Efficient Selection of Representative Combinations of Objects from Large Database

Md. Anisuzzaman Siddique, Asif Zaman, and Yasuhiko Morimoto
Graduate School of Engineering, Hiroshima University
Higashi-Hiroshima, Japan
Email: anis_cst@yahoo.com, {d140094@, morimoto@mis.}hiroshima-u.ac.jp

Abstract—**Many applications require us to select combinations of objects from a database. To select representative combinations is one of the important processes for analysing data in such applications. Skyline query, which retrieves a set of non-dominant objects, is known to be useful to select representative objects from a database. Analogically, skyline query for combinations is also useful. Hence, we consider a problem to select representative distinctive combinations, which we call "objectsets", in a numerical database in this paper. We analyse the properties of skyline objectset computation and develop filtering conditions to avoid needless objectset enumerations as well as comparisons among them. We perform a set of experiments to testify the importance and scalability of our skyline objectset method. In addition, we confirm that those filtering strategies also work for skyline objectset query variant called skyband objectset query. Therefore, we propose another method to compute skyband objectset skyline result. Our experiments also confirm the effectiveness and scalability of skyband objectset skyline method.**

Keywords–Dataset; Skyline queries; Objectsets; Dominance relationship.

I. Introduction

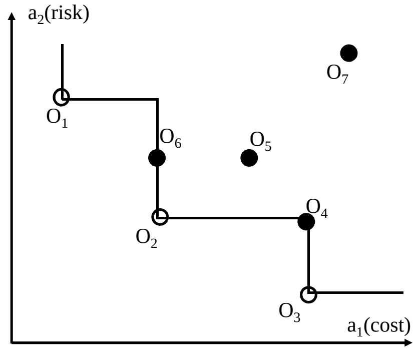
This work propose an algorithm called complete objectset skyline (CSS) to resolve the objectsets skyline query problem. This article is an extended version of [1].

To select representative objects in a database is important to understand the data. Assume that we have a hotel database. To analyse the database, we, first, take a look at representative records, for example, the cheapest one, the most popular one, the most convenient one and so on. Skyline query [2] and its variants are functions to find such representative objects from a numerical database. Given a m-dimensional dataset D, an object O is said to dominate another object O' if O is not worse than O' in any of the m dimensions and O is better than O' in at least one of the m dimensions. A skyline query retrieves a set of non dominate objects. Consider an example in the field of financial investment. In general, an investor tends to buy the stocks that can minimize cost and risk. Based on this general assumption, the target can be formalized as finding the skyline stocks with smaller costs and smaller risks. Figure 1(a) shows seven stocks records with their costs (a_1) and risks (a_2). In the list, the best choice for a client comes from the skyline, i.e., one of $\{O_1, O_2, O_3\}$ in general (see Figure 1(b)).

A key advantage of the skyline query is that it does not require a specific ranking function; its results only depend on the intrinsic characteristics of the data. Furthermore, the skyline does not relay on different scales at different dimensions. For example risk unit or cost unit in Figure 1 may be not same but it does not affect the skyline query result. However, the order of the dimensional projections of the objects is important. Skyline query has broad applications including product or restaurant recommendations [3], review evaluations with user ratings [4], querying wireless sensor networks [5], and graph analysis [6]. Algorithms for computing skyline objects have been discussed in the literature [7] [8] [9] [10].

ID	a_1(cost)	a_2(risk)
O_1	2	8
O_2	4	4
O_3	8	2
O_4	8	4
O_5	6	6
O_6	4	6
O_7	10	10

a) Dataset



b) Skyline

Figure 1. A skyline problem

One of the known limitations of the skyline query is that it can not answer various queries that require us to analyse not just individual object of a dataset but also their combinations. It is very likely that an investor has to invest more than one stock. For example, investment in O_1 will render the lowest cost. However, this investment is also very risky. Are there any other stocks or sets of stocks, which allow us to have a lower investment and/or a lower risk? These answers are often referred to as the investment portfolio. How to efficiently find such an investment portfolio is the principle issue studied in this work.

We consider a skyline query for distinctive combinations of objects (objectsets) in a database. Let k-objectset be a set, which contains k another object O' if O is not worse than O' in any of the m dimensions and O is better than O' in at least one of the m dimensions. A skyline query retrieves a set of non dominate objects. Consider an example in the field of financial investment. In general, an investor tends to buy the stocks that can minimize cost and risk. Based on this general assumption, the target can be formalized as finding the skyline stocks with smaller costs and smaller risks. Figure 1(a) shows seven stocks records with their costs (a_1) and risks (a_2). In the list, the best choice for a client comes from the skyline, i.e., one of $\{O_1, O_2, O_3\}$ in general (see Figure 1(b)). objects n be the number of objects in the dataset. The number of k-objectsets in the dataset amounts to ${}_nC_k$. We propose an efficient algorithm to compute variants of skyline query among the ${}_nC_k$ sets.

Assume an investor has to purchase two stocks. In Figure 1, conventional skyline query outputs $\{O_1, O_2, O_3\}$, which does not suggest sufficient information for the portfolio selection problem. Users may want to choose the portfolios, which

TABLE I. Sets of 2 Stocks

ID	$a_1(cost)$	$a_2(risk)$	ID	$a_1(cost)$	$a_2(risk)$	ID	$a_1(cost)$	$a_2(risk)$
$O_{1,2}$	6	12	$O_{2,4}$	12	8	$O_{3,7}$	18	12
$O_{1,3}$	10	10	$O_{2,5}$	10	10	$O_{4,5}$	14	10
$O_{1,4}$	10	12	$O_{2,6}$	8	10	$O_{4,6}$	12	10
$O_{1,5}$	8	14	$O_{2,7}$	14	14	$O_{4,7}$	18	14
$O_{1,6}$	6	14	$O_{3,4}$	16	6	$O_{5,6}$	10	12
$O_{1,7}$	12	18	$O_{3,5}$	14	8	$O_{5,7}$	16	16
$O_{2,3}$	12	6	$O_{3,6}$	12	8	$O_{6,7}$	14	16

are not dominated by any other sets in order to minimize the entire costs and risks. In such a case, if an user wants to select two stocks at a time from previous skyline result s/he can make two stock set such as $\{O_{1,2}, O_{1,3}, O_{2,3}\}$ and select any of them. However, set created from non dominant objects will be a non dominant set is not always true. For example, objectset $O_{1,3}$ is dominated by $O_{2,6}$ and there is no opportunity to judge this kind dominance relationship if we consider previous result only. That means an investor needs to create all two stocks sets after that perform domination check among those sets. It is very costly and not a very user friendly procedure. Table I shows sets consisting of two stocks, in which attribute values of each set are the sums of two component stocks. Objectsets $\{O_{1,2}, O_{2,3}, O_{2,6}\}$ cannot be dominated by any other objectsets (see Figure 2(a)) and, thus, they are the answers for the objectset skyline query. Furthermore, if the investor wants to buy three stocks then s/he needs to construct all of those three stocks sets and perform domination check among those sets to get the final result. In our running example, objectset skyline query for three stocks will retrieve objectsets $\{O_{1,2,3}, O_{1,2,6}, O_{2,3,4}, O_{2,3,6}\}$ as the query result.

Though a skyline query of objectsets is important in portfolio analysis, privacy aware data analysis, outlier-resistant data analysis, etc., there have been few studies on the objectsets skyline problem because of the difficulty of the problem. Su et al. proposed a solution to find the top-k optimal objectsets according to a user defined preference order of attributes [11]. However, it is hard to define a user preference beforehand for some complicated decision making tasks. Guo et al. proposed a pattern based pruning (PBP) algorithm to solve the objectsets skyline problem by indexing individuals objects [12]. The key disadvantage of the PBP algorithm is that it needs object selecting pattern in advance and the pruning capability depends on this pattern. Moreover, this algorithm is for fixed size objectset k and failed to retrieve result for all k.

