J Inf Process Syst, Vol.13, No.5, pp.1277~1298, October 2017
https://doi.org/10.3745/JIPS.04.0049

ISSN 1976-913X (Print)
ISSN 2092-805X (Electronic)

JOURNAL OF INFORMATION PROCESSING SYSTEMS JIPS

Efficiently Processing Skyline Query on Multi-Instance Data

Shu-I Chiu* and Kuo-Wei Hsu*

Abstract

Related to the maximum vector problem, a skyline query is to discover dominating tuples from a set of tuples, where each defines an object (such as a hotel) in several dimensions (such as the price and the distance to the beach). A tuple, an instance of an object, dominates another tuple if it is equally good or better in all dimensions and better in at least one dimension. Traditionally, skyline queries are defined upon single-instance data or upon objects each of which is associated with an instance. However, in some cases, an object is not associated with a single instance but rather by multiple instances. For example, on a review website, many users assign scores to a product or a service, and a user's score is an instance of the object representing the product or the service. Such data is an example of multi-instance data. Unlike most (if not all) others considering the traditional setting, we consider skyline queries defined upon multi-instance data. We define the dominance calculation and propose an algorithm to reduce its computational cost. We use synthetic and real data to evaluate the proposed methods, and the results demonstrate their utility.

Keywords

Multi-Instance Data, Product Search, Ranking, Recommendation, Skyline Query Processing

1. Introduction

A skyline query is to discover dominating and therefore interesting tuples from a database. A tuple is usually composed of several fields or dimensions, and it dominates another tuple if it is equally good or better in all dimensions and better in at least one dimension. A skyline tuple is one that is in the result of a skyline query or is part of the answer of a skyline query, and it is one that dominates some others and is not dominated by any other. An object could represent a product or a service, such as a hotel or a restaurant, and a tuple is an instance of an object and defines an object in several dimensions, such as the price and the distance to the beach or such as the service, the food, and the décor.

Efficiently processing skyline queries is important and valuable. In theory, skyline queries are related to the maximum vector problem and multi-criteria decision making. In practice, they have seen a wide range of applications, such as product or service search, ranking, and recommendation systems. Let us consider the following two examples showing how skyline queries can be used and can be useful [1].

Example 1, adapted from the classic example given in [2]: a person plans to have a vacation in a city by a sea and he or she is looking for a hotel that is cheap and close to the beach. However, the hotels

Manuscript received May 16, 2017; first revision July 18, 2017; accepted July 20, 2017.
Corresponding Author: Kou-Wei Hsu (hsu@cs.nccu.edu.tw)
* Computer Science, National Chengchi University, Taipei, Taiwan ({d9706, hsu}@cs.nccu.edu.tw)

near the beach are more usually expensive, while and the cheap ones are usually far away from the beach. The person is unable to decide which one is the best hotel for him or her. It will be easier for the person to make the decision if he or she can have a list of interesting hotels, where interesting hotels are those that are not worse than any other hotel in the price and the distance to the beach. A hotel is an object, and it is represented or defined by a tuple in the database, where a tuple is composed of two dimensions, namely the price and the distance to the beach. The list of interesting hotels is the answer of a skyline query, and those interesting hotels are skyline tuples. After having the list, the person can make the decision according to his or her personal preferences towards the two factors, namely the price and the distance to the beach. As explained in [2], there will be a hotel that is best for the person, no matter how he or she weighs his or her personal preferences towards the two factors. This example also implies that skyline queries are also related the problem of multi-criteria decision making.

Example 2, adapted from the example given in [3]: A person is referring to a restaurant from a guidebook, in which each restaurant is reviewed with respect to three criteria, namely the service, the food, and the décor. The person is looking for a restaurant that provides the best service, the best food, and the best décor and is the lowest priced. However, there is no restaurant that is better than all others on every criterion and cheaper than all others on the price; if there is one, the guidebook will become unnecessary. Although there is no restaurant that is absolutely best, the person can at least exclude from consideration the uninteresting ones (non-skyline tuples) each of which is worse than some others on every criterion.

Traditionally, an object is associated with exactly a tuple or an instance, and the answer of a skyline queries is in fact a set of interesting objects, each of which is associated with a dominating tuple. Therefore, traditional skyline queries are defined upon single-instance data. The data in Example 1 is single-instance data, and so is that in Example 2. In some cases, however, an object is not represented or defined by a single tuple but rather by multiple tuples. For example, on a review website, many users assign scores to and/or write reviews of a product or a service. A user's score is a tuple or an instance, and a product or a service is an object that is associated with multiple tuples or instances. We call such data multi-instance data. In Example 1, if hotels are reviewed by visitors with respect to other criteria, especially subjective ones, such as the service and the convenience, then the data is multi-instance data and existing algorithms (for single-instance data) can no longer be used. In Example 2, if reviews are not from the critics (who contribute to the guidebook) but from general users, then a restaurant will possibly be reviewed by many users and be associated with many tuples in the database. The data is multi-instance data and, again, existing algorithms cannot be used. What we are interested in is not tuples or instances but objects. If we directly apply existing algorithms to, for example, the data on the review website, then we will have "dominating scores" (which are not quite meaningful) instead of interesting products or services.

As e-commerce still booms worldwide, the studies on and the applications of skyline queries are blooming. As the number of review websites increases, the demand for discovering skyline objects from multi-instance data becomes stronger. In order to use an existing algorithm to answer or process skyline queries on multi-instance data, we need to do data transformation first. We can average the values of all instances associated with an object. We can do so for each object, and then we can transform multi-instance data into single-instance data, where an instance in the latter is an averaged value. The problem with this approach, an indirect approach, is the same as the problem with any other approach in which the vulnerable averages or means are used. We propose a direct approach. We

consider skyline queries defined upon multi-instance data. First of all, we define the dominance calculation, which is essential for the comparison of tuples and the determination of skyline objects. We propose an algorithm to answer or process skyline queries on multi-instance data. Such data is different from the traditional data (that is, single-instance data). The typical skyline query processing cannot be used on the multi-instance data. Nowadays, data on many review websites belongs to multi-instance data. We propose a formal definition of multi-instance data, and we attempt to directly process the skyline query on multi-instance data. In addition, we propose methods that can reduce the computational cost and consequently boost runtime performance. Finally, the results that we obtain from experiments which synthetic and real data are used in demonstrating the utility of the proposed methods. We use the generated data to examine our methods, BNL algorithm, and D&C algorithm in order to compare with other studies [4,5]. The contributions are two-fold: First, we define the dominance relation on multi-instance data and propose a way speed up the skyline query processing; second, for example, review websites can display not only the average of values of products or services (objects) but also interesting individual reviews (instances) by using our methods. In addition, our proposed functions can finish the skyline computation early.

This paper extends [6]. Compared to [6], this paper generalizes the type of data, reorganizes the theoretical content, provides new experiment results and more discussion, and presents potential applications.

The remainder of this paper is organized as follows. Section 2 reviews related works, defines the dominance calculation and presents the algorithms. Section 3 reports a systematic performance study on synthetic and real data. Finally, Section 4 concludes this paper.

2. Materials and Methods

2.1 Related Works

The skyline operation is proposed to extend database systems [1]. This operation is to discover a set of interesting tuples from a potentially large set of tuples. The basic way to compute the skyline is to apply block-nested-loop (BNL) algorithm and compare every tuple with every other tuple [2]. In [2], the authors also use divide-and-conquer (D&C) algorithm [4] to implement the skyline query. Two progressive techniques are proposed in [5], and they are the Bitmap and the Index techniques. A database index is a data structure that improves the speed of data retrieval operations on a database table at the cost of additional storage space and writes to maintain the index data structure. An index is used to quickly locate data without having to search every row in a database table every time a database table is accessed. Nearest neighbor (NN) uses the results of nearest neighbor search to partition the data universe recursively [5]. NN executes a nearest neighbor query on R-trees. In [5], the authors mention that NN has some desirable features (such as high speed for returning the initial skyline tuples) but presents several inherent disadvantages (such as the need for duplicate elimination if the dimension is larger than 2, multiple accesses of the same node, and large space overhead). So, the authors developed branch-and-bound skyline (BBS) [5,7].

