Algorithm for Computing Convex Skyline Objectsets on Numerical Databases

Md. Anisuzzaman SIDDIQUE\(^{(a)}\), Nonmember and Yasuhiko MORIMOTO\(^{(b)}\), Member

**SUMMARY**
Given a set of objects, a skyline query finds the objects that are not dominated by others. We consider a skyline query for sets of objects in a database in this paper. Let \(s\) be the number of objects in each set and \(n\) be the number of objects in the database. The number of sets in the database amounts to \(n{C}_s\). We propose an efficient algorithm to compute convex skyline of the \(n{C}_s\) sets. We call the retrieve skyline objectsets as “convex skyline objectsets”. Experimental evaluation using real and synthetic datasets demonstrates that the proposed skyline objectset query is meaningful and is scalable enough to handle large and high dimensional databases. Recently, we have to aware individual’s privacy. Sometimes, we have to hide individual values and are only allowed to disclose aggregated values of objects. In such situation, we cannot use conventional skyline queries. The proposed function can be a promising alternative in decision making in a privacy aware environment.

**key words**: skyline query, privacy, skyline objectsets

1. Introduction

Skyline queries retrieve a set of skyline objects so that a user can choose promising objects from them and make further inquiries. Such skyline query functions are important for several database applications including customer information system, multi-criteria decision making, data mining, and so forth. Given a \(k\)-dimensional database \(DB\), an object \(p\) is said to be in skyline of \(DB\) if there is no object \(q\) in \(DB\) such that \(q\) is better than \(p\) in all \(k\) dimension. If there exist such object \(q\), then we say that \(p\) is dominated by \(q\) or \(q\) dominates \(p\). Figure 1 shows a typical example of skyline. The table in the figure is a list of hotels, each of which contains two numerical attributes: “distance” and “price”. In the list, the best choice for a client comes from the skyline, i.e., one of \(\{h_1, h_3, h_4\}\) in general (See Fig. 1 (b)). A number of efficient algorithms for computing skyline have been reported in the literature [1]–[6].

Recently, to preserve individuals’ privacy is one of important data management issues. Sometimes, we have to hide individual values to preserve privacy and we are only allowed to disclose aggregated values of objects. In such situation, we cannot use conventional skyline queries.

In this paper, we consider a skyline query for sets of objects in a database. Let \(s\) be the number of objects in each set and \(n\) be the number of objects in the database. The number of sets in the database amounts to \(n{C}_s\). We propose an efficient algorithm to compute convex skyline of the \(n{C}_s\) sets. We call retrieved skyline objectsets as “convex skyline objectsets”. This function does not disclose value of each object. Instead, it discloses aggregated values of \(s\) objects. It will be one of the most promising alternatives for decision making in a privacy aware environment.

In addition, sometimes users are not interested in individual objects but sets of objects. In such cases, the convex skyline objectsets function can be utilized. For example, if a user has to reserve multiple hotels at a time, she/he has to find a set of hotels that she/he prefers. Then, she/he needs to consider a problem of choosing a preferable set of objects in a database.

Assume an event organizer has to reserve rooms in three different hotels around the event venue. Look at the example in Fig. 1 again. The conventional skyline query, which outputs \(h_1\), \(h_3\), and \(h_4\), doesn’t provide sufficient information for the set selection problem. To solve the problem, we propose a function called convex skyline objectsets query that computes objectsets that lie in the convex hull of all objectsets.

Figure 2 (a) is a list of objectsets, in which all of the combinations of three hotels are listed. The “\(h_{123}\)” denotes a set of \(\{h_1, h_2, h_3\}\). “Distance” and “Price” of “\(h_{123}\)” are the
sum of the “Distance” and “Price” of respective hotels in the set. The event organizer prefers an objectset in the skyline of the combinations of three hotels, i.e., one of \( \{h_{123}, h_{135}, h_{235}, h_{234}, h_{345}\} \) as we can see in Fig. 2 (b). Our convex skyline objectsets query efficiently computes those convex skyline objectsets. We can also employ other aggregation functions to compute s-objectset. If we choose average or mean as an aggregation function, it will also produce the same result. For example, after computing average “Price” and “Distance” for all combination of three hotels, skyline query will retrieve the same result, i.e., \( \{h_{123}, h_{135}, h_{235}, h_{234}, h_{345}\} \) as objectset skyline.