We have introduced the objectsets skyline operator in 2010 [13]. In this work, we developed a method for finding the skyline objectsets that are on the convex hull enclosing all the objectsets. However, it misses objectsets that are not on the convex hull, which may provide meaningful results.

The main challenge in developing an method for objectset skyline is to overcome its computational complexity. The space complexity of objectset skyline computation is exponential in general and a dataset of n records can have up to ${}_nC_k$ skyline objectsets. This means that the time complexity of objectset skyline is also exponential because we have to generate all skyline objectsets during the computation. Since the set of intermediate candidate objectsets may not fit in memory, conventional method have to generate all candidate objectsets in a progressive manner and update the resultant objectset skyline dynamically. Thus, we cannot implement any index structures such as R-trees [9] and ZBtrees [14].

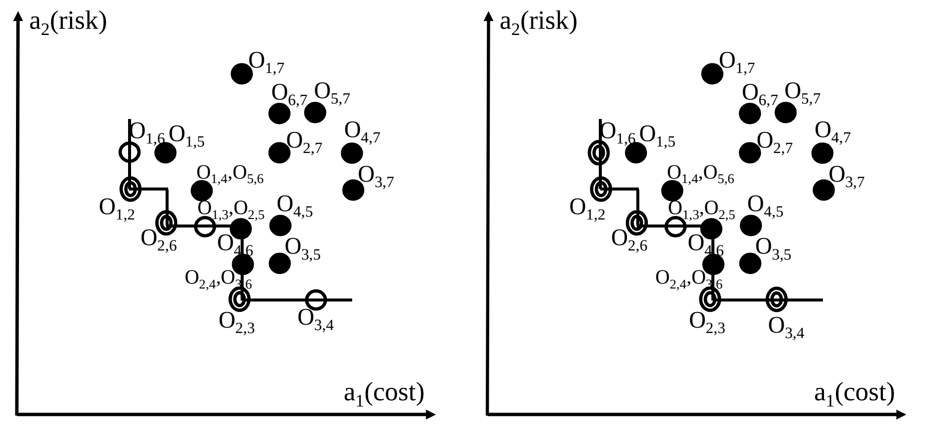
In this paper, we present an efficient solution that can select skyline objectsets, which include not only convex skyline objectsets but also non-convex skyline objectsets. The objectset size k can be varied from 1 to n and within which a user may select a smaller subset of his/her interest.

There is a well-known shortcoming in the skyline query. Though the result of the skyline query contains top-1 object for any scoring function, it cannot be used for selecting top-k ($k > 1$) object, which means that it cannot be used if an user wants more than one object for a specific scoring function. For example, if an user wants to choose top-3 cheapest stocks, the result of the skyline query does not contain all the top-3. Our previous objectset skyline query also has this shortcoming of the skyline query.

To solve the problem, in this paper, we also examine the objectset of another variant of skyline query called "skyband-objectset" query. Skyband-objectset query for K-skyband returns a set of objectsets, each objectset of which is not dominated by K other objectsets. In other words, an objectset in the skyband-objectset query may be dominated by at most $K - 1$ other objectsets.

The skyband-objectset query helps us to retrieve desired objectsets without any scoring function. It can increase (decrease) the number of objectsets by increasing (decreasing) the skyband value of K. From skyband-objectset result an user can easily choose his/her desired objectsets by applying top-k set queries. For the dataset in Figure 1, the skyband-set query for objectset size $s = 1$ and $K = 1$ retrieves all non dominated objectsets i.e., $\{O_1, O_2, O_3\}$. Again, from Figure 1(b) for $s = 1$ and $K = 2$ we get objectsets $\{O_1, O_2, O_3, O_6\}$ those are dominated by at most one objectset. Here O_4, O_5, and O_7 are dominated by more than one objectset so they are excluded from the query result. Next, for $s = 2$ and $K = 1$ it retrieves objectsets $\{O_{1,2}, O_{2,3}, O_{2,6}\}$ shown by double circles in Figure 2(a). Similarly, for $s = 2$ and $K = 2$ skyband-set query will retrieve $\{O_{1,2}, O_{1,6}, O_{2,3}, O_{2,6}, O_{3,4}\}$ (see Figure 2(b)). In this result, objectsets $O_{1,6}$ and $O_{3,4}$ are included because of dominated by only one objectset. Figure 2(b) shows that objectset $O_{1,6}$ is dominated by objectset $O_{1,2}$ and objectset $O_{3,6}$ is dominated by objectset $O_{2,3}$. Therefore, one can use the skyband-set query at the preprocessing step for top-k set query. It is useful for candidate set generation to select his/her desired objectsets without any scoring function.

We introdue CSS to resolve the objectsets skyline query problem. It progressively filters the objectsets that are impossible to be the objectsets skyline result, and uses a filtering mech-



a) Skyline of 2 Stocks b) Objectset skyband for s = 2 and K = 2

Figure 2. Objectset skyline problem

anism to retrieve the skyline objectsets without enumerating all objectsets. We develop two filtering techniques to avoid generating a large number of unpromising objectsets. Moreover, we confirm that proposing filtering strategies is also useful for skyband objectset query computation. We also propose another method called K-skyband objectset skyline (KSS) to compute skyband objectset query result. The efficiency of the both algorithms are then examined with experiments on a variety of synthetic and real datasets.

The remaining part of this paper is organized as follows. Section II reviews the related work. Section III discusses the notions and basic properties for skyline objectset as well as the problem of skyband objectset. In Section IV, we specify details of our algorithms with proper examples and analysis. We experimentally confirm our algorithms in Section VI under a variety of settings. Finally, Section VII concludes this work and describes our future intention.

II. Related Work

Our work is motivated by previous studies of skyline query processing as well as objectsets skyline query processing. Therefore, we briefly review conventional skyline query and objectset skyline query proposed in this work.

A. Skyline Query Processing

Borzsonyi et al. first introduced the skyline operator over large databases and proposed three algorithms: $Block\text{-}Nested\text{-}Loops(BNL)$, $Divide\text{-}and\text{-}Conquer$ $(D\&C)$, and B-tree-based schemes [2]. BNL compares each object of the 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 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 $Sort\text{-}Filter\text{-}Skyline(SFS)$ as a variant of BNL [7]. Among index-based methods, Tan et al. proposed two progressive skyline computing methods Bitmap and Index [15]. In the Bitmap approach, every dimension value of a point is represented by a few bits. By applying bit-wise AND operation on these vectors, a given point can be checked if it is in the skyline without referring to other points. The index method organizes a set of m-dimensional objects into m 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. The current most efficient method is $Branch\text{-}and\text{-}Bound\ Skyline(BBS)$, proposed by Papadias et al., which is a progressive algorithm based on the *best-first nearest neighbor (BF-NN)* algorithm [9]. Instead of searching for nearest neighbor repeatedly, it directly prunes using the R*-tree structure.

Recently, more aspects of skyline computation have been explored. Chan et al. proposed k-dominant skyline and developed efficient ways to compute it in high-dimensional space [16]. Lin et al. proposed n-of-N skyline query to support online query on data streams, i.e., to find the skyline of the set composed of the most recent n elements. In the cases where the datasets are very large and stored distributively, it is impossible to handle them in a centralized fashion [17]. Balke et al. first mined skyline in a distributed environment by partitioning the data vertically [18]. 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 [19]. Tao et al. discuss skyline queries in arbitrary subspaces [20]. More skyline variants such as dynamic skyline [21] and reverse skyline [22] operators also have recently attracted considerable attention.