Some works sort the input data to speed up the performance of queries [1,3,8-13]. The sorting-based algorithms aim to optimize pivot ordering to prune non-skyline tuples early. The first sorting-based

algorithm is sort-filter-skyline (SFS) algorithm [3]. In [7], the authors define the monotone scoring function (ordered from highest to lowest score) which is a topological sort with respect to the skyline dominance partial relation. We also define a monotone function in this paper. Godfrey et al. mention that the maximal vector problem has been rediscovered in the database context with the introduction of skyline query [9]. Computing the skyline is known as the maximum vector problem [8,14,15]. In [9], the authors present a new algorithm for maximal vector computation, linear elimination sort for skyline (LESS), that combines aspects of SFS, BNL, and fast linear expected-time (FLET) [13] but does not contain any aspects of D&C. LESS must sort the tuples initially; LESS is an optimized version of SFS [9]. In [12], sort and limit skyline algorithm (SaLSa) exploits the idea of presorting the input data so as to effectively limit the number of tuples to be read and compared. The SaLSa strives to avoid scanning the complete set of sorted tuples and its feature is the ability of computing the result without having to apply dominance tests to all the tuples in the input relation [12]. Many algorithms such as SFS, LESS and SaLSa need to sort tuples first, and so do our methods.

In addition, the partitioning-based algorithms aim to group tuples into sub-regions which are used for region-based dominance tests. D&C [2] simply divides the problem into multiple sub-problems and merges the local skyline tuples into global ones. Zhang et al. [10] propose an object-based space partitioning (OSP) scheme, which recursively divides the data space into separate partitions with respect to a reference skyline tuple and facilitates progressive retrieval in high dimensional spaces.

Table 1 summarizes the features of skyline query processing algorithms in the literature, and Table 2 summarizes related works according to their features.

Table 1. The features of skyline query processing algorithms

Features	Description	Abbreviation
Sorting technique	Researchers sort the input data by using some functions	ST
Dominance checking approach	Researchers use some methods to reduce calculations	DA
Indexing technique	Researchers build index to speed up	IT
Application	Researchers evaluate their algorithms with real data	Ap

Table 2. The summary of related works

Ref.	Algorithm	ST	DA	IT	Ap
[4]	BNL, D&C	No	No	Yes	Yes
[5]	Bitmap, Index	Yes	No	Yes	No
[3,8]	SFS	Yes	No	No	No
[11]	NN	Yes	No	Yes	Yes
[5,9]	BBS	Yes	Yes	Yes	Yes
[9]	LESS	Yes	Yes	No	No
[12]	SaLSa	Yes	Yes	No	No
[10]	OSP	Yes	Yes	Yes	No

In the typical application to which our methods can be applied, users assign scores to items or objects and then these scores are transformed into multi-instance data by the proportion method. An object can contain a series of probability values. Many prior works show that skyline query is very useful in

multi-criteria decision making applications [3-13]. Uncertainty in data is inherent in many applications such as sensor networks, scientific data management, data integration, where data take different values with probabilities [16]. Probabilistic data are unavoidable in some important applications. The first work on supporting the skyline query on such data, called p-skyline, is reported in [19], in which the authors consider analyzing professional basketball players using multiple technical statistics criteria and attempt to find the player who can achieve the best performance in all aspects. In [19], the authors propose a probabilistic skyline model in which an uncertain tuple may take on a probability of being on the skyline called p-skyline. Given a threshold p ($0 \le p \le 1$), the p-skyline is the set of uncertain objects, each of which takes a probability of at least p to be on the skyline [17,18]. In [17], the definition of an instance is different from that in this paper. Atallah and his colleagues [16,19] propose a general probabilistic skyline query that takes into account different user utilities without any restriction, but they do not use any probability threshold. Liu et al. [1] propose a new uncertain skyline model called u-skyline, and it aims to return an uncertain skyline answer set from a complementary perspective to p-skyline. Furthermore, p-skyline returns individual data tuples with non-dominance probabilities greater than or equal to a specified threshold [1], while u-Skyline focuses on returning an answer set that forms a valid skyline with the maximum probability [1].

Most works assume that uncertainty exists only in attribute values [20]. Zhang et al. [20] address the skyline probability computation problem in scenarios where uncertainty resides in attribute preferences instead of values. The approach used in [20] assumes independent object dominance. The previous works discuss probabilistic skylines and skyline query for probabilistic data; we summarize these works according to their features in Table 3. Our work is different from others in that it is for multi-instance data instead of the traditionally defined uncertain data. The dominance relation between two objects in this paper is the sum of the probabilities that the higher score can dominate the lower score. In p-skyline, a probability for a tuple is defined by aggregating over all the possible worlds within which the tuple is dominated. We calculate the dominance relation between two objects and then determine the one that could potentially be on the skyline. If the determined object is not dominated by others, it is a skyline object and is returned as an answer to the query.

We summarize comparisons between our work and related works in Table 4. Table 4 also describes the originality of our work.

Considering the increasingly large amount of data, Cosgaya-Lozano et al. [22] show that parallel computing is an effective method to speed up the skyline query processing on large datasets. Many works consider parallelized methods that utilize multiple processors or obtain useful partitions of the dataset for parallel processing [23-26]. We focus on effectively processing skyline query on a special type of data. Nevertheless, part of future work could be to parallelize our methods.

Table 3. The summary of related works on probabilistic data

Ref.	Algorithms	ST	DA	IT	Ap
[17]	The bottom-up and the top-down algorithms	Yes	No	Yes	Yes
[16,19]	Weighted dominance counting (WDC)	Yes	Yes	No	No
[21]	Skyline feature algorithm (SFA)	Yes	Yes	Yes	No
[18]	Bottom-up and top-down hybrid algorithm	Yes	Yes	Yes	Yes
[2]	Dynamic programming search algorithm	Yes	Yes	No	No
[20]	Monte Carlo estimation algorithm	Yes	Yes	No	Yes
This paper	Our methods (Sec. 2.3)	Yes	Yes	No	Yes

Table 4. The summary of comparisons between our work and related works

Features	Our work	Related works
Sorting technique	Ours and theirs sort the input data to reduce the computational cost and speed up the performance	[1,3,8-13]
Instance	The instance in their definition is different from that in ours	[17]
Dominance relation	They define a probability for each tuple by aggregating over all the possible worlds within which the tuple is dominated, while we define the dominance relation to be the sum of the probabilities that the higher score can dominate the lower score	[18]
Monotone function	They define the monotone scoring function, and that is different from what we use	[3,9]
Skyline object	If an object is not dominated by others, it is a skyline object and is returned as an answer to a query; this is consistent with the skyline query on certain data	[2-16]

2.2 Definitions

In this subsection, we formally define the dominance relation and skyline query on multi-instance data and review some relevant studies on skyline query.

Definition 1. *(Tuple).* A tuple is a record in a database. It is usually composed of several fields. For example, a tuple could contain a score (usually a vector) given by a user.

Definition 2. *(Dimension).* A dimension is a field, attribute or criterion. A tuple is usually composed of several dimensions.

Definition 3. *(Object).* An object is usually composed of several dimensions, and for example, it could be a restaurant, product or service to which many users assign scores. An object is also called an item.

Definition 4. *(Instance).* A score (usually a vector) given by a user is an instance of an object representing, for example, a restaurant, product or service on a review website.

Definition 5. *(Multi-instance data).* Many users assign scores to and/or write reviews of, for example, a restaurant, product or service. An object is associated with multiple instances. Such data is defined as multi-instance data.

Definition 6. *(Dominance relation).* Let U and V be two tuples in a d-dimensional space. The dominance relation is presented on the preference attributes (from 1 to d). We assume that bigger values are better. For every dimension i ($1 \le i \le d$), if $U_i \le V_i$ and there exists a dimension j ($1 \le j \le d$) such that $U_j < V_j$, then V dominates U, denoted by $U \prec V$.

We propose to identify objects by using the skyline query on multi-instance data. First, we transform score data into multi-instance data for each object.

Let us consider the following example: a restaurant is reviewed by many users and the scores range from 1 to 5. Table 5 presents a 3-dimensional score dataset containing 14 scores (from 14 users). In Table 5, P_{id} and C_{id} are identifiers for a restaurant and a user, respectively, and 3 dimensions are for food, service, and décor. We group these scores by P_{id} and then transform the score data in Table 5 into the multi-instance data in Table 6. Table 6 show multi-instance data for R_1 and R_2, respectively.

If a random variable X is discrete (i.e. it takes a value from a specific set of n values x_i, $1 \le i \le n$), then $P(X = x_i) = p(x_i)$ where $p(x_i)$ is the probability mass function and $p(x_i)$ denotes the probability of score being x_i. When x_i is an assigned score, $p(x_i)$ is calculated by (1):

$$\frac{the\ number\ of\ users\ assigning\ score\ x_i}{\sum users\ assigning\ scores\ to\ an\ object} \tag{1}$$

An object is described by a probability mass function in the data space. We perform data transformation. So, this object R_1 in the example in Table 6 is denoted by <(1,0), (2,0), (3,0.1), (4,0.4), (5,0.5), (1,0), (2,0), (3,0), (4,0.5), (5,0.5), (1,0), (2,0.2), (3,0.5), (4,0.2), (5,0.1)>.

