Skyline Objectset: Efficient Selection of Complete Non-dominate Sets from Database

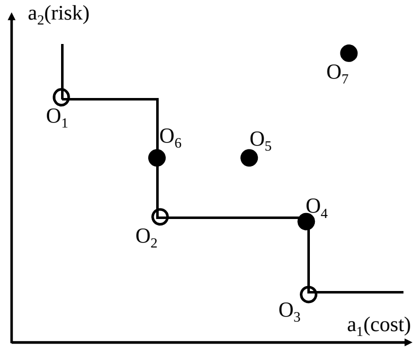
Md. Anisuzzaman Siddique, Asif Zaman, and Yasuhiko Morimoto
Graduate School of Engineering, Hiroshima University
Higashi-Hiroshima, Japan
Email: {siddique@, d140094@, morimoto@mis.}hiroshima-u.ac.jp

***Abstract*—A skyline query retrieves a set of non dominate objects. In this paper, we consider a skyline query for sets of objects (objectsets) in a database. Since a skyline query of objectsets is important in portfolio analysis, privacy aware data analysis, outlier-resistant data analysis, etc., we have previously considered "convex skyline objectsets query", in which we did not select some of skyline objectsets that is not on convex hull. To solve the shortcoming, we propose an efficient algorithm to select complete skyline objectsets in this paper. We investigated the properties of objectset skyline computation and develop two major pruning conditions to avoid unnecessary objectset enumerations as well as comparisons among them. We conduct a set of experiments to show the meaningfulness and scalability of the proposed skyline objectset algorithm.**

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

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

I. Introduction

Skyline query [1] and its variants are functions to find 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 is not affected by potentially different scales at different dimensions (risk unit or cost unit in the example of Figure 1); only the order of the dimensional projections of the objects is important. Skyline query has broad applications including product or restaurant recommendations [2], review evaluations with user ratings [3], querying wireless sensor networks [4], and graph analysis [5]. Algorithms for computing skyline objects have been discussed in the literature [6] [7] [8] [9].

One of the known weaknesses of the skyline query is that it can not answer various queries that require us to analyze 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 main issue studied in this paper.

We consider a skyline query for sets of objects (objectsets) in a database. Let k be the number of objects in each set and n be the number of objects in the dataset. The number of sets in the dataset amounts to ${}_nC_k$. We propose an efficient algorithm to compute skyline of the ${}_nC_k$ sets.

Assume an investor has to buy two stocks. In Figure 1, conventional skyline query outputs $\{O_1, O_2, O_3\}$, which doesn't provide sufficient information for the set selection problem. Users may want to choose the portfolios which are not dominated by any other sets in order to minimize the total costs and risks. Figure 2 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 and thus they are the answers for the objectset skyline query. Again, if the investor wants to buy three stocks then the objectset skyline query 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. The objectsets skyline operator was introduced by Siddique and Morimoto in 2010 [10]. They tried to find the skyline objectsets that are on the convex hull enclosing all the objectsets, yet it misses some skyline objectsets, which are not on the convex hull. 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 difficult 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 problem 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. 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.

We propose an algorithm to resolve the objectsets skyline query problem. It progressively prunes the objectsets that are impossible to be the objectsets skyline result, and uses a filtering mechanism to retrieve the skyline objectsets without enumerating all objectsets. We develop two pruning strategies to avoid generating a large number of unpromising objectsets. The efficiency of the algorithm is then examined with experiments on a variety of synthetic and real datasets.

The rest of this paper is organized as follows. Section II reviews the related work. Section III presents the notions and properties for objectsets as well as the problem of objectsets skyline. In Section IV, we provide details of our proposed algorithm with appropriate examples and analysis. We experimentally evaluate the proposed algorithm in Section V under a variety of settings. Finally, Section VI concludes the paper and introduces our future works.

II. Related Work

Our work is motivated by previous studies of skyline query processing as well as objectsets skyline query processing.

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 [1]. 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 [6]. Among index-based methods, Tan et al. proposed two progressive skyline computing methods Bitmap and Index [13]. 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 [8]. 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 [14]. 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 distributedly, it is impossible to handle them in a centralized fashion [15]. Balke et al. first mined skyline in a distributed environment by partitioning the data vertically [16]. 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 [17]. Tao et al. discuss skyline queries in arbitrary subspaces [18]. More skyline variants such as dynamic skyline [19] and reverse skyline [20] 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" [10]. 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].