B. Objectsets Skyline Query Processing

There are two closely related works, which are "top-k combinatorial skyline queries" [11] and "convex skyline objectsets" [13]. Su et al. studied how to find top-k optimal combinations according to a given preference order in the attributes. Their solution is to retrieve non-dominate combinations incrementally with respect to the preference until the best k results have been found. This approach relies on the preference order of attributes and the limited number (top-k) of combinations queried. Both the preference order and the top-k limitation may largely reduce the exponential search space for combinations. However, in our problem there is no preference order nor the top-k limitation. Consequently, their approach cannot solve our problem easily and efficiently. Additionally, in practice it is difficult for the system or a user to decide a reasonable preference order. This fact will narrow down the applications of [11]. Siddique and Morimoto studied the "convex skyline objectset" problem. It is known that the objects on the lower (upper) convex hull, denoted as CH, is a subset of the objects on the skyline, denoted as SKY. Every object in CH can minimize (maximize) a corresponding linear scoring function on attributes, while every object in SKY can minimize (maximize) a corresponding monotonic scoring function [2]. They aims at retrieving the objectsets in CH, however, we focus on retrieving the objectsets in $CH \subseteq SKY$. Since their approach relies on the properties of the convex hull, it cannot extend easily to solve complete skyline problem.

The similar related work is "Combination Skyline Queries" proposed in [12]. Guo et al. proposed a pattern based pruning (PBP) algorithm to solve the objectsets skyline problem by indexing individuals objects. The key problem of PBP algorithm is that it needs object selecting pattern in advance and the pruning capability depends on this pattern. For any initial wrong pattern this may increase the exponential search space. Moreover, it fails to vary the cardinality of objectset size k. Our solution does not require to construct any pattern previously and also vary the objectset size k from 1 to n. There are some other works focusing on the combination selection problem but related to our work weakly [23] [24]. Roy et al. studied how to select "maximal combinations". A combination is "maximal" if it exceeds the specified constraint by adding any new object. Finally, the k most representative maximal combinations, which contain objects with high diversities, are presented to the user. Wan et al. study the problem to construct k profitable products from a set of new products that are not dominated by the products in the existing market [24]. They construct non-dominate products by assigning prices to the new products that are not given beforehand like the existing products. Moreover, there exist no previous work that focus on skyband objectset query and are not suitable to solve this type query.

III. Preliminaries

This section formally defines objectset skyline query and objectset Skyband query and studies their basic properties.

Given a dataset D with m-attributes $\{a_1, a_2, \cdots, a_m\}$ and n objects $\{O_1, O_2, \cdots, O_n\}$. We use $O_i.a_j$ to denote the j-th dimension value of object O_i. Without loss of generality, we assume that smaller value in each attribute is better.

Dominance

An object $O_i \in D$ is said to dominate another object $O_j \in D$, denoted as $O_i \leq O_j$, if $O_i.a_r \leq O_j.a_r$ $(1 \leq r \leq m)$ for all m attributes and $O_i.a_t < O_j.a_t$ $(1 \leq t \leq m)$ for at least one attribute. We call such O_i as *dominant object* and such O_j as *dominated object* between O_i and O_j. For example, in Figure 1(b) object O_7 is dominated by object O_5. Thus, for this relationship, object O_5 is considered as dominant object and O_7 as dominated object.

Skyline

An object $O_i \in D$ is said to be a *skyline object* of D, if and only if does not exist any object $O_j \in D$ $(j \neq i)$ that dominates O_i, i.e., $O_j \leq O_i$ is not true. The skyline of D, denoted by $Sky(D)$, is the set of skyline objects in D. For dataset shown in Figure 1(a), object O_2 dominates $\{O_4, O_5, O_6, O_7\}$ and objects $\{O_1, O_3\}$ are not dominated by any other objects in D. Thus, skyline query will retrieve $Sky(D) = \{O_1, O_2, O_3\}$ (see Figure 1(b)).

In the following, we first introduce the concept of objectset, and then use it to define objectsets skyline. A k-objectset s is made up of k objects selected from D, i.e., $s = \{O_1, \cdots, O_k\}$ and for simplicity denoted as $s = O_{1,\cdots,k}$. Each attribute value of s is given by the formula below:

$$s.a_j = f_j(O_1.a_j, \cdots, O_k.a_j), (1 \leq j \leq m) \quad (1)$$

where f_j is a monotonic aggregate function that takes k parameters and returns a single value. For the sake of simplicity, in this paper we consider that the monotonic scoring function returns the sum of these values, i.e.,

$$s.a_j = \sum_{i=1}^{k} O_i.a_j, (1 \leq j \leq m) \quad (2)$$

though our algorithm can be applied on any monotonic aggregate function. Recall that the number of k-objectsets in D is ${}_nC_k = \frac{n!}{(n-k)!k!}$, we denote the number by $|S|$. If we consider stocks shown in Figure 1, then the total number of objectset for two stocks is ${}_7C_2$ i.e., $|S|$ = 21 and objectset $O_{1,2}$ is made up from object O_1 and O_2.

Dominance Relationship

A k-objectset $s \in D$ is said to dominate another k-objectset $s' \in D$, denoted as $s \leq s'$, if $s.a_r \leq s'.a_r$ $(1 \leq r \leq m)$ for all m attributes and $s.a_t < s'.a_t$ $(1 \leq t \leq m)$ for at least one attribute. We call such s as dominant k-objectset and s' as dominated k-objectset between s and s'. For example, in Figure 2(a) objectset $O_{1,6}$ is dominated by objectset $O_{1,2}$. In this relationship objectset $O_{1,2}$ is considered as dominant objectset and $O_{1,6}$ as dominated objectset.

Objectsets Skyline

A k-objectset $s \in D$ is said to be a skyline k-objectset if s is not dominated by any other k-objectsets in D. The skyline of k-objectsets in D, denoted by $Sky_k(D)$, is the set of skyline k-objectsets in D. Assume $k = 2$, then for the dataset shown in Table III, 2-objectset $O_{1,2}$, $O_{2,3}$, and $O_{2,6}$ are not dominated by any other 2-objectsets in D. Thus, 2-objectset skyline query will retrieve $Sky_2(D) = \{O_{1,2}, O_{2,3}, O_{2,6}\}$ (see Figure 2(a)).

Domination Objectsets

Domination objectsets of k-objectsets, denoted by $DS_k(D)$ is said to be a set of all dominated k-objectsets in D. Since the 1-objectsets skyline result is $Sky_1(D) = \{O_1, O_2, O_3\}$, then the domination objectsets of 1-objectsets is $DS_1(D) = \{O_4, O_5, O_6, O_7\}$, i.e., $D - Sky_1(D)$.

TABLE II. domRelationTable for 1-objectsets

$Object$	$Dominant\ Object$
O_1	$\varnothing$
O_2	$\varnothing$
O_3	$\varnothing$
O_4	O_2, O_3
O_5	O_2, O_6
O_6	O_2
O_7	$O_1, O_2, O_3, O_4, O_5, O_6$

Objectset Skyband

Objectset skyband query returns a set of objectsets, each objectset of which is not dominated by K other objectsets. In other words, an objectset in the skyband-set query may be dominated by at most $K - 1$ other objectsets. If we want to apply skyband objectset query in D and choose objectset size $s = 2$ and $K = 2$, then the objectset skyband will retrieve $\{O_{1,2}, O_{1,6}, O_{2,3}, O_{2,6}, O_{3,4}\}$ as query result.

IV. Complete Skyline Objectsets Algorithm

In this section, we present our proposed method called Complete Skyline objectSets (CSS). It is a level-wise iterative algorithm. Initially, CSS computes conventional skyline, i.e., 1-objectsets skyline then 2-objectsets skyline, and so on, until k-objectsets skyline.

For $k = 1$, we can compute 1-objectsets skyline using any conventional algorithms. In this paper, we use SFS method proposed in [7] to compute 1-objectsets skyline and receive the following domination relation table called $domRelationTable$.