Note that objects in the convex hull are different from objects in the skyline. Figure 3 shows the difference. Some objectsets in the skyline may not be in the convex hull as “p3” in the figure. There are three reasons why we exclude such non-convex skyline objectsets. (1) Non-convex objects cannot be the optimal point in any criterion if the objective function is linear. Therefore, in general, we will not choose such a point like “p3” in an optimization problem. (2) A skyline query often outputs too many objects to analyze intensively. Therefore, we think that we had better to eliminate such non-convex points to make the size of a query result small. (3) If we concentrate on convex skyline objects, we can compute objectsets efficiently as we will explain later in this paper. Though we exclude some skyline objects that are not in the convex hull, we call the result objectsets as “convex skyline objectsets” since the name “skyline” is popular and is easy to be recognized in the data management literature.

The rest of the paper is organized as follows: In Sect. 2, we review the related work. Section 3 describes the preliminaries for objectsets as well as the problem of convex skyline objectsets. In Sect. 4, we present an algorithm for computing convex skyline objectsets. We present experimental evaluations of the proposed algorithm in Sect. 5. Finally, we conclude the paper with some directions for future work in Sect. 6.

2. Related Work

The basic idea of skyline queries came from some old research topics like contour problem [10], maximum vector [11], and convex hull [12]. Borzsonyi et al. [1] first introduced the skyline operator over large databases and proposed three algorithms: the Block-Nested-Loops (BNL), Divide-and-Conquer (D&C), and B-tree-based schemes. BNL compares each object in a database 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 mem-

ory, and the input relation is sequentially scanned. In this way, a block of skyline objects is produced in each 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. The D&C method divides the data space into several regions, then calculates the skyline in each region, and finally produces the skyline from the objects in the regional skylines.

Chomicki et al. [8] proposed an algorithm named Sort-Filter-Skyline (SFS) as a variant of BNL. SFS requires the dataset to be pre-sorted according to some monotone scoring function. Since the order of the objects guarantees that no object dominates any objects before it in the order, the comparisons of tuples can be simplified.

The SFS algorithm was further significantly improved by Godfrey et al. [9]. They devise an algorithm called Linear Elimination Sort for Skyline that combines external sort and skyline search in a multi-pass fashion. It reduces the sorting cost and provides an attractive asymptotic best-case and average case performance if the number of skyline points is small.

Tan et al. proposed two progressive skyline computing methods [4]. The first method used bitmaps to map each object to an \( m \)-bit vector, and then identifies whether an object is in skyline. The second method introduces the specialized B-tree for each combination list of dimensions that a user might be interested in, and then, the algorithm loads the first batch of each list, and finds the ones that are not dominated by any of the already-found skyline objects into the skyline list.

Kossmann et al. observed that the skyline problem is closely related to the nearest neighbor (NN) search problem. They proposed an algorithm that returns skyline objects progressively by applying a NN search on an \( R^* \)-tree indexed dataset [2]. Branch-and-Bound Skyline, proposed by Papadias et al., is a progressive algorithm based on the Best-First Nearest Neighbor algorithm [3]. Instead of searching for the nearest neighbor again and again, it directly prunes using the \( R^* \)-tree structure. Kapoor studies the problem of dynamic maintenance of data structure for an incremental skyline computation in a 2-dimensional space [13].

The number of skyline objects, sometimes, becomes huge especially in high dimensional databases. The problem of selecting objects with some designated dominance properties has been recently investigated in several papers [14]–[16]. Chan et al. [14] introduce \( k \)-dominant skyline query which relaxes the idea of dominance to \( k \)-dominance so that an object is likely to be dominated by another. They proposed three algorithms, namely, One-Scan Algorithm, Two-Scan Algorithm, and Sorted Retrieval Algorithm. They use the property that a \( k \)-dominant skyline objects cannot be worse than any skyline on more than \( k \) dimensions. Chan et al. introduce a novel metric, called skyline frequency that compares and ranks the interestingness of data objects based on how often they are returned in the skyline when different number of dimensions (i.e., subspaces) are considered [15]. An object with a high skyline frequency is more
interesting as it can be dominated on fewer combinations of the dimensions. Kolotun and Papadimitriou [16] aim to find the minimum set of objects to approximately dominate all data objects. They propose a novel multi-dimensional query that refines the skyline query, and has the advantage that the answers contain considerably fewer objects, without being much less informative.