Table 5. An example score dataset

P_{id}	C_{id}	Food score	Service score	Décor score
R_1	C_1	3	4	3
R_1	C_2	5	5	4
R_2	C_3	3	2	4
R_1	C_4	5	4	3
R_1	C_5	5	5	4
R_1	C_6	5	5	3
R_2	C_7	2	2	3
R_1	C_8	5	5	3
R_1	C_9	4	5	2
R_2	C_{10}	3	1	2
R_1	C_{11}	4	4	5
R_2	C_{12}	2	3	5
R_1	C_{13}	4	4	3
R_1	C_{14}	4	4	2

Table 6. The multi-instance data transformed from the example score dataset

	P_{id}	C_{id}	Food score	Service score	Décor score
Multi-instance data for R_1	R_1	C_1	3	4	3
	R_1	C_2	5	5	4
	R_1	C_4	5	4	3
	R_1	C_5	5	5	4
	R_1	C_6	5	5	3
	R_1	C_8	5	5	3
	R_1	C_9	4	5	2
	R_1	C_{11}	4	4	5
	R_1	C_{13}	4	4	3
	R_1	C_{14}	4	4	2
Multi-instance data for R_2	R_2	C_3	3	2	4
	R_2	C_7	2	2	3
	R_2	C_{10}	3	1	2
	R_2	C_{12}	2	3	5

Definition 7. (*Dominance relation on multi-instance data*). Let U and V be two 1-dimensional objects: $U = \langle (1, p_u(1)), \cdots, (i, p_u(i)), \ldots (n, p_u(n)) \rangle$ and $V = \langle (1, p_v(1)), \cdots, (i, p_v(i)), \ldots (n, p_v(n)) \rangle$. Scores range from 1 to n (which is the highest score). Let $p(x_i)$ denotes the probability of score being x_i, $p(x_i) \geq 0$, and $\sum_{i=1}^{n} p(x_i) = 1$. Therefore, $\sum_{i=1}^{n} p_u(i) = 1$ *and* $\sum_{i=1}^{n} p_v(i) = 1$. Let $Pr[U > V]$ denote the probability that the object U dominates the object V, and $Pr[U > V] = \sum_{i=1}^{n} (p_u(i) \times \sum_{j=1}^{i-1} p_v(j))$.

In the skyline query, a point P_i dominates another point P_j, if and only if P_i is as good or better than P_j in all dimensions and better in at least one dimension. We apply the same concept to the object data containing probability values. We define that the probability values of the higher score can dominate the probability values of the lower score. We explain the dominance relation by using the example below. Let U and V be two 1-dimensional objects with possible scores from 1 to 5: U = <(1,U_1), (2,U_2), (3,U_3), (4,U_4), (5,U_5)> and V = <(1,V_1), (2,V_2), (3,V_3), (4,V_4), (5,V_5)>. The probability value of score 5 of an object (that is, U_5 and V_5) dominates the probability values of scores 4, 3, 2 and 1 of another object. As a result, the probability value of U dominating V is $U_5 \times (V_4+V_3+V_2+V_1)+U_4 \times (V_3+V_2+V_1)+U_3 \times (V_2+V_1)+U_2 \times V_1$, and this probability is denoted by $Pr[U > V]$. Definition 7 is based on this concept. This is consistent with the skyline query on general tuples.

Definition 8. *(Better relation).* If U and V are two d-dimensional objects, for every dimension i ($1 \leq i \leq d$), if $Pr[U_i > V_i] \geq Pr[V_i > U_i]$, there exists a dimension j ($1 \leq j \leq d$) such that $Pr[U_j > V_j] > Pr[V_j > U_j]$. Then, U is better than V in the d-dimensional space.

If an object is better than any other object, this object is a skyline object. Better relation is transitive. If U is better than V and V is better than W, U is also better than W.

We explain how an object dominates another using the example in Table 6. In Table 6, R_1 = <(1,0), (2,0), (3,0.1), (4,0.4), (5,0.5), (1,0), (2,0), (3,0), (4,0.5), (5,0.5), (1,0), (2,0.2), (3,0.5), (4,0.2), (5,0.1)>. Similarly, R_2 = <(1,0), (2,0.5), (3,0.5), (4,0), (5,0), (1,0.25), (2,0.5), (3,0.25), (4,0), (5,0), (1,0), (2,0.25), (3,0.25), (4,0.25), (5,0.25)>. For food and service dimensions, $Pr[R_1 > R_2] > Pr[R_2 > R_1]$, but for décor dimension, $Pr[R_1 > R_2] < Pr[R_2 > R_1]$. As a result, R_1 cannot be dominated R_2 in décor dimension.

Definition 9. *(Skyline)* The skyline of objects set S, denoted as SKY(S), is a subset of objects that are better than another or cannot be dominated by another in at least one dimension.

2.3 Algorithms

Algorithm: *Dominate*(U, V)
Input: Objects U and V
Output: The dominating object
Steps:
1. Calculate i as $U_n*(V_{n-1}+V_{n-2}+\ldots+V_1)+U_{n-1}*(V_{n-2}+\ldots+V_1)+\ldots+U_2*V_1$;
2. Calculate j as $V_n*(U_{n-1}+U_{n-2}+\ldots+U_1)+V_{n-1}*(U_{n-2}+\ldots+U_1)+\ldots+V_2*U_1$;
3. IF $i > j$ THEN
4. RETURN 1
5. ELSE IF $i < j$ THEN
6. RETURN 0
7. ELSE
8. RETURN -1

Fig. 1. The algorithm for dominance relation comparison.

Fig. 1 presents the algorithm to do dominance relation comparison in one dimension. It is defined by Definition 7 in the previous subsection. For the multi-instance data, the skyline query makes 2 comparisons between two 1-dimensional objects. If we have m d-dimensional objects, there are $m \times (m-1) \times d/2$ comparisons for the dominance relation. For efficiency, we have to reduce the calculation cost. We first do presorting according to the values calculated by one of the three functions described in Table 7. Next, we do filtering. Then, if an object satisfies the conditions described by Theorem 1, further calculation on it can be skipped.

Table 7. A summary table for three functions

Function	Definition	Description
First	It is the probability of the highest score, or it is $p(x_n)$, where n is the highest score.	Intuitively, if an object has a larger probability value for having the highest score, it possibly dominates another one. For example, if a restaurant is assigned the highest score by most of the users, it is the most recommended one.
Second	It is the expected score and calculated by $\sum_{i=1}^{n} x_i \times p(x_i)$.	The expected value is a weighted average of all scores. If the expected value of an object is larger, the chance that this object dominates another object is higher.
Third	It is the net probability value between the two groups, and it is calculated by $\sum_{i=(n+1)/2}^{n} p(x_i) - \sum_{i=1}^{n/2} p(x_i)$.	We divide the probability values of each object into two groups: one is the sum of probabilities of the group with higher scores, and the other is the sum of probabilities of the group with lower scores. This extends from the first function.

We have objects with different dimensions for using one of the three functions and measure execution time for these objects by using Theorem 1 to reduce the number of calculations for the dominance relation. For the sorted objects by using the first function, the first object could be a skyline object because it has a larger the probability of the highest score to dominate another one. The expected value given by the second function is calculated by multiplying each of the probability values by the corresponding scores and summing all the values. We presort data by using one of three functions for reducing its computational cost.

THEOREM 1. Let A and B be two 1-dimensional objects and n be the highest score: $A = \langle (1, p_a(1)), \cdots, (i, p_a(i)), \ldots (n, p_a(n)) \rangle = \langle (1, A_1), \cdots, (i, A_i), \ldots (n, A_n) \rangle$ and $B = \langle (1, p_b(1)), \cdots, (i, p_b(i)),$ *...n,pbn*=1,*B*1,⋯,*i*,*Bi*, *...n,Bn*. (i) For n=2, if $A_2 > B_2$, then A is better than B. (ii) For n>2, if $A_n > (1/(2-B_n))$, then A is better than B.