For the objectsets skyline query problem, the total number of objectsets is $|S|$ = ${}_nC_k$ for a dataset D containing n objects when we select objectsets of size k. This poses serious algorithmic challenges compared with the traditional skyline problem. For example, Brute Force approach needs to calculate each objectset s and also needs to judge domination check among all $|S|$ objectsets. Fortunately, large parts of the computations can be avoided with our proposed CSS algorithm. As Table III illustrates, $|S|$ = 21 possible combinations are generated from only seven objects when $k = 2$. Even for a small dataset with thousands of entries, the number of objectsets is prohibitively large. Thanks to the Theorem 1, which gives us opportunity to prune many non-promising objectsets.

Theorem 1. *If all k members of an objectset s are in $DS_{k-1}(D)$, where $DS_{k-1}(D) = DS_1(D) \cup \cdots \cup DS_{k-1}(D)$, then objecsets $s \notin Sky_k(D)$.*

Proof: Assume $s = \{O_1, \cdots, O_k\}$ and $s \in Sky_k(D)$. Since all members of s are in $DS_{k-1}(D)$, then there must be k distinct dominant objectsets for each member of s. Suppose $\{O'_1, \cdots, O'_k\}$ are those k distinct objectsets and $\{O'_{1,\cdots,k}\}$ construct an objectset s'. Now, it implies dominance relationship $s' \leq s$, which contradict initial assumption $s \in Sky_k(D)$. Hence, a non dominant k-objectsets contains at least one skyline objectset. ■

Dominance relation given in Table II retrieves $DS_1(D)$ = $\{O_4, O_5, O_6, O_7\}$. By using Theorem 1 for $k = 2$, we can safely prune ${}_4C_2 = 6$ objectsets such as $\{O_{4,5}, O_{4,6}, O_{4,7}, O_{5,6}, O_{5,7}, O_{6,7}\}$ from $Sky_2(D)$ computation, since these combinations do not have any member from $Sky_{k-1}(D)$. After pruning by this theorem the number of remaining objecsets, i.e., 15 (21-6), is still too large for our running example. To avoid large objecsets comparison, CSS applies the second pruning strategy as follows:

Theorem 2. *Suppose S_1, S_2, and S_3 be the three objectsets in D. If objectset $S_1 \leq S_2$, then $S_1 \cup S_3 \leq S_2 \cup S_3$ is true.*

Proof: Given that S_1, S_2, and S_3 are the three objectsets in D and $S_1 \leq S_2$ is true. Now, if we think another objectset S_3 as a constant and add it on both side then we will get the relationship $S_1 \cup S_3 \leq S_2 \cup S_3$. Thus if objectset $S_1 \leq S_2$ is true, then $S_1 \cup S_3 \leq S_2 \cup S_3$ is also true. ■

Theorem 2 gives us another opportunity to eliminate huge number objectsets. Table II shows that object O_4 is dominated by O_2 and O_3 ($O_2 \leq O_4$ and $O_3 \leq O_4$). By using Theorem 2, we get following dominance relations for 2-objectsets: $O_{1,2} \leq O_{1,4}$, $O_{2,3} \leq O_{2,4}$, $O_{2,3} \leq O_{3,4}$. Similarly, object O_5 is dominated by O_2 and O_6 ($O_2 \leq O_5$ and $O_6 \leq O_5$). From this relation, we can derive $O_{1,2} \leq O_{1,5}$, $O_{2,3} \leq O_{3,5}$, and $O_{2,6} \leq O_{2,5}$. From $O_2 \leq O_6$, we can derive $O_{1,2} \leq O_{1,6}$, $O_{2,3} \leq O_{3,6}$. Finally, from the last row of Table II we have the dominance relationship $\{O_1, O_2, O_3, O_4, O_5, O_6 \leq O_7\}$. Now if we use objects $\{O_1, O_2, O_3\}$ as common objects then we get following additional dominance relationship $\{O_{1,2} \leq O_{1,7}$, $O_{2,3} \leq O_{3,7}$, $O_{2,6} \leq O_{2,7}\}$. Thus, according to Theorem 2, we can safely prune more 11 objectsets such as $\{O_{1,4}$, $O_{2,4}$, $O_{3,4}$, $O_{1,5}$, $O_{2,5}$, $O_{3,5}$, $O_{1,6}$, $O_{3,6}$, $O_{1,7}$, $O_{2,7}$, $O_{3,7}\}$ for $Sky_2(D)$. Actually, CSS algorithm will compose remaining (15-11) = 4 objecsets such as $\{O_{1,2}, O_{1,3}, O_{2,3}, O_{2,6}\}$ and it needs to perform domination checks among them. After performing domination check it retrieves $\{O_{1,2}, O_{2,3}, O_{2,6}\}$ as $Sky_2(D)$ query result. However, during this procedure CSS also updates the dominance relation table for 2-objectsets as shown in Table III.

For k = 2 dominance relation Table III retrieves domination objectsets $DS_2(D)$ = $\{O_{1,3}$, $O_{1,4}$, $O_{1,5}$, $O_{1,6}$, $O_{1,7}$, $O_{2,4}$, $O_{2,5}$, $O_{2,7}$, $O_{3,4}$, $O_{3,5}$, $O_{3,6}$, $O_{3,7}\}$. When $k = 3$, conventional skyline algorithm needs to check dominance relation among $|S|$ = 35 (${}_7C_3$) objectsets. In contrast to such conventional algorithms, our CSS algorithm does not compose those 3-objectsets if the distinct 3 objects are in $DS_1(D)$ or $DS_1(D) \cup DS_2(D)$. For $DS_1(D) = \{O_4, O_5, O_6, O_7\}$, CSS does not compute ${}_4C_3 = 4$ objectsets. They are $\{O_{4,5,6}, O_{4,5,7}, O_{4,6,7}\}$, and $O_{5,6,7}$. For $DS_1(D) \cup DS_2(D)$, CSS pruned another 22 objectsets. These 22 objectsets are $\{O_{1,3,4}$, $O_{1,3,5}$, $O_{1,3,6}$, $O_{1,3,7}$, $O_{1,4,5}$, $O_{1,4,6}$, $O_{1,4,7}$, $O_{1,5,6}$, $O_{1,5,7}$, $O_{1,6,7}$, $O_{2,4,5}$, $O_{2,4,6}$, $O_{2,4,7}$, $O_{2,5,6}$, $O_{2,5,7}$, $O_{2,6,7}$, $O_{3,4,5}$, $O_{3,4,6}$, $O_{3,4,7}$, $O_{3,5,6}$, $O_{3,5,7}$, $O_{3,6,7}\}$ After using Theorem 1, the remaining objectset number is reduced to 9 (35-26). After applying Theorem 2, CSS deos not need to compute another 5 objectets such as $\{O_{1,2,4}, O_{1,2,5}, O_{1,2,7}, O_{2,3,5}\}$, and $O_{2,3,7}$. Finally, the proposed algorithm will compose only four objectsets $\{O_{1,2,3}$, $O_{1,2,6}$, $O_{2,3,4}$, $O_{2,3,6}\}$ and perform domination check among these four objectsets to obtain $Sky_3(D)$. After the domination check, since these objectsets are not dominated by each other thus, CSS retrieves $\{O_{1,2,3}, O_{1,2,6}, O_{2,3,4}, O_{2,3,6}\}$ as $Sky_3(D)$ result. CSS will continue similar iterative procedure for the rest of the k values to compute $Sky_k(D)$.