Recently, various aspects of skyline computation have been explored. Yuan et al. [17] studies the problem of computing the skylines in all subspaces and develop efficient algorithms. Jian et al. [6] explores the structure of skylines in subspaces and uses the concepts of skyline groups and decisional subspaces to capture the semantics of subspace skylines. Methods in [17] and [6] compute skylines for every subspace and interestingly, both studies suggest that a top-down first search framework may favor efficient computation.

Tian and Zhang [19] propose the compressed skycube, which concisely encodes the essential information of all subspaces skylines, and can be updated with much lower cost than the traditional skycube. Tian et al. [26] proposed a faster algorithm for computing sky-cube from scratch. Tao et al. [18] discuss skyline queries in arbitrary subspaces. Ji et al. [22] argue that skyline analysis is also meaningful on uncertain data. They proposed probabilistic skyline algorithm for skyline evaluation has also been studied in the context of spatial environments [20], [25] and in the context of partially-ordered attribute domains [21].

3. Convex Skyline Objectsets Problem

We consider the database $DB$ having $k$ attributes and $n$ records. Let $a_1, a_2, \ldots, a_k$ be the $k$ attributes of $DB$. Without loss of generality, we assume that smaller value in each attribute is better.

3.1 Skyline Objectsets

Let $s$-objectset be an objectset whose size is $s$. We assume $s$ is a relatively small number such that $2 \leq s \leq 10$ though our method is applicable for larger $s$. Let $S$ be the set of all $s$-objectsets in $DB$. Note that the number of $s$-objectsets in $DB$ is $\binom{n}{s}$, we denote the number by $|S|$. We assume a virtual database of $S$ on the $k$ dimensional space of $DB$. Each record of the database is an $s$-objectset whose value of each attribute (dimension) is the sum of $s$ values of corresponding $s$ objects. We denote $p.a_i$ as the $l$-th attribute value of a record $p$ in $S$.

An $s$-objectset $p \in S$ is said to dominate another $s$-objectset $q \in S$, denoted as $p \preceq q$, if $p.a_i \leq q.a_i$ for all $1 \leq i \leq k$. An $s$-objectset $p \in S$ is said to be a skyline objectset if $p$ is not dominated by any other $s$-objectset in $S$.

3.2 Convex Skyline

We can consider a record in $S$ to be a point in $k$-dimensional vector space. Convex hull is the minimum convex solid that encloses all of the points of $S$. The dotted line polygon in the left of Fig. 4 is an example of convex hull in two dimensional space.

In the Fig. 4, $p_1$ and $p_4$ are the points that have the minimum value of attribute $a_1$ and $a_2$, respectively. Notice that such points must be in the convex hull. We call the line between $p_1$ and $p_4$ the initial facet. Among all points in the convex hull, points that lie outside of the initial facet are skyline objects and we call such points convex skyline objects.

The triangle surrounded by $p_1$, $p_2$, and $p_3$ in the right of Fig. 4 is an example of the initial facet in three dimensional space, where $p_1$, $p_2$, and $p_3$ have the smallest value in $a_1$, $a_2$, and $a_3$, respectively. Convex skyline objects are points in the convex hull outside the initial facet.

In $k$-dimensional space, we compute such initial hyperplane surrounded by $k$ points as the initial facet. Then, we compute convex skyline objects that lie in the convex hull and lie outside the initial facet.

The definition of convex skyline objectsets problem can be simplified as follows: Given a natural number $s$, find all $s$-objectsets that lies in both the convex hull and the skyline of $S$.