Proof: (i) For each dimension of an object, the sum of these probability values is 1. For two 1-dimensional objects A=<(1, A_1), (2, A_2)> and B=<(1, B_1), (2, B_2)>, where 1 and 2 are scores, $A_1+A_2=1$ and $B_1+B_2=1$. $Pr[A > B] - Pr[B > A] > 0$ if A is better than B. Since $A_2=1-A_1$ and $B_2=1-B_1$. $Pr[A > B] = A_2 \times B_1 = A_2 \times (1-B_2) = A_2 - A_2 \times B_2$ and $Pr[B > A] = B_2 \times A_1 = B_2 \times (1-A_2) = B_2 - B_2 \times A_2$. So, $(A_2 - A_2 \times B_2) - (B_2 - B_2 \times A_2) > 0$, $A_2 - A_2 \times B_2 - B_2 + B_2 \times A_2 > 0$, and $A_2 - B_2 > 0$. So, if $A_2 > B_2$ then A is better than B. (ii) A

and B are two 1-dimensional objects with possible scores from 1 to n, so $\sum_{i=1}^{n} A_i = 1$ and $\sum_{i=1}^{n} B_i = 1$. A is better than B, if $Pr[A > B] > Pr[B > A]$ is satisfied. In the worst case, A is equal to <(1, (1−A_n)), (2,0),…, (n−1,0), (n, A_n) > and B is equal to <(1, 0), (2, 0),…, ((n-2), 0), ((n−1), (1−B_n)), (n, B_n)>. That is, the probability value of the highest score n of A is large enough to dominate B. If A is better than B, $Pr[A > B] - Pr[B > A] > 0$, $[A_n \times (1- B_n)] - [B_n \times (1- A_n) + (1- B_n) \times (1- A_n)] > 0$, $A_n - A_n \times B_n - (1-A_n) > 0$, $2 \times A_n - A_n \times B_n - 1 > 0$, $A_n (2- B_n) > 1$, and $A_n > (1/(2-B_n))$. If $A_n > (1/(2 - B_n))$ then A is better than B.

Fig. 2 presents the algorithm to do skyline query processing. We first sort the input data by using one of the three functions defined in Table 7.

Algorithm: *RetrieveSkyline*(S, D, n)
Input: A set of objects S, the number of dimensions D, the highest score n (while the lowest is 1)
Output: The skyline objects in S
Steps:

```
1.  SKY ← Ø, NOSKY ← Ø
2.  Sort S according to the values returned by a function defined in Table 7
3.  FOR EACH object A ∈ S DO
4.      IF A ∈ NOSKY THEN NEXT
5.      FOR EACH object B ∈ S and B≠A DO
6.          IF B ∈ NOSKY THEN NEXT
7.          WHILE D > 0 DO
8.              IF A_n > 1/(2-B_n) THEN
9.                  SKY ← SKY ∪ {A}
10.                 NOSKY ← NOSKY ∪ {B}
11.                 NEXT
12.             ELSE
13.                 x ← Dominate(A,B)
14.                 IF x = 1 THEN
15.                     SKY ← SKY-{B}
16.                     NOSKY ← NOSKY ∪ {B}
17.                     SKY ← SKY ∪ {A}
18.                 ELSE IF x = 0 THEN
19.                     SKY ← SKY-{A}
20.                     NOSKY ← NOSKY ∪ {A}
21.                     SKY ← SKY ∪ {B}
22.                 ELSE
23.                     SKY ← SKY ∪ {A,B}
24.                 D ← D-1
25.         END WHILE
26.     END FOR EACH
27. END FOR EACH
28. RETURN SKY
```

Fig. 2. The algorithm for skyline query processing and dominance relation comparison.

3. Results and Discussion

The goal of experiments is to show that our methods can efficiently process the skyline query on multi-instance data. We use the C programming language to implement our algorithms and conduct experiments on a general PC.

To verify the efficiency, we use synthetic datasets from random and Gaussian (or normal) distributions [27,28] and real datasets. For a synthetic dataset, the dimensionality is 1, 5, or 10. For a synthetic dataset from random distribution, the size (or the number of objects) is 5,000 (5K), 50,000 (50K), or 500,000 (500K). For a synthetic dataset from Gaussian distribution, the highest score of every dimension is 5 or 10. The real datasets include a review dataset and a dataset from Facebook.

We first generate 10 groups of objects by using random distribution. The size of each group is 100,000 (10K) with 1, 5, and 10 dimensions. Not all dimensions are independent. Therefore, we additionally generate 10 other groups by using Gaussian distribution [28,29]. To have four distribution patterns, we use the means 1.5, 2.5, 3.5 and 4.5 with the standard deviation 0.5. The generated values are rounded to an integer between 1 and 5 and they are scores assigned to an object. The dimension being 10 (D=10) means that users assign scores to an object in each of the 10 dimensions. So, an object has 50 probability values that are transformed from the data. The dominance values of two objects are calculated by converting the corresponding populations of the scores for each dimension into probabilities. With the naïve method, calculating three 1-dimensional objects requires six runs of dominance relation computation. When the number of dimensions increases, the chance of one object dominating another object can possibly be low, and the skyline query may return a large number of objects. The complexity of the processing increases when the number of dimensions increases. Nevertheless, as the number of dimensions increases, according to Theorem 1, we can decrease the number of comparisons of dominance relation for some dimensions.

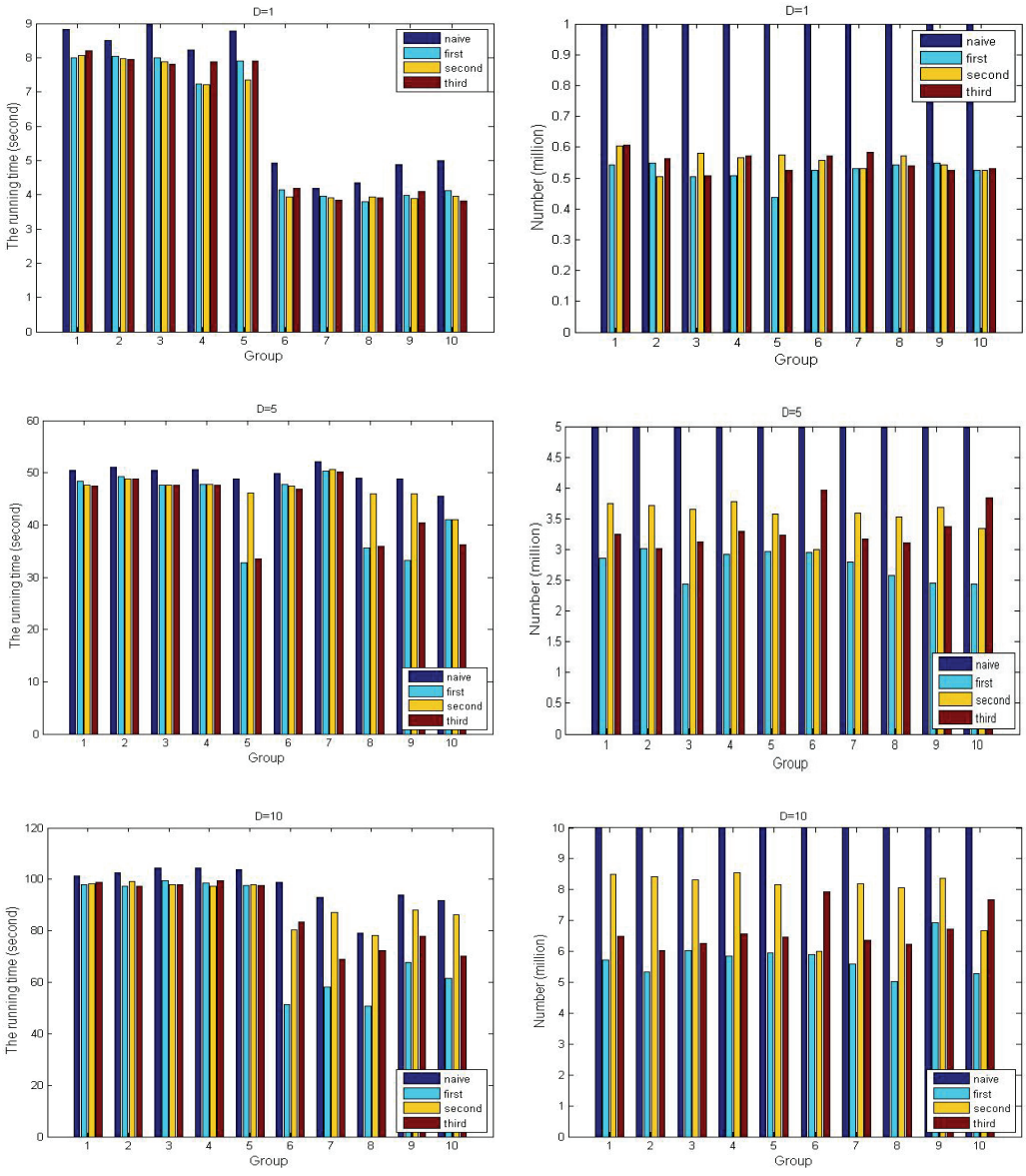
In experiments, we compare the running time of using the naïve method and that of using the presorting method with each of the three functions defined in Table 7. We also compare the number of dominance relation comparisons.