TABLE III. domRelationTable for 2-objectsets

$Objectset$	$Dom.\ Objectset$	$Objectset$	$Dom.\ Objectset$
$O_{1,2}$	$\varnothing$	$O_{3,4}$	$O_{2,3}$
$O_{1,3}$	$\varnothing$	$O_{3,5}$	$O_{2,3}, O_{3,6}$
$O_{1,4}$	$O_{1,2}, O_{1,3}$	$O_{3,6}$	$O_{2,3}$
$O_{1,5}$	$O_{1,2}, O_{1,6}$	$O_{3,7}$	$O_{1,3}, O_{2,3}, O_{3,4}, O_{3,5}, O_{3,6}$
$O_{1,6}$	$O_{1,2}$	$O_{4,5}$	$O_{2,5}, O_{3,5}, O_{2,4}, O_{4,6}$
$O_{1,7}$	$O_{1,2}, O_{1,3}, O_{1,4}, O_{1,5}, O_{1,6}$	$O_{4,6}$	$O_{2,6}, O_{3,6}, O_{2,4}$
$O_{2,3}$	$\varnothing$	$O_{4,7}$	$O_{1,4}, O_{2,4}, O_{3,4}, O_{4,5}, O_{4,6}, O_{2,7}, O_{3,7}$
$O_{2,4}$	$O_{2,3}$	$O_{5,6}$	$O_{2,6}, O_{2,5}$
$O_{2,5}$	$O_{2,6}$	$O_{5,7}$	$O_{1,5}, O_{2,5}, O_{3,5}, O_{4,5}, O_{5,6}, O_{2,7}, O_{6,7}$
$O_{2,6}$	$\varnothing$	$O_{6,7}$	$O_{1,6}, O_{2,6}, O_{3,6}, O_{4,6}, O_{5,6}, O_{2,7}$
$O_{2,7}$	$O_{1,2}, O_{2,3}, O_{2,4}, O_{2,5}, O_{2,6}$		

V. Objectsets Skyband Computation

We adapt previous CSS method for objectset skyband query computation. We term this method as k-skyband objectset (KSS). Initially consider a skyband objectset query where objectset size $s = 1$ and skyband size $K = 1$ i.e., conventional skyline query. Then it retrieves $\{O_1, O_2, O_3\}$ as the objectset skyband query result. If we keep objectset size $s = 1$ and increase the skyband value K to 2, then objectset skyband result becomes $\{O_1, O_2, O_3, O_6\}$. Similarly, skyband objectset query for $s = 1$ and $K = 3$ retrieves $\{O_1, O_2, O_3, O_4, O_5, O_6\}$. Finally, for $s = 1$ and $K = 4$ KSS retrieves all objects as a result.

Similar to objectset skyline query problem if we select objectsets of size s, then the number of objectsets is $|S| = {}_nC_s$ for a dataset D containing n objects. Unfortunately, to compute skyband objectset we can not use Theorem 1. This is because, even if all k member of an objectset s are in $DS_k(D)$, it can be retrieved as the member of objectset skyband. However, to compute objectset skyband query efficiently, we introduce Theorem 3, which is useful to filter objectsets as well as to reduce the number of comparisons required domination check.

Theorem 3. *If an objectset, say s, is dominated by at least K other objectsets, then we do not need to compose objectsets that contain s for K-skyband-set computation.*

Proof: Let S_1, S_2, S_3, and S_4 be objectsets in DS. Assume that objectset S_1 is dominated by two objectsets S_2 and S_3 ($S_2 \leq S_1$ and $S_3 \leq S_1$). Thanks to Theorem 2, we can say that if $S_2 \leq S_1$ is true, then for super objectset $S_2 \cup S_4 \leq S_1 \cup S_4$ is also true. Similarly, $S_3 \cup S_4 \leq S_1 \cup S_4$ is true. There exist at least two other objectset such as $S_2 \cup S_4$ and $S_3 \cup S_4$ that can dominate super objectset $S_1 \cup S_4$. It implies that an objectset is dominated by two objectsets, it cannot be an objectset of 2-skyband-set. Thus, it is proved that if an objectset is dominated by at least K other objectsets then we do not need to compose super objectsets that contain the dominated objectset for skyband-set computation. ■

From Table II, we can easily construct similar $domRelationTable$ for objectset size $s = 2$ as shown in Table III by using Theorem 2 and Theorem 3. Dominance relation Table III retrieves candidates for objectset skyband queries when objecetset size $s = 2$. For example, if an user specifies skyband objectset query for $s = 2$ and $K = 1$, then the proposed algorithm will retrieve candidate objectsets $\{O_{1,2}, O_{1,3}, O_{2,3}, O_{2,6}\}$ from Table III. Note that the proposed algorithm will compose only four objectsets and perform domination check among them to obtain skyband objectset result $\{O_{1,2}, O_{2,3}, O_{2,6}\}$. Here, objectset $O_{1,3}$ is dominated by objectset $O_{2,6}$. Next, if the user concerns about skyband objectset query with $s = 2$ and $K = 2$, then the proposed algorithm will choose candidate objectsets $\{O_{1,2}, O_{1,3}, O_{1,6}, O_{2,3}, O_{2,4}, O_{2,5}, O_{2,6}, O_{3,4}, O_{3,6}\}$ and perform domination check among these objectsets. Finally, it retrieves $\{O_{1,2}, O_{1,6}, O_{2,3}, O_{2,6}, O_{3,4}\}$ as skyband-set query result. The dominance relation Table III retrieves candidate objectsets for any skyband objectset query when $s = 2$. The proposed method will continue similar iterative procedures to construct dominance relation table each time for larger value of s and ready to report objectset skyband queries result for any skyband value of K.

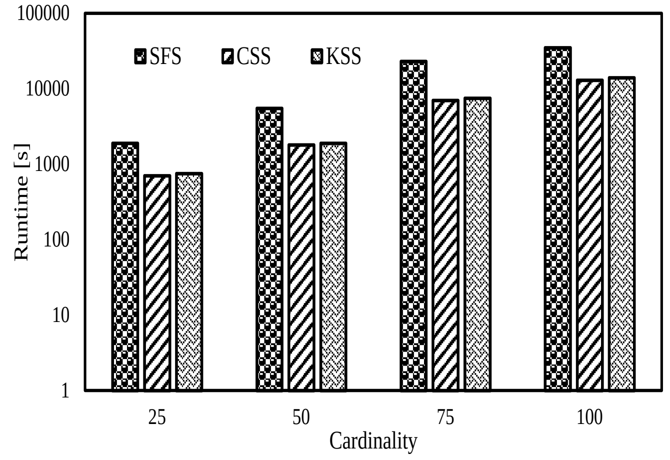
VI. Performance Evaluation

In the experimental section, we empirically evaluated the performance of our proposed CSS and KSS approaches. We do not compare our algorithms with the algorithm for computing top-k combinatorial skyline [11]. The reasons are twofold. First, RCA requires the user to determine the ranking of every dimension of the dataset, and its performance varies depending on the user preference. Second, while our algorithms compare only groups of the same size, RCA compares groups of different sizes as well. However, in order to measure their relative performance, we adapt SFS skyline algorithm to compute set skyline algorithm [7]. To make the comparison fair, we have excluded all the pre-processing cost of SFS method such as cost of objectset generation.

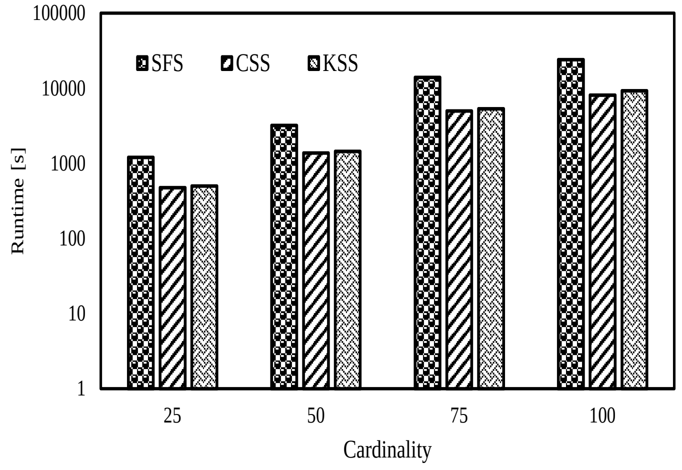
We conduct a set of investigations with different dimensionalities (m), data cardinalities (n), and objectset size (k) to judge the effectiveness and efficiency of proposed methods. All experiments were run on a computer with Intel Core i7 CPU 3.4GHz and 4 GB main memory running on Windows. We compiled the source codes under jdk 1.8. Each experiment is repeated five times and we report the average results for performance evaluation. The execution times in the graphs shown in this section are plotted in log scale.

