skylining with Flexible Dominance Relation”, in Proceedings of ICDE, 2008, pp. 1397-1399.
[2] S. Borzsonyi, D. Kossmann, and K. Stocker, “The skyline operator”, in Proceedings of ICDE, 2001, pp. 421-430.
[3] D. Kossmann, F. Ramsak, and S. Rost, “Shooting stars in the sky: An online algorithm for skyline queries”, in Proceedings of VLDB, 2002, pp. 275-286.
[4] J. Chomicki, P. Godfrey, J. Gryz, and D. Liang, “Skyline with Presorting”, in Proceedings of ICDE, 2003, pp. 717-719.
[5] D. Papadias, Y. Tao, G. Fu, and B. Seeger, “Progressive skyline computation in database systems”, in ACM Trans- actions 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, 2006, pp. 503-514.
[7] K.-L. Tan, P.-K. Eng, and B. C. Ooi, “Efficient Progressive Skyline Computation”, in Proceedings of VLDB, 2001, pp. 301-310.
[8] W. T. Balke, U. Guntzer, and J. X. Zheng, “Efficient distributed skylining for web information systems”, in Proceedings of EDBT, 2004, pp. 256-273.
[9] S. Kapoor, “Dynamic Maintenance of Maxima of 2-d Point Sets”, in SIAM Journal on Computing, vol. 29(6), pp. 1858-1877, April 2000.
[10] Y. Tao and D. Papadias, “Maintaining Sliding Window Skylines on Data Streams”, in IEEE Transactions on Knowledge and Data Engineering, vol. 18(3), pp. 377-391, March 2006.
[11] Z. Huang, H. Lu, B. Ooi, and A. Tung, “Continuous skyline queries for moving objects”, in IEEE Transactions on Knowledge and Data Engineering, vol. 18(12), 1645-1658, Dec. 2006.
[12] R. Fagin, A. Lotem, and M. Naor, “Optimal aggregation algorithms for middleware”, in Proceedings of ACM PODS,

2001, pp. 102-113.
[13] K. C. K. Lee, B. Zheng, H. Li, and W. C. Lee, "Approaching the Skyline in Z Order", in Proceedings of VLDB, 2007, pp. 279-290.
[14] A. Vlachou, C. Doulkeridis, Y. Kotidis, and M. Vazirgiannis, "SKYPEER: Efficient Subspace Skyline Computation over Distributed Data", in Proceedings of ICDE, 2007, pp. 416-425.
[15] K. Fotiadou and E. Pitoura, "BITPEER: C ontinuous Subspace Skyline Computation with Distributed Bitmap Indexes", in Proceedings of DaAMaP, 2008, pp. 35-42.
[16] C. Y. Chan, H. V. Jagadish, K-L. Tan, A-K. H. Tung, and Z. Zhang, "On High Dimensional Skylines", in Proceedings of EDBT, 2006, pp. 478-495.
[17] Y. Tao, X. Xiao, and J. Pei, "Subsky: Efficient Computation of Skylines in Subspaces", in Proceedings of ICDE, 2006, pp. 65-65.
[18] K. Deng, X. Zhou, and H. T. Shen, "Multi-source Skyline Query Processing in Road Networks", in Proceedings of ICDE, 2007, pp. 796-805.
[19] M. Sharifzadeh and C. Shahabi, "The Spatial Skyline Query", in Proceedings of VLDB, 2006, pp. 751-762.
[20] C.-Y. Chan, P.-K. Eng, and K.-L. Tan, "Stratified Com- putation of Skylines with Partially-Ordered Domains", in Proceedings of ACM SIGMOD, 2005, pp. 203-214.
[21] 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, 2006, pp. 659-670.
[22] J. Pei, B. Jiang, X. Lin, and Y. Yuan, "Probabilistic Skylines on Uncertain Data", in *P*roceedings of VLDB, 2007, pp. 15-26.
[23] M. Morse, J. M. Patel, and H. V. Jagadish, "Efficient Skyline Computation over Low-Cardinality Domains", in Proceedings of VLDB, 2007, pp. 267-278.
[24] E. Dellis and B. Seeger, "Efficient Computation of Reverse Skyline Queries", in Proceedings of VLDB, 2007, pp. 291-302.

AUTHORS PROFILE

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

Yasuhiko Morimoto is an Associate Professor at Hiroshima University. 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.