3.1 Synthetic Datasets

3.1.1 Datasets from random distribution

When D is 1, 5, or 10, using one of the three functions is faster than using the naïve method, and on average there is a reduction of 10%–20% in the running time, as shown in Fig. 3. For D is 5 or 10, the second function uses more time than do the other two functions. In Fig. 3, for the number of dominance relation comparisons, using the first function is similar to using the third function.

We discuss the datasets of different sizes with 5-dimensional random distribution. We compare three different sizes: 5,000 (5K), 50,000 (50K), and 500,000 (500K). In the running time, our methods are about 50 times faster on the dataset whose size is 5K than on the dataset whose size is 50K, as shown in Fig. 4. Our methods are about twice faster one the dataset whose size is 50K than on the dataset whose size is 500K. Due to a large amount of data, the two datasets whose sizes are 50K and 500K need additional I/O processing time. The running time of each of the two datasets is significantly longer than the dataset whose size is 5K. When it runs on the dataset whose size is 500K and is with the first function defined in Table 7, our methods use fewer calculations for dominance comparisons.



(a) (b)

Fig. 3. The running time (a) and the number of dominance relation comparisons (b) for datasets from random distribution.

Both the execution time and the number of comparisons can be reduced when our methods are used for situations where random distribution or Gaussian distribution is used to generate attribute values. Obviously, if an object is assigned poor scores in any of the dimensions, it would be eliminated according to Theorem 1. This means that no more calculation is required for dominance relation comparison and then the execution time is saved. For the situations where Gaussian distribution is used to generate the attribute values, the execution time decreases about 40% for D being 10, and the number of comparisons decreases 20%–30%.

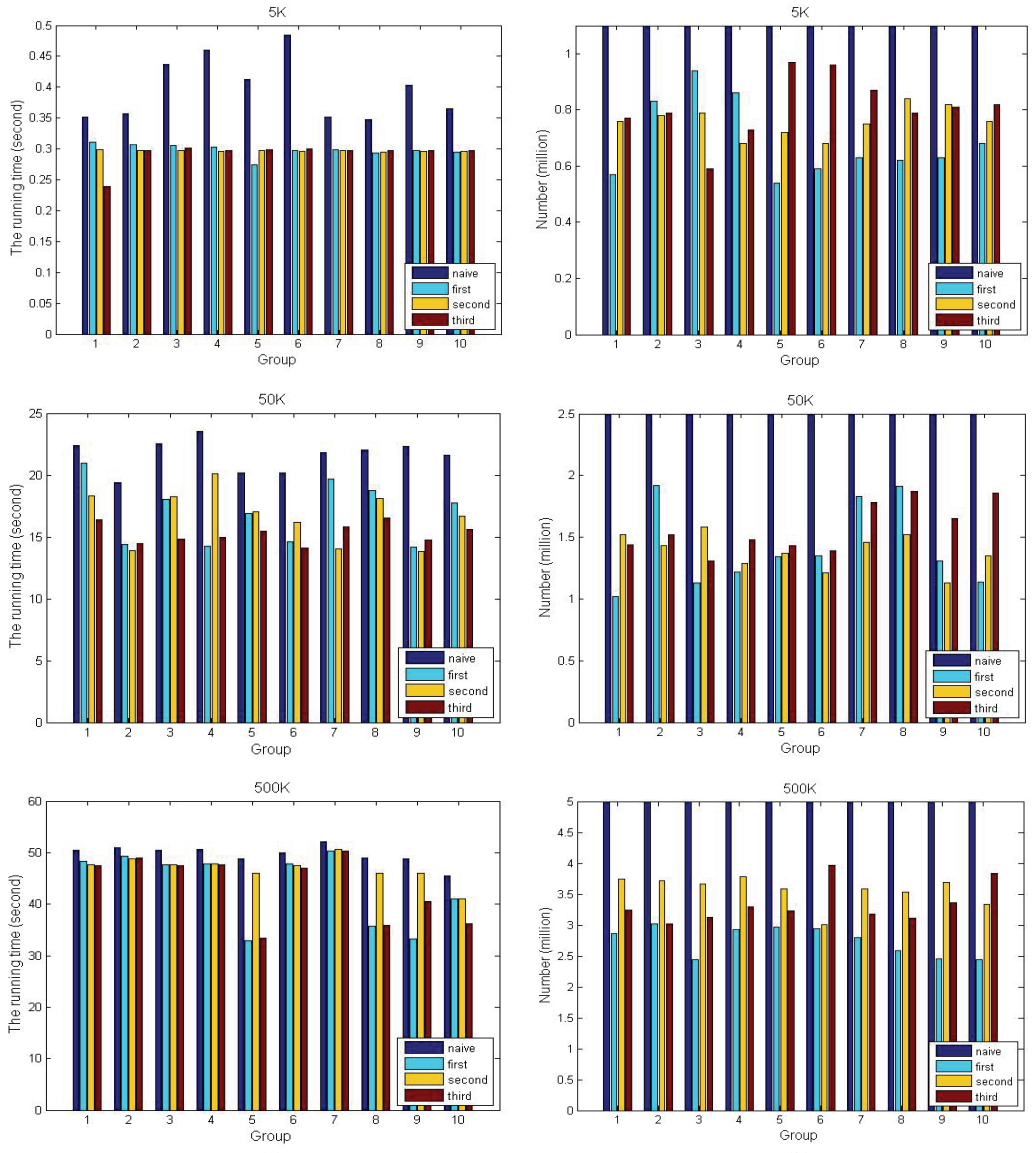
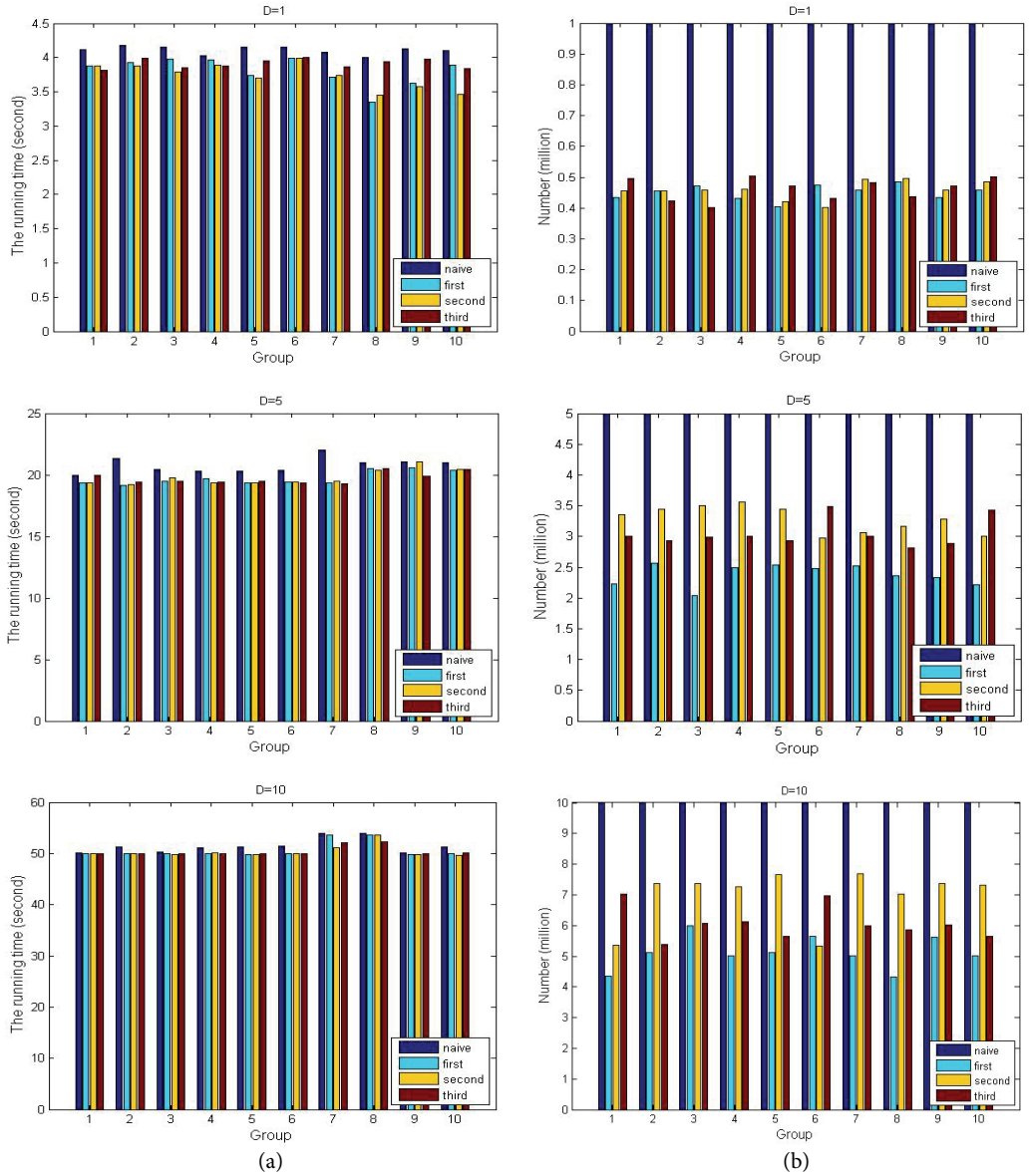


Fig. 4. The performance comparison between datasets from random distribution of different sizes. (a) The running time. (b) The number of dominance relation comparisons.