A. Performance on Synthetic Datasets

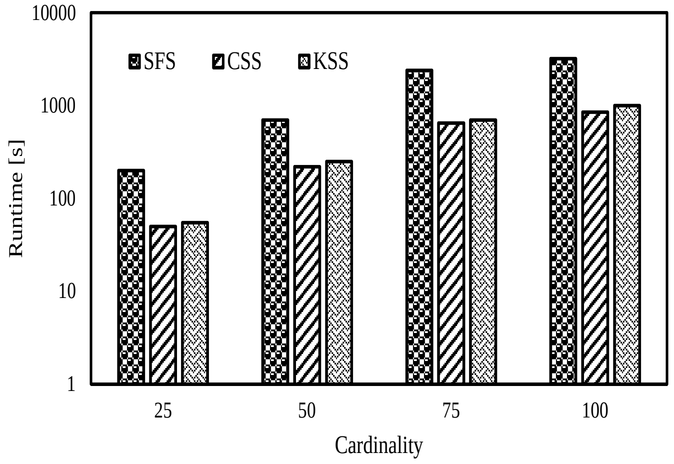
We prepared synthetic datasets with three data distributions correlated, anti-correlated, and independent, which are used in [2]. The results are shown in Figures 3, 4, and 5. The sizes of resulting synthetic datasets are varied from 2.3k to 161.7k depending on the number of objects n.



a) Anti-correlated



b) Independent



c) Correlated

Figure 3. Performance for different cardinality

Effect of Cardinality

For this experiment, we specify the data dimensionality m to 4, objectset size k to 3, and vary dataset cardinality n. If the cardinality n takes the values of 25, 50, 75, and 100 then total objectset size become 2.3k, 19.6k, 67.5k, and 161.7k, respectively. We plot the running times of the algorithms in Figure 3. Figure 3(a), (b), and (c) respectively reports the performance on the correlated, the independent, and the anti-correlated datasets. The horizontal line represents the data cardinality and the vertical line represents execution time. We observe that all methods are affected by data cardinality. If the data cardinality increases then their performances fall down. The results demonstrate that proposed methods significantly outperforms the SFS method. The performance of SFS method degrades rapidly as the the dataset size increases, especially for the anti-correlated data distribution. This represents that the proposed methods can successfully prune the objectset composing as well as many unnecessary comparisons. However, the difference between CSS and KSS is not very significant. KSS needs little bit extra time to retrieve results for all K.

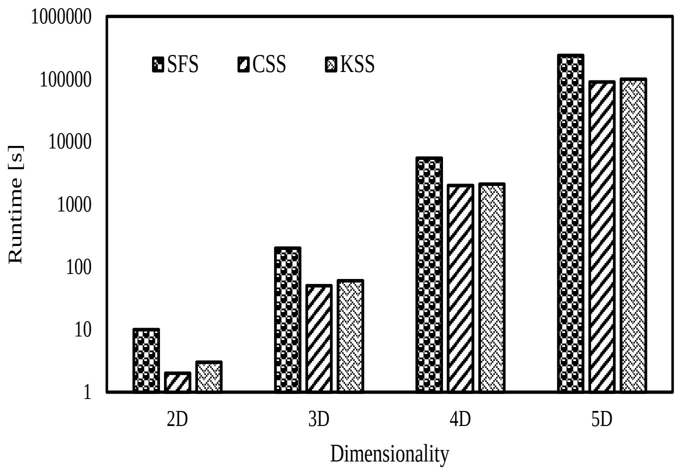
Effect of Dimensionality

In these experiments, we set the data cardinality n to 50, objectset size k to 3 and vary dataset dimensionality m ranges from 2 to 5. The elapsed time results are shown in Figure 4(a), (b), and (c). The horizontal line shows the data dimensionality m and the vertical line shows the execution time. The results showed that as the dimension m increases the performance of the all methods becomes slower. This is because checking the dominance relationship between two objectsets becomes more expensive with large values of m. For higher dimension, when the number of non dominant objectset increases the performance of all methods become sluggish. The running time of the proposed algorithms achieve satisfactory even when the dimension size is large. The results on correlated datasets are 9-20 times faster than the independent and the anti-correlated datasets. We observed that the time difference between CSS and KSS is not notable for the same reason as we discuss in previous section.

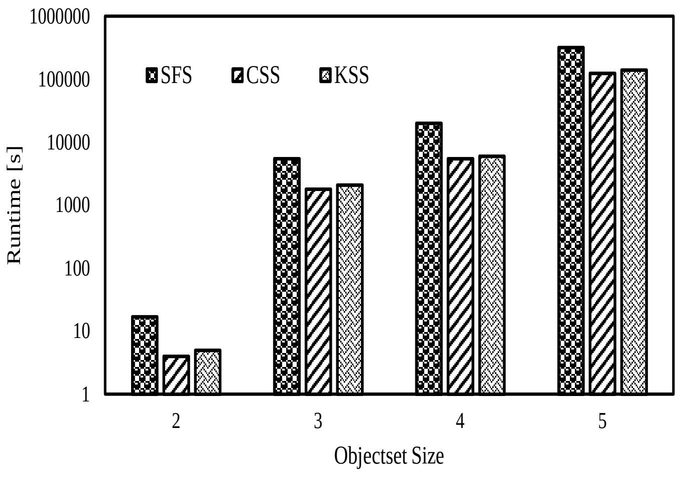
Effect of Objectset Size

In these experiments, we examine the performance of the proposed CSS and KSS under various objectset size k. We limit the data cardinality n to 50 and dataset dimensionality m to 4. The results are described in Figure 5(a), (b), and (c). The horizontal line shows the objectset size k and the vertical line shows the time taken. The results showed that as the objectset size k rises up, the performance of all methods are fallen down. The performance of SFS method is much worse than that of the proposed methods when the objectset size k is greater than 1. This is because the proposed methods use SFS algorithm for $k = 1$ to construct $domRelationTable$, and executing domination check. For higher value of s, it does not require to compose all objectset and it reduces the number of unnecessary comparisons. We notice that the time difference between CSS and KSS is not significant and CSS is faster that KSS.