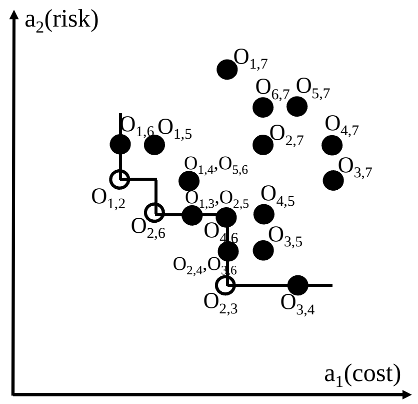
Siddique and Morimoto studied the "convex skyline objectset" problem. It is known that the objects on the lower (upper) convex hull, denoted as CH, is a subset of the objects on the skyline, denoted as SKY. Every object in CH can minimize (maximize) a corresponding linear scoring function on attributes, while every object in SKY can minimize (maximize) a corresponding monotonic scoring function [1]. They aims at retrieving the objectsets in CH, however, we focuses 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 Querues" 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

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

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

a) Sets of 2 Stocks



b) Skyline of 2 Stocks

Figure 2. Objectset skyline problem

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 [21] [22]. 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 [22]. They construct non-dominate products by assigning prices to the new products that are not given beforehand like the existing products.

III. Preliminaries

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.

Skyline

An object $O_i \in D$ is said to be a *skyline object* of D, if and only if there 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:

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

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.,

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

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 $|S|$.

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'.

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 Figure 2(a), 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(b)).

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-objcetsets 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 I. 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	$O_{1,\cdots,6}$

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.

Initially, $k = 1$ and we can compute 1-objectsets skyline using any conventional algorithms. In this paper, we use SFS method proposed in [6] to compute 1-objectsets skyline and receive the following domination relation table called $domRelationTable$.

For the objectsets skyline query problem, the 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. As Figure 2(a) shows, $|S| = 21$ (${}_7C_2$) 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 eliminate many non-promising objectsets without composing them.

Theorem 1. *If all k member of an objectset s are in $DS_k(D)$, where $DS_k(D) = DS_1(D) \cup \cdots \cup DS_k(D)$, 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 $DS_k(D)$, then there must be k distinct dominant objectsets for each member of s. Suppose they are $\{O'_1, \cdots, O'_k\}$ and construct an objectset s'. Now, according to dominance relationship $s' \leq s$, which contradict initial assumption $s \in Sky_k(D)$. Hence, a k-objectsets contains at least one skyline objectset. ■

Dominance relation given in Table I retrieves $DS_1(D) = \{O_4, O_5, O_6, O_7\}$. By using Theorem 1 and $k = 2$ we can safely prune ${}_4C_2 = 6$ objectsets such as $\{O_{4,5}, O_{4,6}, O_{4,7}, O_{5,6}, O_{5,7}, O_{6,7}\}$ for $Sky_2(D)$ query without composing them. The remaining objecsetsets number 15 (21-6) is still too large for our running example. However, 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 their super objectset with S_3 also dominates, i.e., $S_1S_3 \leq S_2S_3$ is true.*

Proof: Suppose $S_1S_3 \leq S_2S_3$ is not true. After eradicate S_3 from both objectsets we get $S_1 \leq S_2$, which contradict our assumption. Thus, if $S_1 \leq S_2$ and S_3 is another objectset then $S_1S_3 \leq S_2S_3$ is always true. ■

