Summer 6-27-2016

SKYLINE QUERY PROCESSING FOR RATING DATA

Shu-I Chiu
National Chengchi University, d9706@cs.nccu.edu.tw

Kuo-Wei Hsu
National Chengchi University, hsu@cs.nccu.edu.tw

SKYLINE QUERY PROCESSING FOR RATING DATA

Shu-I Chiu, Department of Computer Science, National Chengchi University, Taipei, Taiwan, d9706@cs.nccu.edu.tw

Kuo-Wei Hsu, Department of Computer Science, National Chengchi University, Taipei, Taiwan, hsu@cs.nccu.edu.tw

Abstract

Although electronic commerce has been developed and deployed for more than two decades, an essential problem remains: How to help customers efficiently find the products that they want? Skyline queries were proposed to address the problem, and we study the skyline queries defined on rating data. A product is represented by attributes each of which is an aspect that customers perceive it and can be individually rated, and ratings given by some consumers to it can be aggregated to assist in processing queries for product search. Therefore, our approach uses the rating data to get more suitable search results for customers. We first define the skyline query upon ratings and then propose an algorithm for its processing. We further propose a presorting method to speed up the whole computation. Results from experiments indicate that by using the proposed method the execution time can be notably shortened.

Keywords: Electronic Commerce, Product Search, Rating, Skyline Query Processing

1. INTRODUCTION

Soon after the rise of the Internet, electronic commerce started to evolve, expand, and effect many important changes in the world of business. It has been developed and deployed for more than two decades, but an essential problem remains: How to help customers efficiently find the products that they want? Aiding customers is to advance business, the first B in B2B (Business-to-Business). The customer can be the C in B2C (Business-to-Consumer) or the second B in B2B. From the database perspective, skyline query was proposed to address the problem of searching the database for products that best satisfy customers' requirements.

In recent years, skyline query has become a popular topic because it can be used to efficiently filter data records or tuples of several attributes. An attribute is a perspective on a tuple. A skyline is defined as a set of tuples that are not dominated by any other tuples. A tuple dominates another tuple if it is equally good or better in all attributes and better in at least one attribute. Let us consider the classic example: If we look for inexpensive hotels near a beach, a skyline query can choose hotels that are either less expensive or closer to the beach. If there is a more expensive hotel far from the beach, it will not be selected in a skyline. In this case, the skyline will only contain those hotels that are not worse than any other hotel in the price and the distance to the beach. The skyline tuples are considered to be important because they exhibit the three properties (Liu 2013):

- Skyline tuples are not dominated by any tuple outside the skyline set.
- Skyline tuples do not dominate each other, i.e., they hold on to their own importance in skyline against each other.
- All skyline tuples dominate all the non-skyline tuples, i.e., each non-skyline tuple is dominated by at least one skyline tuple.

If a tuple is a product and each of its attributes represent an aspect that customers perceive it, the skyline query performs filtering so that it keeps only those products that are not worse than any other product when all aspects are taken into consideration. It is widely used in multi-criteria decision making, and it helps people make intelligent decisions over complex data, where multiple criteria are considered (Soliman 2007, Wu 2006). Skyline tuples are "the best" under some monotonic preference function. As shown later in this paper, we propose monotonic aggregation functions for the particular type of data that we consider.

Today, there are many websites that allow or even encourage customers to express their experiences in products by rating them or assigning ratings to them. This is the customers' evaluation. When a customer wants to buy a product, he or she can consult these ratings to make a decision. Given a database of rating data, where a tuple is a rating given by a customer to a product in a certain category, we would like to search all the products in the category for the one having the best overall rating; or, if a tuple in the given database is a rating given by a customer to a vendor of a product, we would like to find the vendor having the best overall rating.

Let us consider the classic example again: A skyline query is to search for hotels that have high quality ratings and low prices. Generally, the hotels with high quality ratings tend to be expensive, and hotels with low prices are of low quality. There is one price but are many ratings attached to the hotel (or more precisely, some rooms in some season). Moreover, the ratings are given on several attributes, and hence the data is multi-dimensional data. What we want to find are not ratings on the skyline but the target or object (e.g. the hotel) to which the ratings are given; this is the reason why the traditional skyline query cannot be applied to rating data. Therefore, the problem that we address in this paper can be described as follows: How to efficiently find targets or objects that are not dominated by any other target or object in all aspects by using ratings or scores given to them?