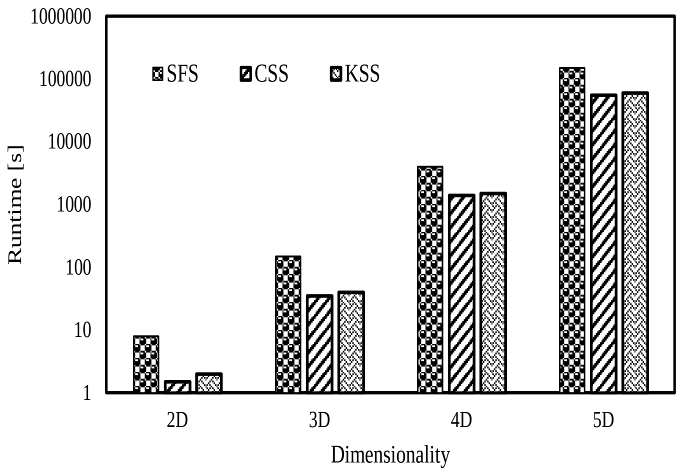
From the experimental results, we observe a pattern that the speed up of the proposed methods over SFS is 4-10 times faster. Since the number of skyline objectsets of the anti-correlated datasets is generally larger than those of the independent datasets and the correlated datasets, the algorithms



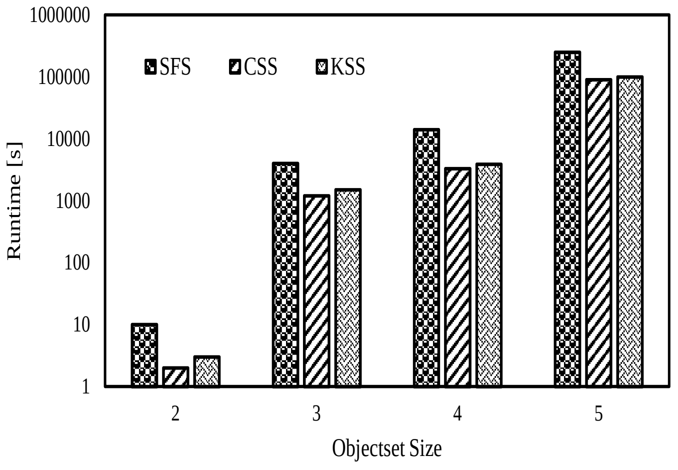
a) Anti-correlated



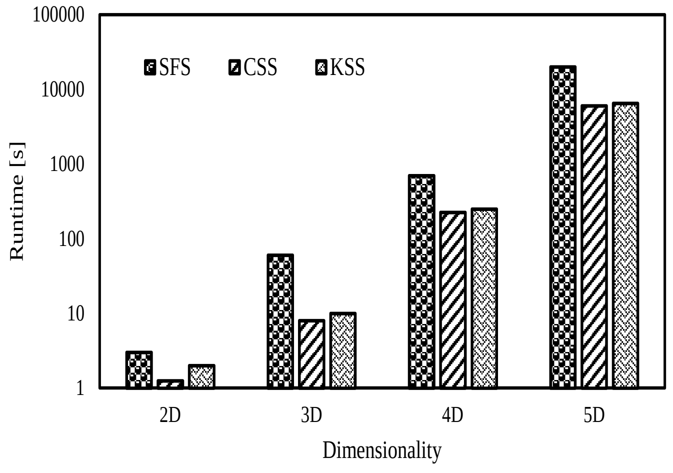
a) Anti-correlated



b) Independent

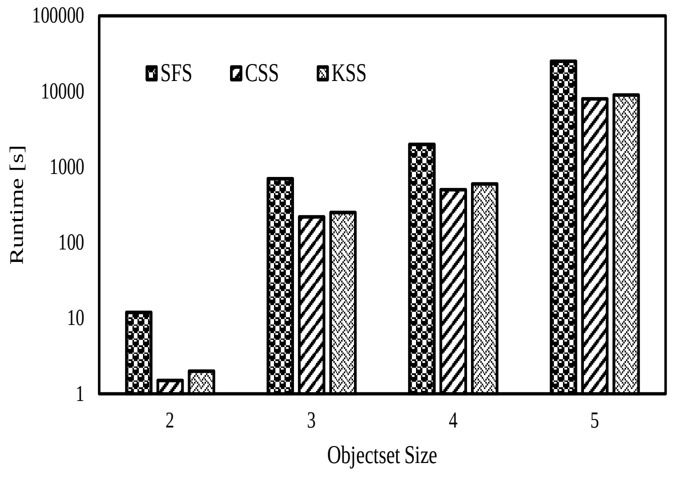


b) Independent



c) Correlated

Figure 4. Performance for different data dimension



c) Correlated

Figure 5. Performance for different objectset size

with the anti-correlated datasets take generally more execution time than those of the other datasets.

B. *Performance on Real Dataset*

We next present the experimental results of the proposed algorithms with real datasets. To examine the results for real dataset, we select the FUEL dataset, which is obtained 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.). The attribute domain range for this dataset is [8, 89].

To deal with FUEL dataset, we conducted similar experiments like synthetic datasets. First, to study dimensionality, we specify the data cardinality n to 50, objectset size k to 3 and vary dataset dimensionality m from 2 to 5.

Figure 6(a) illustrates the performance the objectset skyline and objectset skyband queries of different dimension sizes. As the dimensions increases, the running time for all methods increases accordingly. However, CSS and KSS outperforms than SFS technique.

Our second experiments on real data examine the performance of different data cardinality n. For these experiments, we limit the dimensionality m to 4, objectset size k to 3, and vary dataset cardinality n from 25 to 100. The results are shown in Figure 6(b). As the data size increases, the running time for all methods increase sharply. We can observe that CSS is better than KSS and SFS.

In the final experiments, we examine the performance under various objectset size k. We fix the data cardinality n to 50 and dimensionality m to 4. The results are shown in Figure 6(c). As the objectset size increases, the execution time for all methods also increase. We can observe that the running time of CSS and KSS are much superior to SFS.

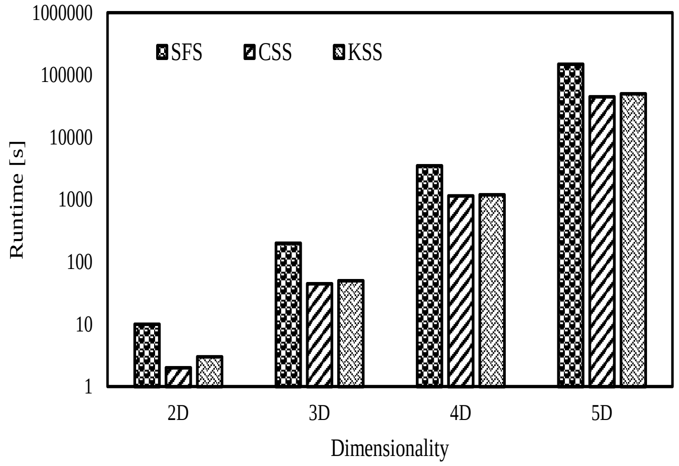
Notice that we get similar standardized results like independent distribution that represents the scalability of CSS and KSS on real dataset for all experiments with FUEL dataset.

These experiments demonstrate that our proposed CSS and KSS are consistently better than the SFS method on both synthetic and real datasets. Therefore, our experiments confirm the effectiveness and scalability of our algorithms.

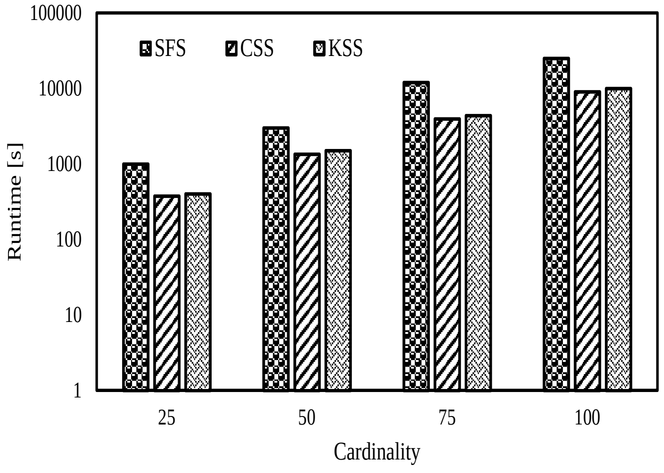
VII. Conclusion

This paper addresses a skyline query for set of objects in a dataset. After analysis various properties of objectsets skyline, we propose an efficient and general algorithm called CSS to compute objectsets skyline. Because of the exponential growth in the number of combinations of tuples, objectsets skyline computation using conventional method is inherently expensive in terms of both time and space. Therefore, in order to prune the search space and improve the efficiency, we have developed two major pruning strategies. Using synthetic and real datasets, we demonstrate the scalability of proposed method. Intensive experiments confirm the effectiveness and superiority of our CSS algorithm.