Conventional skyline queries do not solve this problem of objectsets since some non-skyline object (dominated object) can be a member of a convex skyline objectset. For example, as we can see in Fig. 2(b), after computing the convex skyline of 3 hotels we get five hotel sets $h_{123}$, $h_{135}$, $h_{235}$, $h_{234}$, and $h_{345}$. Here both of $h_2$ and $h_5$ are the members...
of a convex skyline 3-objectsets but they are not in skyline of individual hotels.

4. Algorithm for Convex Skyline Objectsets

In this section, we present an algorithm for computing convex skyline s-objectsets. If we compute all of the s-objectsets from the original database and make a database containing |S| records, the problem can be solved by conventional skyline query algorithms. However, |S| is unacceptably large when the original database size is large. Therefore, we consider an algorithm for finding convex skyline objectsets without computing |S| s-objectsets.

4.1 Touching Oracle

Each s-objectset in S can be represented as a k-dimensional point \( x = (x_1, x_2, \ldots, x_k) \) where \( x_i \) (1 ≤ i ≤ k) is the sum of the i-th attribute’s value of the s objects in DB.

**Touching oracle** function is a method to compute a point on the convex hull without generating S. It computes the tangent point of the convex hull of S and a (k − 1)-dimensional hyperplane directly from DB.

Consider the examples of Fig. 1 and Fig. 2 again. There are 10 points in S if s = 3. Since there are two attributes in the database, those 10 points are in two-dimensional space as in Fig. 2(b). The dotted polygon in Fig. 5 is the convex hull of the 10 points. In the example, there are five records in the original databases DB as in Fig. 1(a). Each of the five records is also represented as a two-dimensional point, which we call an atomic point, which is denoted as a.

Assume there is a (k − 1)-dimensional hyperplane which is a line if k = 2, whose normal vector is \( \Theta_1 = (-1, 0) \) in the two-dimensional space. In order to find the tangent point with the 1-dimensional hyperplane (line) and the convex hull without precomputing all points in S, we compute \( (\Theta_1, a) \), i.e., inner products of the normal vector and each atomic point as in the second column of Table 1. We choose the top three inner products, i.e., \( \{h_1, h_2, h_3\} \). Those top three inner products compose the tangent point (12, 15), which is the 3-objectset, \( h_{123} \). Similarly, for a line with \( \Theta_2 = (0, -1) \), we can find \( \{h_3, h_4, h_5\} \) is the top three in the inner products of \( (\Theta_2, a) \). It composes the tangent point (20, 8), which is the tangent point of the convex hull and the 1-dimensional hyperplane (line) whose normal vector is \( \Theta_2 \).

In k-dimensional case, we choose a (k − 1)-dimensional hyperplane composed by k points on the convex hull and compute normal vector of the hyperplane. If there is a point of the convex hull outside the hyperplane, we can touch the point with a (k − 1)-dimensional hyperplane that has the same normal vector. By using the normal vector, we can find the tangent point with the tangent (k − 1)-dimensional hyperplane by using the touching oracle function, which chooses top s inner products from n atomic points in DB. The touching oracle function chooses the top s among n objects. We can find the m-th object for any m (1 ≤ m ≤ n) in O(n) without sorting the n objects [27]. Therefore, we can choose the top s objects in O(sn). Since s is negligible small constant compared to n, the touching oracle function is approximated to O(n).

4.2 Convex Hull Search

Next, we discuss how to use the touching oracle function to compute all convex skyline s-objectsets. First of all, we compute initial k tangent points that can be computed by touching oracle with initial k vectors \( \Theta_x = (\theta_1, \theta_2, \ldots, \theta_k) \) where \( \theta_i = -1 \) if \( i = x \), otherwise \( \theta_i = 0 \) for each \( x = 1, \ldots, k \). Note that those k initial tangent points are on the horizon of the initial facet ((k − 1)-dimensional hyperplane). Convex skyline s-objectsets are points lie outside of the initial facet and are in the convex hull.