Theorem 2 gives us another opportunity to eliminate huge number objectsets without computing them. Table I shows that object O_4 is dominated by O_2 and O_3. By considering $\{O_1, O_2, O_3\}$ as common objects and using Theorem 2 without any computation as well as any comparisons we get $O_{1,2} \leq O_{1,4}, O_{2,3} \leq O_{2,4}, O_{2,3} \leq O_{3,4}$ dominance relation for 2-objectsets. Similarly, object O_5 is dominated by $\{O_2, O_6\}$ and using $\{O_1, O_2, O_3\}$ as common objects gives us $O_{1,2} \leq O_{1,5}, O_{2,3} \leq O_{3,5}, O_{2,6} \leq O_{2,5}$. For $O_2 \leq O_6$ and $\{O_1, O_3\}$ as common objects produces $O_{1,2} \leq O_{1,6}, O_{2,3} \leq O_{3,6}$. Finally, for $\{O_{1,\cdots,6} \leq O_7\}$ gives guarantee of the following dominance relationship $\{O_{1,2} \leq O_{1,7}, O_{1,2} \leq O_{2,7}, O_{1,3} \leq O_{3,7}\}$. Thus according to Theorem 2 we can safely prune 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 $Sky_2(D)$ query without composing them. Actually, CSS algorithm will compose remaining (15 - 11) four objecsets such as $\{O_{1,2}, O_{1,3},\ O_{2,3}, O_{2,6}\}$ and perform domination checks among them. After performing domination check it retrieves $\{O_{1,2}, O_{2,3}, O_{2,6}\}$ as $Sky_2(D)$. However, during this procedure CSS also updates the dominance relation table for 2-objectsets as shown in II.

Dominance relation table II retrieves $DS_2(D) = \{O_{1,3}, O_{1,4}, O_{1,5}, O_{1,6}, O_{1,7}, O_{2,4}, O_{2,5}, O_{2,7}, O_{3,4}, O_{3,5}, O_{3,6}, O_{3,7}\}$. When $k = 3$, then for any conventional skyline algorithm needs to dominance relation check among $|S| = 35$ (${}_7C_3$) objectsets. However, according to Theorem 1 we are enough lucky and need not to compose any 3-objectsets if the distinct 3 objects in $DS_1(D) \cup DS_2(D)$. For $DS_1(D)$, CSS does not compute ${}_4C_3 = 4$ objectsets such as $\{O_{4,5,6}, O_{4,5,7}, O_{4,6,7}, O_{5,6,7}\}$. Whereas for $DS_1(D) \cup DS_2(D)$ we have checked that CSS pruned another 22 objectsets. After implementing Theorem 1 successfully the remaining objectset number is 9 (35-26). Applying Theorem 2 CSS does not need to compute another 5 objectets. Finally, proposed algorithm will compose only five objecsets such as $\{O_{1,2,3}, O_{1,2,6}, O_{2,3,4}, O_{2,3,6}\}$ and perform domination check among them to obtain $Sky_3(D)$. After domination check CSS retrieves $\{O_{1,2,3}, O_{1,2,6}, O_{2,3,4}, O_{2,3,6}\}$ as $Sky_3(D)$. CSS will continue similar iterative procedure for the rest of the k values to compute $Sky_k(D)$.

V. Performance Evaluation

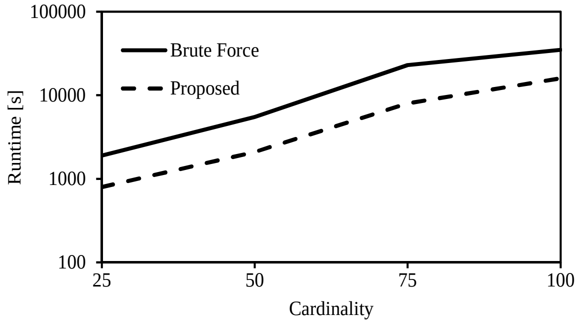
We conducted a set of experiments with different dimensionalities (m), data cardinalities (n), and objectset size (k) to evaluate the effectiveness and efficiency of our proposed method. All experiments are run on a computer with Intel Core i7 CPU 3.4GHz and 4 GB main memory. We compiled the source codes under Java V8 in Windows 8 32-bit operating system. We also compared the performance with Brute Force method. To make the comparison fair, we have exclude all the pre-processing cost,i.e., cost of objectset generation. Each experiment is repeated five times and the average result is considered for performance evaluation.