The remainder of this paper is organized as follows: We briefly review the related work in Section 2, describe the proposed method in Section 3, report results from experiments in Section 4, and conclude this paper in Section 5.

2. RELATED WORK

The concept of the skyline query was proposed in 2001 (Borzsonyi 2001). The naïve way to compute a skyline is to apply a nested-loop algorithm and compare every tuple with every other tuple. In studies conducted by Chomicki et al. (2003, 2005), Sort-Filter-Skyline (SFS) algorithm was mainly advocated as a way to bring in the first positions those points that are likely to dominate many other points, thus leading to a reduction in the number of dominance tests. This algorithm first sorts the input data using a monotonic function. Our method also designs monotonic aggregation functions to reduce the number of comparisons of dominant relation between objects.

Our rating data is given by customers or websites' reviewers. Generally speaking, the arithmetic mean of rating data presents the consumers' evaluation. The mean is vulnerable to the effects of extreme value. It could be affected by the extreme values where biases would occur. For this reason, we use proportion method to transfer the probabilistic values to one record of several dimensions. Such a record contains a series of probabilistic values. Each probabilistic value corresponds to one rating data. We study how to process the skyline query for the probabilistic data.

Probabilistic (or uncertain) data are unavoidable in some important applications. Pei et al. (2006) proposed a probabilistic skyline model in which an uncertain object may take on a probability of being on the skyline (Pei 2007) called p-skyline. Given a probability threshold p ($0 \leq p \leq 1$), the p-skyline (Pei 2007, Jiang 2012) is the set of uncertain objects each of which takes a probability of at least p to be on the skyline. Atallah et al. (2009) proposed a general probabilistic skyline analysis that takes into account different user utilities without any restriction, but they do not use any probability threshold

(Mikhail 2009, 2011). Liu proposed a new uncertain skyline model called u-skyline, and it aims to return an uncertain skyline answer set from a different but complementary perspective to p-skyline (Liu 2013). Whereas p-skyline considers the global dominance among all data, u-skyline considers the global relationship among tuples. The dominant relation between two objects in our study (i.e. the relation in which one object dominates the other) is the sum of the probabilities that the higher ratings can dominate the lower ratings. In p-skyline, the authors define a probability for each tuple by aggregating over all the possible worlds within which the tuple is dominated. In our study, we calculate the dominant relation between two objects and then determine the proper one; if the determined object is not dominated by the other one, it is a skyline object and will be returned as answer to the query or search. This is consistent with the certain data of skyline query; however, what p-skyline defines is to check the probability of each object in skyline and a probability will be set up as the threshold. In this paper, we define the probability dominance between two objects for rating data, and then we deal with the skyline query on these objects in a multi-dimensional space.

3. METHOD

3.1 Skylines on certain data

There are two points, u and v, in a d-dimensional space $D = (D_1,\ldots,D_d)$. The dominant relation is presented on the preference attributes $D_1,\ldots,D_d$. We assume that bigger values are better. For every dimension D_i ($1 \leq i \leq d$), if $u.D_i \leq v.D_i$, and there exists a dimension D_j ($1 \leq j \leq d$) such that $u.D_j < v.D_j$, then v can dominate u.