Next, we compute the normal vector of the initial facet. In the example above, we have initial two tangent points: we have \( p_1 = (12, 15) \) with the normal vector \( \Theta_1 = (-1, 0) \) and we have \( p_2 = (20, 8) \) with the normal vector \( \Theta_2 = (0, -1) \). Using the facet containing the two initial points, we can compute the normal vector of the facet as \( \Theta_{1,2} = ((-15 - 8), (12 - 20)) = (-7, -8) \), which directs outside of the facet. Using this normal vector, we can find new tangent point \( h_{235} \), which is (16, 10). The new tangent point expands the initial facet into two facets, which are the facet surrounded by \( p_1 = (12, 15) \) and (16, 10) and the facet surrounded by (16, 10) and \( p_2 = (20, 8) \).

We recursively compute tangent points for each of the expanded facet. If we find new point outside the facet, we expand the facet further. We continually adopt the recursive operation while we can find new tangent point outside the facet. Finally, we can find all convex skyline s-objectsets. We can apply this recursive operation for higher k-dimensional space [7]. In the k-dimensional case, new tangent point, which is found by the touching oracle, divides the initial facet into k facets. In high dimensional case, the normal vector of each facet can be computed as follows:

**Three Dimensional Case**: Assume we have a facet sur-

---

**Table 1**: Inner product with tangent lines.

| a   | (\( \Theta_1, a \)) | (\( \Theta_2, a \)) | (\( \Theta_{1,2}, a \)) |
|-----|---------------------|---------------------|---------------------|
| \( h_1 \) | -3                  | -8                  | -85                 |
| \( h_2 \) | -5                  | -4                  | -67                 |
| \( h_3 \) | -4                  | -3                  | -72                 |
| \( h_4 \) | -9                  | -2                  | -79                 |
| \( h_5 \) | -7                  | -3                  | -73                 |
rounded by three points \( P_1 = (p_1, p_1, p_1) \), \( P_2 = (p_2, p_2, p_2) \), and \( P_3 = (p_3, p_3, p_3) \). We assume that \( P_1 \), \( P_2 \), and \( P_3 \) are clockwise order when we look the facet from outside of the convex hull. Now, we can compute two edge vectors by using the three points as follows. Suppose the edges vectors are \( V_1 = (v_1, v_1, v_1) = (p_1, p_2, p_3) - (p_1, p_1, p_1) \) and \( V_2 = (v_2, v_2, v_2) = (p_3, p_3, p_3) - (p_1, p_1, p_1) \). The outside normal vector of this facet is computed as the expansion of the following symbolic determinant.

\[
V_1 \otimes V_2 = \begin{vmatrix}
e_1 & e_2 & e_3 \\
v_{11} & v_{12} & v_{13} \\
v_{21} & v_{22} & v_{23}
\end{vmatrix}
\]

In the formula, \( e_1, e_2, \) and \( e_3 \) are the elementary vectors \((1,0,0)\), \((0,1,0)\), and \((0,0,1)\) respectively. Using this normal vector, we can divide this facet into three facets if we can find a new tangent point outside of the facet by the touching oracle function. If \( P \) is found outside of the facet, then the three new facets are \((P_1, P, P_3), (P_1, P_2, P), \) and \((P, P_2, P_3)\). The normal vectors of these three facets are \((P - P_1) \otimes (P_3 - P_1), (P_2 - P_1) \otimes (P - P_1), \) and \((P_2 - P) \otimes (P_3 - P)\) if points in each facet are clockwise order when we look the facet from outside of convex hull.

**Four Dimensional Case:** Assume that we have a facet surrounded by four points \( P_1 = (p_1, p_1, p_1, p_1), P_2 = (p_2, p_2, p_2, p_2), P_3 = (p_3, p_3, p_3, p_3), \) and \( P_4 = (p_4, p_4, p_4, p_4) \). Using similar operations of 3D case, we can compute three vectors \( V_1 = (v_1, v_1, v_1, v_1), V_2 = (v_2, v_2, v_2, v_2), V_3 = (v_3, v_3, v_3, v_3) \), and \( V_4 = (v_4, v_4, v_4, v_4) \). The outside normal vector of the facet that directs outside can be computed as the expansion of the following determinant.