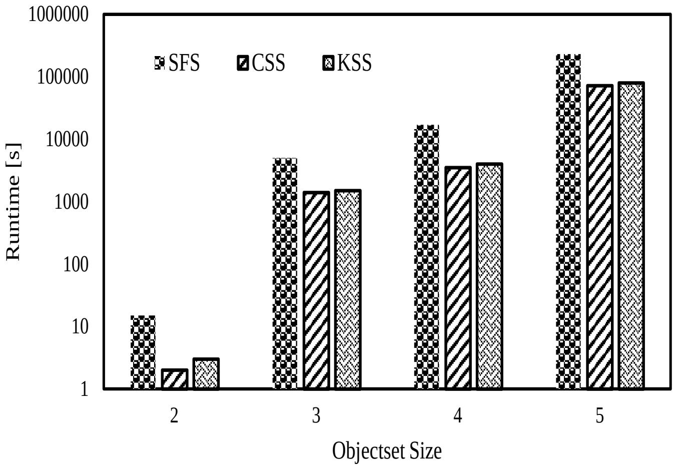
In the future, first, we intend to implement the objectsets skyline problem when the aggregation function is not monotonic. As another direction for future work, we may consider a problem of finding a small number of representative skyline objectsets, similarly to finding a small number of



a) Varying dimensionality



b) Varying cardinality



c) Varying objectset size

Figure 6. Experiments on FUEL dataset

representative skyline objectsets. Third, we want to design more optimized mechanisms for objectsets computation on distributed MapReduce environment.

Acknowledgments

This work is supported by KAKENHI (23500180, 25.03040) Japan. M. A. Siddique is supported by Japanese government "JSPS postdoctoral fellowship for foreign researchers program" and A. Zaman is supported by Japanese government MEXT scholarship.

References

[1] M. A. Siddique, A. Zaman, and Y. Morimoto, "Skyline Objectset: Efficient Selection of Non-dominate Sets from Database," in Proceedings of the 7th International Conference on Advances in Databases, Knowledge, and Data Applications (DBKDA) May 24–29, 2015, Rome, ITALY, 2015, pp. 114–120.

[2] S. Borzsonyi, D. Kossmann, and K. Stocker, "The skyline operator," in Proceedings of the 17th International Conference on Data Engineering (ICDE) April 2–6, 2001, Heidelberg, Germany, 2001, pp. 421–430.

[3] J. Lee, S. Hwang, Z. Nie, and J.-R. Wen, "Navigation system for product search," in Proceedings of the 26th International Conference on Data Engineering (ICDE) March 1–6, 2010, California, USA, 2010, pp. 1113–1116.

[4] T. Lappas and D. Gunopulos, "CREST:Efficient confident search in large review corpora," in Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD) September 20–24, 2010, Barcelona, Spain, 2010, pp. 467–478.

[5] G. Wang, J. Xin, L. Chen, and Y. Liu, "Energy efficient reverse skyline query processing over wireless sensor networks," IEEE Transactions on Knowledge Data Engineering (TKDE), vol. 24, no. 7, 2012, pp. 1259–1275.

[6] L. Zou, L. Chen, M. T. Ozsu, and D. Zhao, "Dynamic skyline queries in large graphs," in Proceedings of the Database Systems for Advanced Applications (DASFAA) April 1–4, 2010, Tsukuba, Japan, 2010, pp. 62–78.

[7] J. Chomicki, P. Godfrey, J. Gryz, and D. Liang, "Skyline with Presorting."

[8] D. Kossmann, F. Ramsak, and S. Rost, "Shooting stars in the sky: An online algorithm for skyline queries," in Proceedings of the 28th International Conference on Very Large Data Bases (VLDB) August 20–23, 2001, Hong Kong, China, 2002, pp. 275–286.

[9] D. Papadias, Y. Tao, G. Fu, and B. Seeger, "Progressive skyline computation in database systems," ACM Transactions on Database Systems, vol. 30, no. 1, 2005, pp. 41–82.

[10] T. Xia, D. Zhang, and Y. Tao, "On Skylining with Flexible Dominance Relation," in Proceedings of the 24th International Conference on Data Engineering (ICDE) April 7–12, 2008, Cancun, Mexico, 2008, pp. 1397–1399.

[11] I.-F. Su, Y.-C. Chung, and C. Lee, "Top-k combinatorial skyline queries," in Proceedings of the Database Systems for Advanced Applications (DASFAA) April 1–4, 2010, Tsukuba, Japan, 2010, pp. 79–93.

[12] X. Guo, C. Xiao, and Y. Ishikawa, "Combination Skyline Queries," Transactions on Large-Scale Data- and Knowledge-Centered Systems VI, vol. 7600, 2012, pp. 1–30.

[13] M. A. Siddique and Y. Morimoto, "Algorithm for computing convex skyline objectsets on numerical databases," IEICE TRANSACTIONS on Information and Systems, vol. E93-D, no. 10, 2010, pp. 0916–8532.

[14] K. C. K. Lee, B. Zheng, H. Li, and W.-C. Lee, "Approaching the skyline in Z order," in Proceedings of the 33th International Conference on Very Large Data Bases (VLDB) September 23–27, 2007, Vienna, Austria, 2007, pp. 279–290.

[15] K.-L. Tan, P.-K. Eng, and B. C. Ooi, "Efficient progressive skyline computation," in Proceedings of the 27th International Conference on Very Large Data Bases (VLDB) September 11–14, 2001, Rome, Italy, 2001, pp. 301–310.

[16] 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 the ACM SIGMOD June 26–29, 2006, Chicago, USA, 2006, pp. 503–514.

[17] X. Lin, Y. Yuan, W. Wang, and H. Lu, "Stabbing the sky: Efficient Skyline computation over sliding windows," in Proceedings of the 21th International Conference on Data Engineering (ICDE) April 5–8, 2005, Tokyo, Japan, 2005, pp. 502–513.

[18] W.-T. Balke, U. Gntzer, and J.-X. Zheng, "Efficient distributed skylining for web information systems," in Proceedings of the 9th International Conference on Extending Database Technology (EDBT) March 14–18, 2004, Crete, Greece, 2004, pp. 256–273.

[19] A. Vlachou, C. Doulkeridis, Y. Kotidis, and M. Vazirgiannis, "SKYPEER: Efficient Subspace Skyline Computation over Distributed Data," in Proceedings of the 23th International Conference on Data Engineering (ICDE) April 15–20, 2007, Istanbul, Turkey, 2007, pp. 416–425.

[20] Y. Tao, X. Xiao, and J. Pei, "Subsky: Efficient Computation of Skylines in Subspaces," in Proceedings of the 22th International Conference on Data Engineering (ICDE) April 3–7, 2006, Georgia, USA, 2006, pp. 65–65.

[21] D. Papadias, Y. Tao, G. Fu, and B. Seeger, "An optimal and progressive algorithm for skyline queries," in Proceedings of the ACM SIGMOD June 9–12, 2003, California, USA, 2003, pp. 467–478.

[22] E. Dellis and B. Seeger, "Efficient Computation of Reverse Skyline Queries," in Proceedings of the 33rd International Conference on Very Large Data Bases (VLDB) September 23–27, 2007, Vienna, Austria, 2007, pp. 291–302.

[23] S. B. Roy, S. Amer-Yahia, A. Chawla, G. Das, and C. Yu, "Constructing and exploring composite items," in Proceedings of the ACM SIGMOD June 6–11, 2010, Indiana, USA, 2010, pp. 843–854.

[24] Q. Wan, R. C.-W. Wong, and Y. Peng, "Finding top-k profitable products," in Proceedings of the 27th International Conference on Data Engineering (ICDE) April 11–16, 2011, Hannover, Germany, 2011, pp. 1055–1066.