3.2 Skylines for rating data

Let us consider that a rating is given by a consumer and it ranges from 1 to 5, and a product includes many ratings. Many ratings are corresponding to an object. For example, Table 1 presents a 3-dimensional rating data set containing ratings given by 10 consumers.

r_id	Restaurant Name	Reviews	Food Rating	Service Rating	Décor Rating
R1	Craftsteak Steak	u1	3	4	3
R1	Craftsteak Steak	u2	5	5	4
R1	Craftsteak Steak	u3	4	4	5
R1	Craftsteak Steak	u4	5	4	3
R1	Craftsteak Steak	u5	5	5	4
R1	Craftsteak Steak	u6	5	5	3
R1	Craftsteak Steak	u7	4	4	2
R1	Craftsteak Steak	u8	5	5	3
R1	Craftsteak Steak	u9	4	5	2

R1	Craftsteak Steak	u10	4	4	3

Table 1. The rating data with 3 dimensions

We transform the ratings in Table 1 to a tuple as in Table 2. If a random variable X is discrete, i.e., it may take a value from a specific set of n values x_i, $i = 1$ to n, then $P(X = x_i) = p(x_i)$, $p(x)$ is the probability mass function, where $p(x_i)$ denotes the probability of rating being x_i. An object is described by a probability mass function in the data space. We transform these ratings in Table 1 to a tuple for an object. So, this object 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)> in Table 2.

r_id	f_5	f_4	f_3	f_2	f_1	s_5	s_4	s_3	s_2	s_1	d_5	d_4	s_3	s_2	s_1
R1	(5,0.5)	(4,0.4)	(3,0.1)	(2,0)	(1,0)	(5,0.5)	(4,0.5)	(3,0)	(2,0)	(1,0)	(5,0.1)	(4,0.2)	(3,0.5)	(2,0.2)	(1,0)

Table 2: A 3-dimensional object

Definition 1. Let u and v be two 1-dimensional objects: $u = <(1, p_u(1)), (2, p_u(2)), \ldots, (n, p_u(n))>$ and $v = <(1, p_v(1)), (2, p_v(2)), \ldots, (n, p_v(n))>$; 1, 2, …, n are ratings. $P(x_i)$ denotes the probability of rating being x_i and the total of all $P(x_i)$ is 1. $Pr[u > v]$ denotes the probability that the object u dominates the object v, and $Pr[u > v]=p_u(n)\times[p_v(1)+\ldots+p_v(n-1)]+p_u(n-1)\times[p_v(1)+\ldots+p_v(n-2)]+\ldots+p_u(2)\times p_v(1)$.

In the skyline query, a point Pi dominates another point Pj, if and only if in any dimension the value of Pi is not larger than that of Pj. We apply the same concept to the rating data containing probabilistic values, and accordingly we define how the probabilistic values of the higher rating can dominate the probabilistic values of the lower rating. Let u and v be two 1-dimensional objects with possible ratings 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 rating 5 of a point (i.e. P(5) for u_5 and v_5) can dominate the probability values of rating 4, 3, 2 and 1 of another point. So, the probability value of u dominating v, denoted by $Pr[u > v]$, is equal to $u_5\times(v_4+v_3+v_2+v_1)+u_4\times(v_3+v_2+v_1)+u_3\times(v_2+v_1)+u_2\times v_1$. Definition 1 is based on this concept.

Example 1. For $n = 5$ (the highest rating), $u = \langle(1,0.1), (2,0.1), (3,0.3), (4,0.3), (5,0.2)\rangle$ and $v = \langle(1,0), (2,0.2), (3,0.4), (4,0.2), (5,0.2)\rangle$ are two 1-dimensional objects. The probability that u dominates v is $Pr[u > v] = 0.2 \times (0.2 + 0.4 + 0.2) + 0.3 \times (0.4 + 0.2) + 0.3 \times 0.2 + 0.1 \times 0 = 0.4$. The probability that v dominates u is $Pr[v > u] = 0.2 \times 0.8 + 0.2 \times 0.5 + 0.4 \times 0.2 + 0.2 \times 0.1 = 0.36$, This shows that u is *better* than v, so we will answer the query or search by returning the object u, i.e. product u. We define the *better* relation between u and v as follows: If u and v are two d-dimensional objects, for every dimension D_i ($1 \le i \le d$), if $Pr[u.D_i > v.D_i] \ge Pr[v.D_i > u.D_i]$, there exists a dimension D_j ($1 \le j \le d$) such that $Pr[u.D_j > v.D_j] > Pr[v.D_j > u.D_j]$. u is better than v in the d-dimensional space. We denote $u.D_i > v.D_i$ for all i. If u is *better* than v and v is *better* than w, than u is *better* than w. So the *better* relation between two objects satisfies the conditions of being transitive. If an object is better than any other object, then this object is a skyline object. Our data flow is denoted in Figure 1.