\[
V_1 \otimes V_2 \otimes V_3 \otimes V(k - 1) = \begin{vmatrix}
e_1 & \cdots & e_k \\
v_{11} & \cdots & v_{1k} \\
\cdots & \cdots & \cdots \\
v_{(k-1)1} & \cdots & v_{(k-1)k}
\end{vmatrix}
\]

If \( P \) is found outside of the facet, then the \( k \) new facets are \((P_1, P_2, \ldots, P_k - 1, P_k), \ldots, (P_1, P_2, \ldots, P_k - 1, P)\). The normal vectors of these \( k \) facets are \(((P_2 - P) \otimes \cdots \otimes (P_{k-1} - P) \otimes (P_{k} - P_k)), \ldots, ((P_2 - P_1) \otimes \cdots \otimes (P_{k-1} - P_1) \otimes (P_{k} - P_1))\) if points in each facet are clockwise order when we look the facet from outside of convex hull.

**Figure 6:** Convex skyline object sets algorithm.

### 4.3 Pruning Atomic Points

Every time we compute a facet, we compute inner product for each atomic point with the normal vector of the facet as we mentioned in 4.1. During the computation, we can prune some atomic points that will not be a member of a convex skyline objectset.

Each \( s \)-objectset is composed of \( s \) atomic points. Sum of \( s \) inner product values reflects how far does the point of the \( s \)-objectset lie from the facet. For example, assume we compute inner product for each atomic point with the normal vector \( \Theta_{12} \) of the initial facet as in Table 1. Since both of \( h_{123} \) and \( h_{345} \) are in the initial facet, sum of inner product of those two points is same, which is \(-204\) \((h_1 + h_2 + h_3 = -85 - 67 - 52)\) and \(h_1 + h_4 + h_5 = -52 - 79 - 73\).

If sum of \( s \) atomic points is larger than \(-204\), the point of the \( s \)-objectset lies outside the facet. Otherwise, it lies inside and must not be a convex skyline objectset.

If there are atomic points whose inner product value is
smaller than or equal to the value, each of such points is, by itself, inside of the facet and will not be a member of a convex skyline objectset. During the convex hull search, we prune some atomic points by using the threshold value. Note that if a facet is expanded, more atomic objects are likely to be pruned.

5. Experiments

In this section, we conduct a series of experiments to evaluate the performance of our method using different types of datasets. We examined the proposed algorithm by using synthetic datasets and real datasets. We evaluated the efficiency and the scalability of the proposed methods in term of the cardinality and sets size of the datasets. The proposed algorithm was implemented using Java J2SE V6.0. We conduct simulation experiments on a PC with an Intel(R) Core2 Duo, 2 GHz CPU, and 3 GB main memory, running on Microsoft Windows XP operating systems. Each experiment is repeated five times and the average is taken.

5.1 Performance on Synthetic Datasets

We experimentally evaluate our proposed convex skyline objectsets algorithm on synthetic datasets. As benchmark databases, we use the databases proposed by Borzsonyi et al. [1]. Objects are generated using one of the following three value distributions:

- **Correlated**: a correlated database represents an environment in which points which are good in one dimension are also good in the other dimensions. In a correlated database, fairly few objects dominate many other objects.

- **Anti-Correlated**: an anti-correlated database represents an environment in which objects which are good in one dimension are bad in one or all of the other dimensions. As a result, the total number of non-dominating objects of an anti-correlated database is typically quite large.

- **Independent**: for this type of database, all attribute values are generated independently using uniform distribution. Under this distribution, objects rarely dominate each other when the dimension number grows. Thus the total number of non-dominating objects become large.

The generation of the synthetic datasets is controlled by the parameters “k”, “Size”, and “Dist”, where “k” is the number of attributes, “Size” is the number of objects in the dataset, and “Dist.” can be one of the three distributions above. We restrict dimensionality within five, because from the users’ point of view, we consider that higher dimension will increase the decision complexity and the result of skyline becomes useless in practice. Similar to others skyline related works in the literature, we employ the elapsed time as the performance metric.