A. Performance on Synthetic Datasets

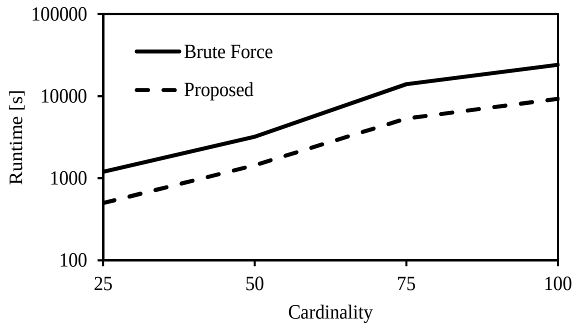
Three data distributions such as correlated, anti-correlated, and independent data distribution are considered to do all of the experiments. The results are shown in Figure 3, 4, and 5. All figures are shown in a logarithmic scale. From the experimental results we observe a pattern that the speedup of proposed method over Brute Force is 2 to 3 times faster.

TABLE II. domRelationTable for 2-objectsets

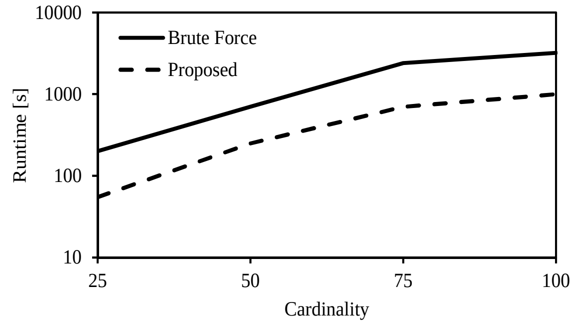
$Objectset$	$Dom.\ Objectset$	$Objectset$	$Dom.\ Objectset$	$Objectset$	$Dom.\ Objectset$
$O_{1,2}$	$\varnothing$	$O_{1,7}$	$O_{1,2}$	$O_{2,7}$	$O_{2,3}$
$O_{1,3}$	$O_{2,6}$	$O_{2,3}$	$\varnothing$	$O_{3,4}$	$O_{2,3}$
$O_{1,4}$	$O_{1,2}$	$O_{2,4}$	$O_{2,3}$	$O_{3,5}$	$O_{2,3}$
$O_{1,5}$	$O_{1,2}$	$O_{2,5}$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



a) Anti-correlated



b) Independent



c) Correlated

Figure 3. Performance for different cardinality

Effect of Cardinality

For this experiment, we fix the data dimensionality m to 4, objectset size k to 3, and vary dataset cardinality n. n takes the values of 25, 50, 75, and 100 that means total objectset size respectively become 2.3k, 19.6k, 67.5k, and 161.7k. Figure 3(a), (b), and (c) reports the performance on correlated, independent, and anti-correlated datasets. Where both of the methods are affected by data cardinality. If the data cardinality increases then their performances decreases. The result shows that proposed method significantly outperforms the Brute Force method. However, the performance of Brute Force method degrades rapidly as the the dataset size increases, especially for anti-correlated data distribution. This represents that proposed method can successfully avoid objectset composing as well as many unnecessary comparisons.

Effect of Dimensionality

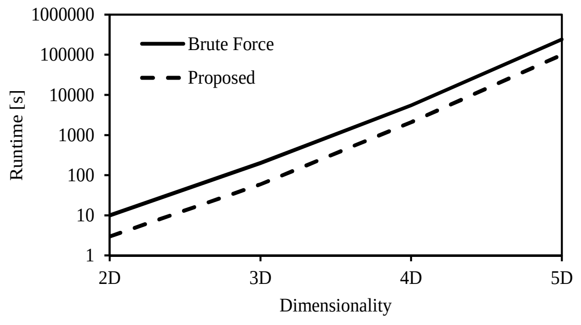
We study the effect of dimensionality on our technique. We fix the data cardinality n to 50, objectset size k to 3 and vary dataset dimensionality m ranges from 2 to 5. The elapsed time results for this experiment are shown in Figure 4(a), (b), and (c). The result exhibits that as the dimension increases the performance of the both methods becomes slower. This is because for high dimension the number of non dominant objectset increases as a result the performance of both methods degraded. The proposed algorithm achieves a satisfactory running time even when the dimension size is large. However, the result on correlated data dataset is 9 times and 16 times faster than independent and anti-correlated data dataset respectively.