3.1.2 Datasets from Gaussian distribution

Gaussian distribution is a very common continuous probability distribution [26]. It is important in statistics and often used in sciences [26,27]. When D is 1, 5, or 10, using the three functions to presort the data is faster than using the naïve method, and on average there is a reduction of 5%–16% in the running time. When D is 5 or 10, using the first function uses fewer calculations than do the other two functions, as shown in Fig. 5. This indicates that the first function works well in high dimension. When the dimensionality is 1, our methods significantly reduce the number of comparisons. When D is 5 or 10, using the second function uses larger numbers of comparisons.



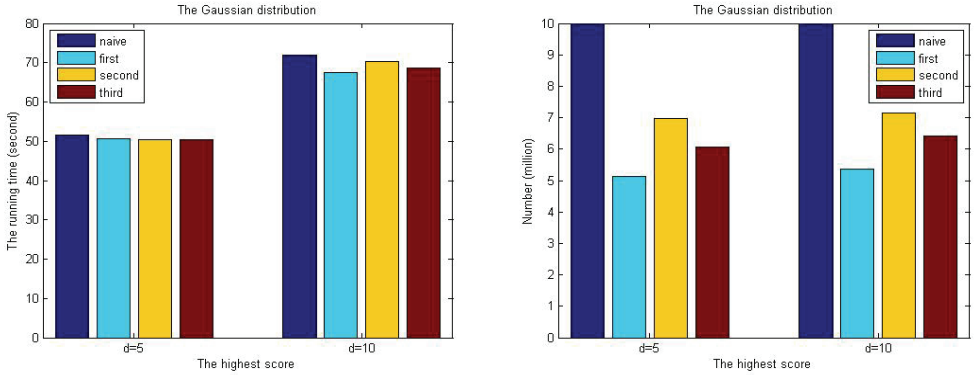
(a) (b)

Fig. 5. The running time (a) and the number of dominance relation comparisons (b) for datasets from Gaussian distribution

Both the execution time and the number of comparisons can be reduced when our methods are used for situations where random distribution or Gaussian distribution is used to generate attribute values. The reason is that the distributions of attribute values are dependent. For example, if an object is assigned poor scores in any of the dimensions, it would be eliminated according to Theorem 1. This means that no more calculation is required for dominance relation comparisons and then the execution time is less. For the situations where Gaussian distribution is used to generate the attribute values, the execution time decreases about 40% for D being 10, and the number of comparisons decreases 20%–30%.

We generate another 10-dimensional dataset with Gaussian distribution and the size of this dataset is

100,000 (100K). The scores of every dimension are from 1 to 10 in this dataset. It denotes that users assign scores from 1 to 10 to an object. It is different from the generated datasets described earlier. To have 9 distribution patterns, we use the means 1.5, 2.5, …, 9.5, and we use the standard deviation 0.5. When the score range increases from 5 to 10, the number of the probability values is doubled for every object. Compared to another dataset with the same dimension and distribution but different score ranges, this dataset increases the running time by about 40%, as shown in Fig. 6(a). When the score range increases, the growth rate of the running time is less than the growth rate of score range by using our methods. Fig. 6(b) shows that there is no significant difference between the datasets in the number of the dominance relation comparisons.



(a) (b)

Fig. 6. The performance comparison between 10-dimensional datasets from Gaussian distribution with different score ranges. (a) The running time. (b) The number of dominance relation comparisons.

Fig. 6 shows that the score range influences the performance of skyline query processing. As shown in Fig. 6, this dataset is as twice large in the number of probability values as the other, but its execution time increases by about 40%. In addition, a larger score range for every object increases the overall processing time, especially I/O processing time. When the number of probability values has a 100% increase, the overall processing time of any of our methods has a 40% increase, which means that, practically speaking, the runtime complexity of any of our methods is better than linear.

3.2 Real Datasets

3.2.1 The review dataset

We include a review dataset in our experiments. It is the restaurant and consumer (RC) dataset for recommender systems, and it is originally crawled from TripAdvisor (www.tripadvisor.com) for a period of one month and used in [29,30]. In this dataset, an object is a restaurant and a review is an instance; reviewers are asked to provide ratings on 3 dimensions in each review, namely overall, food and service rating, and each ranges from 0 to 2. We first perform pre-processing on the original dataset: (1) remove the reviews with any missing rating; (2) transform the rating data; (3) duplicate data and generate a much larger dataset for the performance test. After the pre-processing, the number of objects is 100,100 and the number of reviews is 1,789,480. Below are simple statistics for ratings: Overall is

1.20±0.04, food is 1.22±0.46, and service is 1.09±0.05. Further, the overall rating is dependent on the food and service ratings in the RC dataset.

We compare the RC dataset, the synthetic datasets from random distribution and Gaussian distribution. The results are in Fig. 7. Because the RC dataset contains only one group, we use the average of the two generated datasets. We found that the RC dataset is close to the dataset of Gaussian distribution in the running time and the number of comparisons. The RC dataset with the first function works well. Using one of the three presorting functions with Theorem 1 is better than using the naïve method for synthetic datasets and the RC dataset, as shown in Fig. 7.

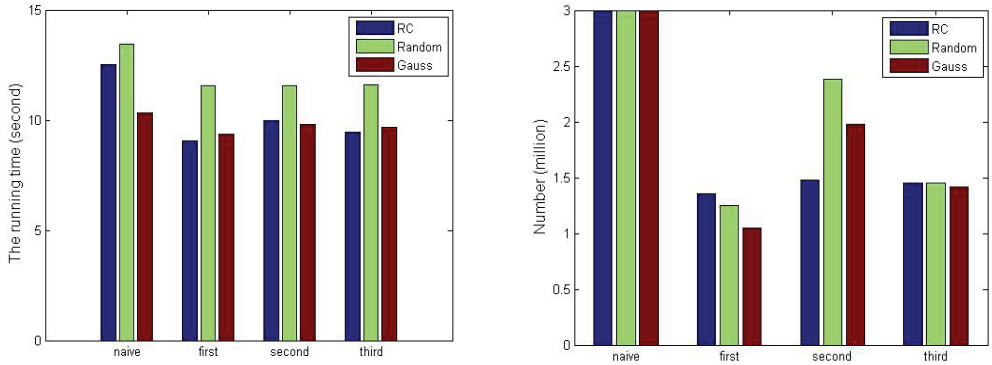


Fig. 7. The performance comparison between the RC dataset and two synthetic datasets. (a) The running time. (b) The number of dominance relation comparisons.

3.2.2 The dataset from Facebook

We use a real dataset collected from a social media platform, Facebook (FB), which is one of the most popular websites. The data is collected from the 20 selected fan pages through Facebook Graph API. The 20 selected fan pages are against the Cross-Strait Service Trade Agreement (CSSTA) on Facebook in Taiwan. The data contains the posts of the selected fan pages between March 18 and April 11, 2014. This period is known as the sunflower student movement. Every post contains two attributes, namely the number of shares and the number of comments. The task is to retrieve the posts that dominate others (or the skyline posts) in shares and comments.

The FB dataset includes 2,533 posts, 256,027 shares, 176,942 comments, and 84,781 users who react to these posts during this movement on Facebook. Reactions include sharing posts and commenting on posts. The share rating ranges from 1 to 5, and so does the comment rating. When a user shares a post within the first hour after the post is created, we assign 5, the highest score, to this user-post pair for the sharing rating. When a user shares a post between the first hour and the second hour, the share rating is 4. When a user shares a post over the fourth hour, the share rating is 1. We illustrate the method with an example given in Table 8. P_{id} and U_{id} are post identifier and user identifier, respectively. The post P_1 is denoted by <(1,0.2), (2,0.2), (3,0.1), (4,0.3), (5,0.2)>. Similarity, we use the same method to generate comment ratings. The definitions of share and comment ratings might seem arbitrary, but they are not unreasonable. Our goal is simply to have a dataset for the performance test.