Figure 7 shows the number of retrieved objectsets of 2D case on different distributions and different database size. Five different synthetic datasets with cardinality 10k, 25k, 50k, 75k, and 100k are used in this experiment. We vary objectsets size “s” between 2 to 10. The result shows that the number of convex skyline objectsets maintains a positive correlation with objectsets size, i.e., when “s” increases, the number of returned objectsets also increase as well.

We evaluate the response time of the convex skyline objectsets algorithm on the three different distributions. Figure 8 shows the results of 2D, 3D, 4D, and 5D cases for synthetic datasets with 100k. We observe that our algorithm becomes gradually slow if “s” increases. As in Fig. 7, if “s” becomes large the number of retrieved objectsets also increases. Considering this fact, we can conclude that our algorithm can perform well even if “s” becomes large.

Next, we evaluate the effect of database size. We used synthetic datasets with cardinality 10k, 25k, 50k, 75k, and 100k. In this experiment, we fix “s” to 10. Figure 9 shows the results. We observe that the response time increases if the database size increases. We also observe that it gradually increases if the dimension, “k”, increases. Similar to the previous experiments, the total elapsed time increases if database size and dimension increase.

5.2 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). Among 13, we choose five aspects whose domain have range [0, 4000]. The dataset approximates a correlated data distri-
The second dataset is FUEL dataset and extracted from “www.fueleconomy.gov”. FUEL dataset is 24 k 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). Similar to NBA dataset, we also choose five attributes from this dataset whose domain have range [8, 89]. Using both datasets we conduct the following two experiments.

In the first experiment, we report the number of retrieves objectsets for both type real datasets in Table 2. For this experiment we vary “s” from 2 to 10. Table 2 shows that when “s” increases, the number of returned objectsets for both datasets also increase as well. However, NBA dataset does not have a large number of returned convex skyline objectsets. This is due to the fact that the dataset is fairly correlated.

In the next experiment, we examine the effect of objectsets’ size, “s”. We vary “s” from 2 to 10 for this evaluation. Figure 10(a) and (b) show that the total elapsed time for both NBA and FUEL datasets. Similar to the experiments with synthetic dataset, the time increases if the objectsets’ size “s” and the dimension “k” increase.

As a short summary, our performance evaluations on synthetic as well as real datasets indicate that the proposed algorithm is efficient and scalable regarding data size. To the best of our knowledge, there is no previous work that focuses on the skyline set problem. As we mentioned, the problem can be solved by conventional skyline query after we compute all combinations of s-objects. However, it is unacceptably time and space consuming and our advantage is apparent since we do not have to compute the all combinations of s-objects.

6. Conclusion

We consider a convex skyline query for sets of objects in a database in this paper. Especially in privacy aware environments, we have to hide individual values and are only allowed to disclose aggregated values of objects. In such situations, the proposed function can be a promising alternative in decision-making. The proposed algorithm can compute all convex skyline s-objectsets without making all sets. Intensive experimental results showed that it is scalable and is efficient enough to handle large databases.

There are several avenues for future works. From a practical point of view, the integration of the proposed query into SQL and their treatment by the query optimizer of a DBMS deserve further investigation. Like the conventional skyline queries, the number of retrieved objects may be too large to analyze. In such situation, we can easily eliminate some objects by using another selective criterion like δ-dominant skyline in the post-processing of our algorithm. We also plan to consider a problem to maintain the convex skyline objectsets to handle updates of the database.