Effect of Objectset Size

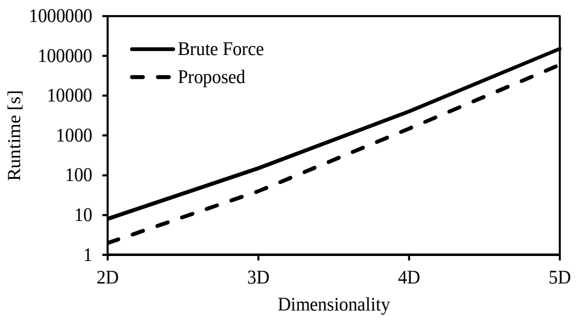
In another experiment, we study the performance of proposed method under various objectset size k. We fix the data cardinality n to 50 and dataset dimensionality m to 4. The results are reported in Figure 5(a), (b), and (c). The result indicate that as the objectset size k increases the performance of the both methods becomes slower. However, the result of Brute Force method is much worse than that of proposed method when the value of objectset size k is greater than 1. This is because for $k = 1$ the proposed method use SFS algorithm to construct $domRelationTable$, and performing domination check. After that for higher s it does not required to compose all objectset as well as succeeded to avoid huge number of unnecessary comparisons.

B. *Performance on Real Dataset*

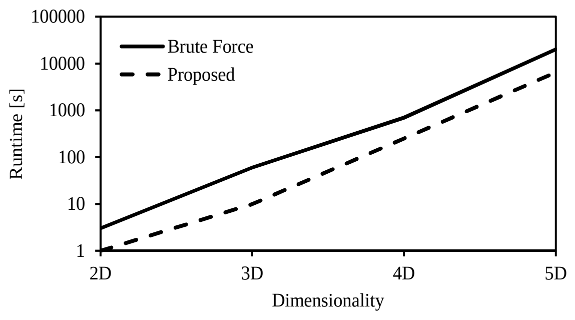
To evaluate the performance for real dataset, we use the FUEL dataset which is extracted from "`www.fueleconomy.gov`". FUEL dataset is 24k 6-dimensional objects, in which each object stands for the performance of a vehicle (such as mileage per gallon of gasoline in city and highway, etc). For this dataset attribute



a) Anti-correlated

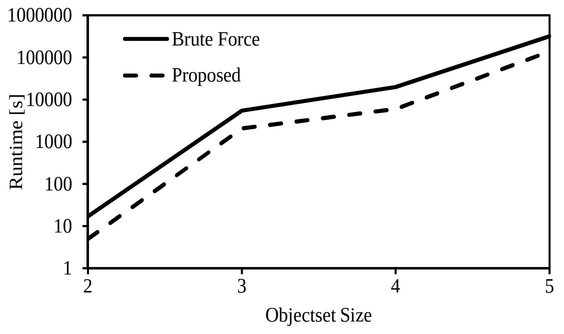


b) Independent

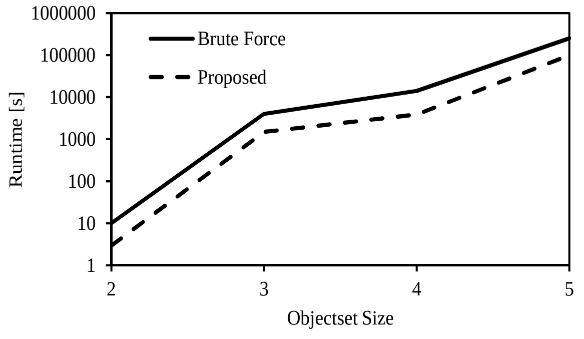


c) Correlated

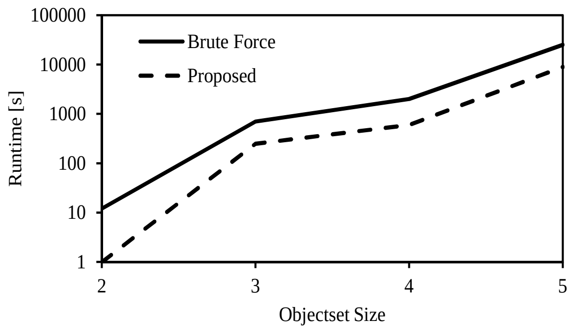
Figure 4. Performance for different data dimension