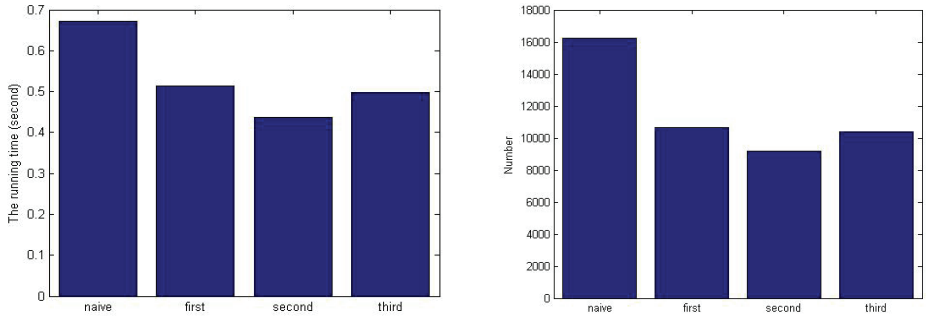
Table 8. A post is shared by users

P_{id}	PostCreatedTime	ShareCreatedTime	U_{id}	Sharing rating
P_1	2014-03-20 09:40	2014-03-20 10:03	U_1	5
P_1	2014-03-20 09:40	2014-03-20 10:31	U_2	5
P_1	2014-03-20 09:40	2014-03-20 10:45	U_3	4
P_1	2014-03-20 09:40	2014-03-20 11:10	U_4	4
P_1	2014-03-20 09:40	2014-03-20 11:35	U_5	4
P_1	2014-03-20 09:40	2014-03-20 12:20	U_6	3
P_1	2014-03-20 09:40	2014-03-20 12:50	U_7	2
P_1	2014-03-20 09:40	2014-03-20 13:30	U_8	2
P_1	2014-03-20 09:40	2014-03-20 18:40	U_9	1
P_1	2014-03-20 09:40	2014-03-21 10:00	U_{10}	1

Next, we perform data transformation. Every object contains two dimensions, namely share rating and comment rating. After transformation, simple statistics are given in Table 9.

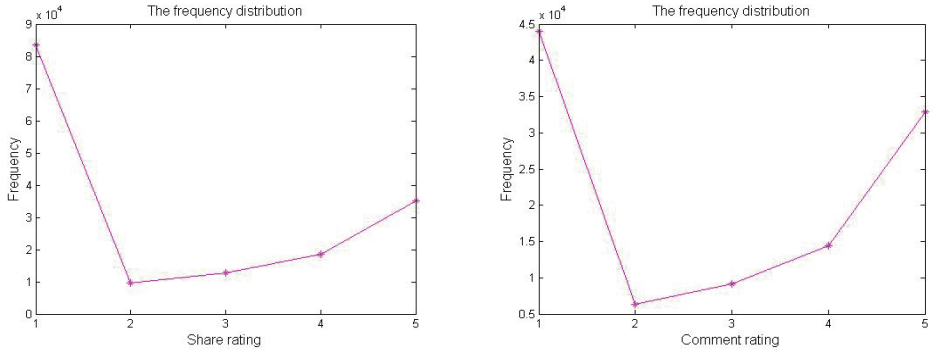
Table 9. Statistics for share and comment ratings in the FB dataset

	Number	Mean	Median	Mode	Standard deviation	Standard error
Share rating	256,027	2.45	1	1	1.69	0.004
Comment rating	176,942	2.86	3	1	1.75	0.005



(a) (b)

Fig. 8. The running time (a) and the number of dominance relation comparisons (b) for the FB dataset.



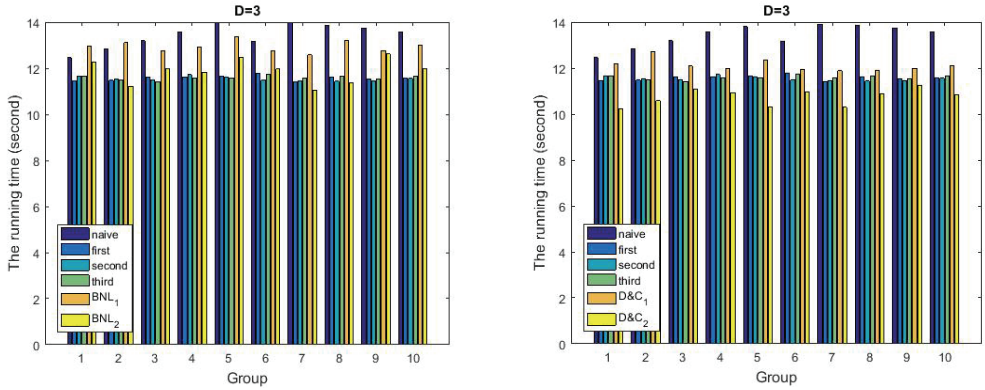
(a) (b)

Fig. 9. The frequency distributions of share rating (a) and comment rating (b) in the FB dataset.

Fig. 8(a) and (b) show the running time and the number of the dominance relation computation, respectively. This indicates that the second function works well in the FB dataset. This result is different from the results from the experiments on the above-mentioned datasets. The frequency distributions of share rating and comment rating are shown in Fig. 9(a) and (b), respectively. The frequency distributions of the two dimensions are different from the random and Gaussian distributions. From Fig. 9, the comment rating being 5 is the maximum number, and this presents that most users comment on posts within the first hour after the post is created. However, the share rating being 5 is less significant. Even when the distributions of values (ratings) of dimensions are skew, our methods are still better than the naïve method. In the number of dominance relation comparisons, our methods are about 30-40% lower than the naïve method (and hence our methods are faster).

3.3 Comparison with Other Skyline Query Processing Algorithms

A typical example of a skyline query is on the data objects in a multi-dimensional space. For example, an object is a product or service. Existing algorithms assume that an object has a value in a dimension. They handle single-instance data. In reality, however, an object can have many values in a dimension, since a product or service can have many review scores in a dimension, such as satisfaction. Therefore, we propose to directly handle multi-instance data. Our methods and existing algorithms are not totally comparable, so we first perform data transformation and then apply popular traditional algorithms to the transformed data. To process a skyline query, BNL algorithm compares every object with every other object in the dataset. In [4], D&C algorithm for skyline query processing is proposed. Our methods are compared with BNL and D&C algorithms. We use the synthetic datasets with 3-dimensional random distribution to examine the two algorithms and our methods. The data which a typical skyline query runs on is single-instance data. Therefore, we transform the multi-instance data to single-instance data and then we use two methods to process the skyline query. One method is to use all reviewers' scores as objects; that is, we treat instances as objects. With this transformation method, the result is the skyline over reviewer's scores, and the result is less meaningful. The other method to transform the data is to use average scores of objects, and this is common on many review websites today.



(a) (b)

Fig. 10. Our methods compared with BNL algorithm (a) and D&C algorithm (b).

We generate 10 groups of objects by using random distribution. The size of each group is 100,000 (10K) with 3 dimensions. Fig. 10(a) shows the running time of our methods and BNL algorithm applied to the data transformed by the two aforementioned methods (BNL_1 and BNL_2). Time spent on data transformation is excluded. BNL_1 is to use all reviewers' scores to transform the data and then use BNL. It processes the skyline query on all scores directly. The skyline tuples are instances (scores), not objects (products or services). BNL_1 cannot recommend objects. BNL_2uses average scores of objects to transform the data and then uses BNL. The skyline tuples are objects. We compute average scores of objects first and then process the skyline query for objects. Prior to the skyline query, BNL_2 transforms the data. There is no surprise that BNL_2 is faster than BNL_1. BNL_2 and our methods can recommend objects directly. Nevertheless, using averages will cause information lost and other problems, such as that averages are affected by extreme values. BNL_1 is faster than the naïve method in 8 groups. Our methods with three presorting functions proposed in this paper are faster than BNL_1 in all groups, and they are faster than BNL_2 in 7 groups. Fig. 10(b) shows the running time of our methods and D&C algorithm with the two aforementioned data transformation methods ($D\&C_1$ and $D\&C_2$). Similarity, $D\&C_1$ uses scores (that is, it treats instances as objects), and $D\&C_2$ uses average scores of objects. $D\&C_1$ is faster than the naïve method in all groups. $D\&C_1$ cannot recommend objects directly. Our methods are faster than $D\&C_1$ in all groups. Although $D\&C_2$ is faster than our methods, our methods can evaluate the contribution of every instance (and therefore can help find important instances, such as interesting reviews), and they can handle extreme values. The average will be affected by extreme values and it will have a bias. Nevertheless, the results show the advantage of D&C algorithm in running time. Please note that we do not include time spent on data transformation.

BNL and D&C algorithms use the Bitmap and the Index techniques. Our methods use presorting functions to speed up query processing, and they use Theorem 1 to reduce the number of calculations of the dominance relation. The first presorting function can finish the skyline computation earlier. Therefore, it performs well in most datasets.