Acknowledgements

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

References

[1] S. Borzsonyi, D. Kossmann, and K. Stocker, “The skyline operator,” Proc. ICDE, pp.421–430, 2001.
[2] D. Kossmann, F. Ramsak, and S. Rost, “Shooting stars in the sky: An online algorithm for skyline queries,” Proc. VLDB Conference, pp.275–286, 2002.
[3] D. Papadias, Y. Tao, G. Fu, and B. Seeger, “An optimal and progressive algorithm for skyline queries,” Proc. ACM SIGMOD Conference, pp.467–478, 2003.
[4] K.-L. Tan, P.-K. Eng, and B.C. Ooi, “Efficient progressive skyline computation,” Proc. VLDB Conference, pp.301–310, 2001.
[5] C. Li, B.C. Ooi, A.-K.H. Tung, and S. Wang, “DADA: A data cube for dominant relationship analysis,” Proc. ACM SIGMOD Conference, pp.659–670, 2006.
[6] J. Pei, W. Jin, M. Ester, and Y. Tao, “Capturing the best views of skyline: A semantic approach based on decisive subspaces,” Proc. VLDB Conference, pp.253–264, 2005.
[7] Y. Morimoto, T. Fukuda, H. Matsuzawa, K. Yoda, and T. Tokuyama, “Algorithms for mining association rules for binary segmentations of huge categorical databases,” Proc. VLDB Conference, pp.380–391, 1998.
[8] J. Chomicki, P. Godfrey, J. Gryz, and D. Liang, “Skyline with presorting,” Proc. ICDE, pp.717–719, 2003.
[9] P. Godfrey, R. Shipley, and J. Gryz, “Maximal vector computation in large data sets,” Proc. VLDB Conference, pp.229–240, 2005.
[10] D.H. McLain, “Drawing contours from arbitrary data points,” Comput. J., pp.318–324, 1974.
[11] H.T. Kung, F. Luccio, and F. Preparata, “On finding the maxima of a set of vectors,” JACM, pp.469–476, 1975.
[12] P.F. Preparata and M.I. Shamos, Computational Geometry: An Introduction, Springer Verlag, 1985.
[13] S. Kapoor, “Dynamic maintenance of maxima of 2-d point sets,” SIAM J. Comput., pp.1858–1877, 2000.
[14] C.Y. Chan, H.V. Jagadish, K.-L. Tan, A.-K.H. Tung, and Z. Zhang, “Finding k-dominant skyline in high dimensional space,” Proc. ACM SIGMOD Conference, pp.503–514, 2006.
[15] C.Y. Chan, H.V. Jagadish, K.-L. Tan, A.-K.H. Tung, and Z. Zhang, “On high dimensional skylines,” Proc. EDBT, pp.478–495, 2006.
[16] V. Kolot and C.H. Papadimitriou, “Approximately dominating representatives,” Proc. ICDT, pp.204–214, 2005.
[17] Y. Yuan, X. Lin, Q. Liu, W. Wang, J.X. Yu, and Q. Zhang, “Efficient computation of the skyline cube, ” Proc. VLDB Conference, pp.241–252, 2005.
[18] Y. Tao, X. Xiao, and J. Pei, “Subsky: Efficient computation of skylines in subspaces,” Proc. ICDE, pp.65–65, 2006.
[19] T. Xia and D. Zhang, “Refreshing the sky: The compressed skycube with efficient support for frequent updates,” Proc. ACM SIGMOD Conference, pp.491–502, 2006.
[20] K. Deng, X. Zhou, and H.T. Shen, “Multi-source skyline query processing in road networks,” Proc. ICDE, pp.796–805, 2007.
[21] C.-Y. Chan, P.-K. Eng, and K.-L. Tan, “Stratified computation of skylines with partially-ordered domains,” Proc. ACM SIGMOD Conference, pp.203–214, 2005.
[22] J. Pei, B. Jiang, X. Lin, and Y. Yuan, “Probabilistic skylines on uncertain data,” Proc. VLDB Conference, pp.15–26, 2007.
[23] M. Morse, J.M. Patel, and H.V. Jagadish, “Efficient skyline computation over low-cardinality domains,” Proc. VLDB Conference, pp.267–278, 2007.
[24] E. Dellis and B. Seeger, “Efficient computation of reverse skyline queries,” Proc. VLDB Conference, pp.291–302, 2007.
[25] M. Sharifzadeh and C. Shahabi, “The spatial skyline query,” Proc. VLDB Conference, pp.751–762, 2006.
[26] J. Pei, A.W.-C. Fu, X. Lin, and H. Wang, “Computing compressed multidimensional skyline cubes efficiently,” Proc. ICDE, pp.96–105, 2007.
[27] T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein, Introduction to Algorithms, 2nd ed., pp.189–192. MIT Press, England, 2001.

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.