a) Anti-correlated



b) Independent



c) Correlated

Figure 5. Performance for different objectset size

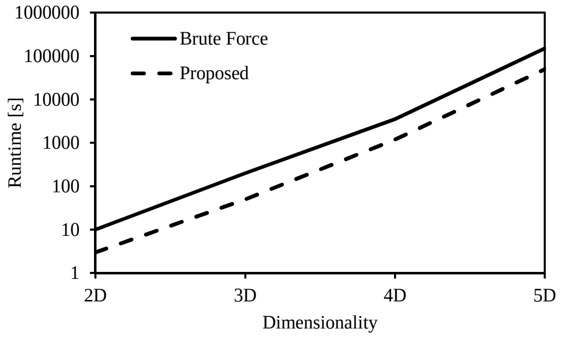
domain range is [8, 89] and we conduct the following experiments.

For FUEL dataset, we performed similar experiments like synthetic datasets. For cardinality experiment, we set the dimensionality m to 4 , objectset size k to 3, and vary dataset cardinality n from 25 to 100. Result is shown in Figure 6(a). To study dimensionality, we fix the data cardinality n to 50, objectset size k to 3 and vary dataset dimensionality m ranges from 2 to 5. Figure 6(b) shows the result. In the final experiment, we study the performance under various objectset size k. We fix the data cardinality n to 50 and dimensionality m to 4. The result is reported in Figure 6(c). For all experiments with FUEL dataset, we obtain similar result like independent dataset that represents the scalability of the proposed method on real dataset. However, in all experiments with real FUEL dataset proposed method outperform than Brute Force method.

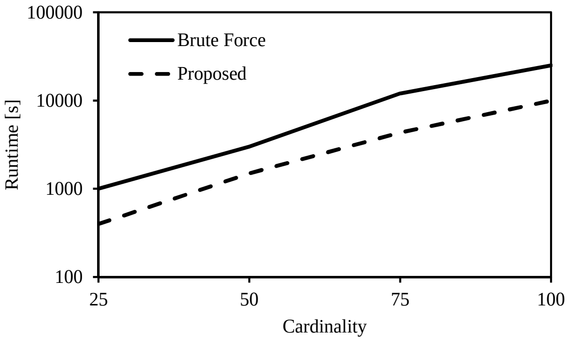
VI. Conclusion

This paper addresses a skyline query for set of objects in a dataset. We propose an efficient and general algorithm called CSS to compute objectsets skyline. 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.

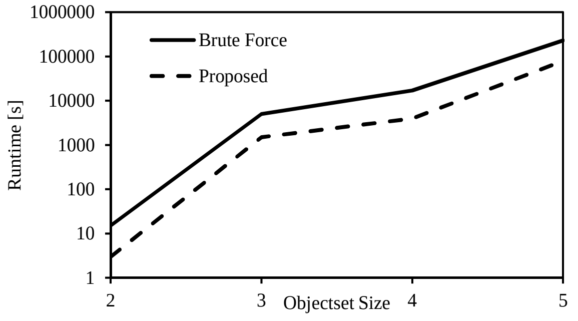
It is worthy of being mentioned that this work can be expanded in a number of directions. First, how to solve the problem when the aggregation function is not monotonic. Secondly, to design more efficient objectsets computation on distributed MapReduce architectures. Finally, to find small number of representative objectsets is another promising future research work.



a) Varying dimensionality



b) Varying cardinality



c) Varying objectset size

Figure 6. Experiments on FUEL dataset

ACKNOWLEDGMENTS

This work is supported by KAKENHI (23500180, 25.03040) Japan.

REFERENCES

[1] 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.

[2] 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.

[3] 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.

[4] 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.

[5] 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.

[6] J. Chomicki, P. Godfrey, J. Gryz, and D. Liang, “Skyline with Presorting,” in Proceedings of the 19th International Conference on Data Engineering (ICDE) March 5–8, 2003, Bangalore, India, 2003, pp. 717–719.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[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] 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.

[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,” in Proceedings of the ACM SIGMOD June 26–29, 2006, Chicago, USA, 2006, pp. 503–514.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] 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.