4. Conclusions and Future Work

A skyline query is to retrieve dominating tuples from a set of tuples. A skyline tuple is one that is in the result of a skyline query or is part of the answer to a skyline query. Skyline query processing becomes an important research problem because it can be used to identify interesting tuples efficiently. Skyline queries are related to the maximum vector problem and multi-criteria decision making. Therefore, efficiently processing skyline queries is important and valuable. Today, many websites are offering users experience-sharing services and many users assign scores to a product or a service, called an object. For an object, one instance is a user's score and then these scores become many instances. Such data is called multi-instance data. Traditional skyline queries are defined upon single-instance data. However, our study focuses on efficiently processing skyline queries on multi-instance data. We define the dominance calculation and propose methods to reduce its computational cost. We propose three functions to presort the data. In experiments, we compare the running time of using the naïve method and each of the three functions. Overall, using one of the three presorting functions with Theorem 1 is better than using the naïve method for synthetic and real datasets. When the score range

or dimension increases, the growth rate of the running time is less than the growth rate of score range or dimension by using our methods. The first function has the best performance on the synthetic datasets from the two distributions. For the real datasets, the first function works well on the RC dataset, which is the restaurant and consumer dataset, and the second function works well on the FB dataset, which is regarding messages on a popular social media website.

As mentioned earlier, part of future work could be to parallelize our methods. Additionally, in the future, we plan to use our methods in a review website for e-commerce. This can lead to better user experience because our methods recommend products to users by using individual review scores rather than an overall review score. We propose methods to recommend interesting objects. Nowadays, many review websites only use the averages of review scores of objects. Using an average would possibly cause information lost and other problems. By using our methods, the review websites can give users not only the average of review scores of objects but also interesting individual reviews. In addition, we plan to design more functions to presort the data.

References

[1] X. Liu, D. N. Yang, M. Ye, and W. C. Lee, "U-skyline: a new skyline query for uncertain databases," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 4, pp. 945-960, 2013.

[2] S. Borzsony, D. Kossmann, and K. Stocker, "The skyline operator," in *Proceedings of the 17th International Conference on Data Engineering*, Heidelberg, Germany, 2001, pp. 421-430.

[3] J. Chomicki, P. Godfrey, J. Gryz, and D. Liang, "Skyline with presorting: theory and optimizations," in *Intelligent Information Processing and Web Mining*, Berlin, Germany: Springer, 2005, pp. 595-604.

[4] 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*, Roma, Italy, 2001, pp. 301-310.

[5] D. Papadias, Y. Tao, G. Fu, and B. Seeger, "An optimal and progressive algorithm for skyline queries," in *Proceedings of ACM SIGMOD International Conference on Management of Data*, San Diego, CA, 2003, pp. 467-478.

[6] S. I. Chiu and K. W. Hsu, "Skyline query processing for rating data," in *Proceedings of the Pacific Asia Conference on Information Systems*, Chiayi, Taiwan, 2016, pp. 305.

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

[8] J. Chomicki, P. Godfrey, J. Gryz, and D. Liang, "Skyline with presorting," in *Proceedings of the 19th International Conference on Data Engineering*, Bangalore, India, 2003, pp. 717-719.

[9] P. Godfrey, R. Shipley, and J. Gryz, "Maximal vector computation in large data sets," in *Proceedings of the 31st International Conference on Very Large Data Bases*, Trondheim, Norway, 2005, pp. 229-240.

[10] S. Zhang, N. Mamoulis, and D. W. Cheung, "Scalable skyline computation using object-based space partitioning," in *Proceedings of ACM SIGMOD International Conference on Management of Data*, 2009, pp. 483-494.

[11] 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*, Hong Kong, China, 2002, pp. 275-286.

[12] I. Bartolini, P. Ciaccia, and M. Patella, "Efficient sort-based skyline evaluation," *ACM Transactions on Database Systems*, vol. 33, no. 4, pp. 1-45, 2008.

[13] J. L. Bentley, K. L. Clarkson, and D. B. Levine, "Fast linear expected-time algorithms for computing maxima and convex hulls," *Algorithmica*, vol. 9, no. 2, pp. 168-183, 1993.

[14] H. T. Kung, F. Luccio, and F. P. Preparata, "On finding the maxima of a set of vectors," *Journal of the ACM*, vol. 22, no. 4, pp. 469-476, 1975.

[15] F. P. Preparata and M. I. Shamos, "Introduction," in *Computational Geometry*, New York, NY: Springer, 1985, pp. 1-35.

[16] M. J. Atallah and Y. Qi, "Computing all skyline probabilities for uncertain data," in *Proceedings of the 28th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, Providence, RI, 2009, pp. 279-287.

[17] J. Pei, B. Jiang, X. Lin, and Y. Yuan, "Probabilistic skylines on uncertain data," in *Proceedings of the 33rd International Conference on Very Large Data Bases*, Vienna, Austria, 2007, pp. 15-26.

[18] B. Jiang, J. Pei, X. Lin, and Y. Yuan, "Probabilistic skylines on uncertain data: model and bounding-pruning-refining methods," *Journal of Intelligent Information Systems*, vol. 38, no. 1, pp. 1-39, 2012.

[19] M. J. Atallah, Y. Qi, and H. Yuan, "Asymptotically efficient algorithms for skyline probabilities of uncertain data," *ACM Transactions on Database Systems (TODS)*, vol. 36, no. 2, article no. 12, 2011.

[20] Q. Zhang, P. Ye, X. Lin, and Y. Zhang, "Skyline probability over uncertain preferences," in *Proceedings of the 16th International Conference on Extending Database Technology*, Genoa, Italy, 2013, pp. 395-405.

[21] J. B. Rocha-Junior, A. Vlachou, C. Doulkeridis, and K. Norvag, "Efficient processing of top-k spatial preference queries," in *Proceedings of the 37th International Conference on Very Large Data Bases*, Seattle, WA, 2010, pp. 93-104.

[22] A. Cosgaya-Lozano, A. Rau-Chaplin, and N. Zeh, "Parallel computation of skyline queries," in *Proceedings of the 21st International Symposium on High Performance Computing Systems and Applications*, Saskatoon, Canada, 2007, p. 12.

[23] P. Wu, C. Zhang, Y. Feng, B. Y. Zhao, D. Agrawal, and A. El Abbadi, "Parallelizing skyline queries for scalable distribution," in *Proceedings of the 10th International Conference on Extending Database Technology*, Munich, Germany, 2006, pp. 112-130.

[24] B. Cui, H. Lu, Q. Xu, L. Chen, Y. Dai, and Y. Zhou, "Parallel distributed processing of constrained skyline queries by filtering," in *Proceedings of the IEEE 24th International Conference on Data Engineering*, Cancun, Mexico, 2008, pp. 546-555.

[25] H. Kohler, J. Yang, and X. Zhou, "Efficient parallel skyline processing using hyperplane projections," in *Proceedings of the ACM SIGMOD International Conference on Management of Data*, Athens, Greece, 2011, pp. 85-96.

[26] A. Vlachou, C. Doulkeridis, and Y. Kotidis, "Angle-based space partitioning for efficient parallel skyline computation," in *Proceedings of the ACM SIGMOD International Conference on Management of Data*, Vancouver, Canada, 2008, pp. 227-238.

[27] W. Bryc, *The Normal Distribution: Characterizations with Applications*. New York, NY: Springer, 1995.

[28] G. Casella and R. L. Berger, *Statistical Inference,* 2nd ed. Pacific Grove, CA: Duxbury Press, 2001.

[29] H. Wang, Y. Lu, and C. X. Zhai, "Latent aspect rating analysis without aspect keyword supervision," in *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, San Diego, CA, 2011, pp. 618-626.

[30] H. Wang, Y. Lu, and C. Zhai, "Latent aspect rating analysis on review text data: a rating regression approach," in *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Washington, DC, 2010, pp. 783-792.

Shu-I Chiu

She received B.S. and M.S. degrees from Tamkang University (Taiwan) and National Yang-Ming University (Taiwan), respectively. She is with the Department of Computer Science at National Chengchi University (Taiwan) as a Ph.D. candidate.

Kuo-Wei Hsu http://orcid.org/0000-0002-3496-5439

He received his B.S. degree in Electrical Engineering from National Chung Hsing University (Taiwan) in 1999, M.S. degree in Computer Science and Information Engineering from National Taiwan University (Taiwan), and Ph.D. degree in Computer Science and Engineering from University of Minnesota – Twin Cities (USA) in 2011. Since February 2011, he is with the Department of Computer Science at National Chengchi University (Taiwan) as an assistant professor.