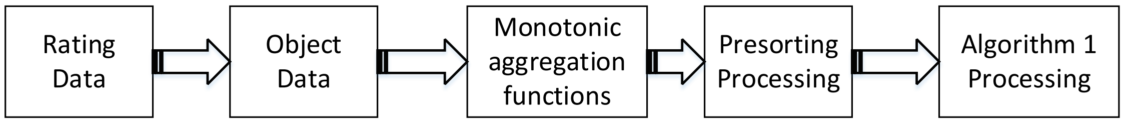


Figure 1. The data flow

Our approach makes 2 comparisons between two *d*-dimensional objects. If we have *n* *d*-dimensional objects, there will be $n \times (n - 1) \times d$ comparisons for the dominant relation. For efficiency, we have to reduce the calculation cost. We first sort the objects according to the defined aggregation values, and then we filter the presorted objects through the proposed algorithm. If an object satisfies the conditions described below, further calculation on it can be skipped. The results from our experiments show the efficiency of our approach. Before processing the comparison, two aggregation values will be generated for each object, and two other tables will be produced for further reference based on each of the two aggregation values.

- The first aggregation value is based on the probability of the highest rating. Intuitively, if an object has a larger probability value for having the highest rating, it possibly dominates another one. For example, if a restaurant is given the highest rating by most of the people, and it is the most recommended one by the system.
- The second aggregation value is sorted by the expected value (i.e. the expected rating) for every object. The expected value is a weighted average of all appearing values or ratings. Generally speaking, the expected value of an object is bigger, the chance that this object dominates another object is better. There may appear counter-examples of some unusual situations. Consider the following example. An object *A* is <(5, 0), (4, 0), (3, 0), (2, 1), (1, 0)>, and its expected value is 2. An object *B* is <(5, 0.4), (4,0), (3,0), (2,0), (1, 0.6)>, and its expected value is 2.6. Comparing the expected values, we can see that *B* is better than *A*. However, $\Pr[A > B]=0.6$ and $\Pr[B > A]=0.4$. Since $\Pr[A > B] > \Pr[B > A]$, *A* is *better* than *B*. In this special case, the object which has the larger expected value may not be able to dominate another object which has a smaller expected value.

We propose the second aggregation value, which comes from the probability theory. The expected value is calculated by first multiplying each of the possible outcomes by the likelihood that each outcome will occur and then summing all the values. In our study, the expected value is calculated by multiplying each of the probability values by the corresponding ratings and summing all the values. We use the following lemmas to reduce the amount of calculation of the dominant relation.

LEMMA 1: For n>2, let $A = <(1, a_1),\ldots, (n, a_n)>$ and $B = <(1, b_1),\ldots,(n, b_n)>$ are two objects. If $a_n > b_n$ and $a_n > (1/(2-b_n))$, then *A* is *better* than *B*.

Proof: Let A = <(1, a_1), (2, a_2),…, (n, a_n)> and B = <(1, b_1), (2, b_2),…, (n, b_n)> be two objects. Both are two 1-dimensional objects with possible ratings 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 and only if Pr[$A > B$] > Pr[$B > A$] is satisfied. In the worst case, A=<(1,(1-x)), (2,0),…,(n-1,0), (n, x) > and B=<(1, 0), (2, 0),…, ((n-2), 0), ((n-1), (1-y)), (n, y)>, where x=a_n and y=b_n are probability values. That is, the probability value of the highest rating n of A is large enough to dominate B. If A is *better* than B, Pr[$A > B$]-Pr[$B > A$] > 0, [x×(1-y)]-[y×(1-x)+(1-y)×(1-x)] > 0, x-xy-(1-x) >0, 2x-xy-1 > 0, x(2-y) > 1, and x >[1/(2-y)]. So, if a_n > (1/(2-b_n)), then A is *better* than B.

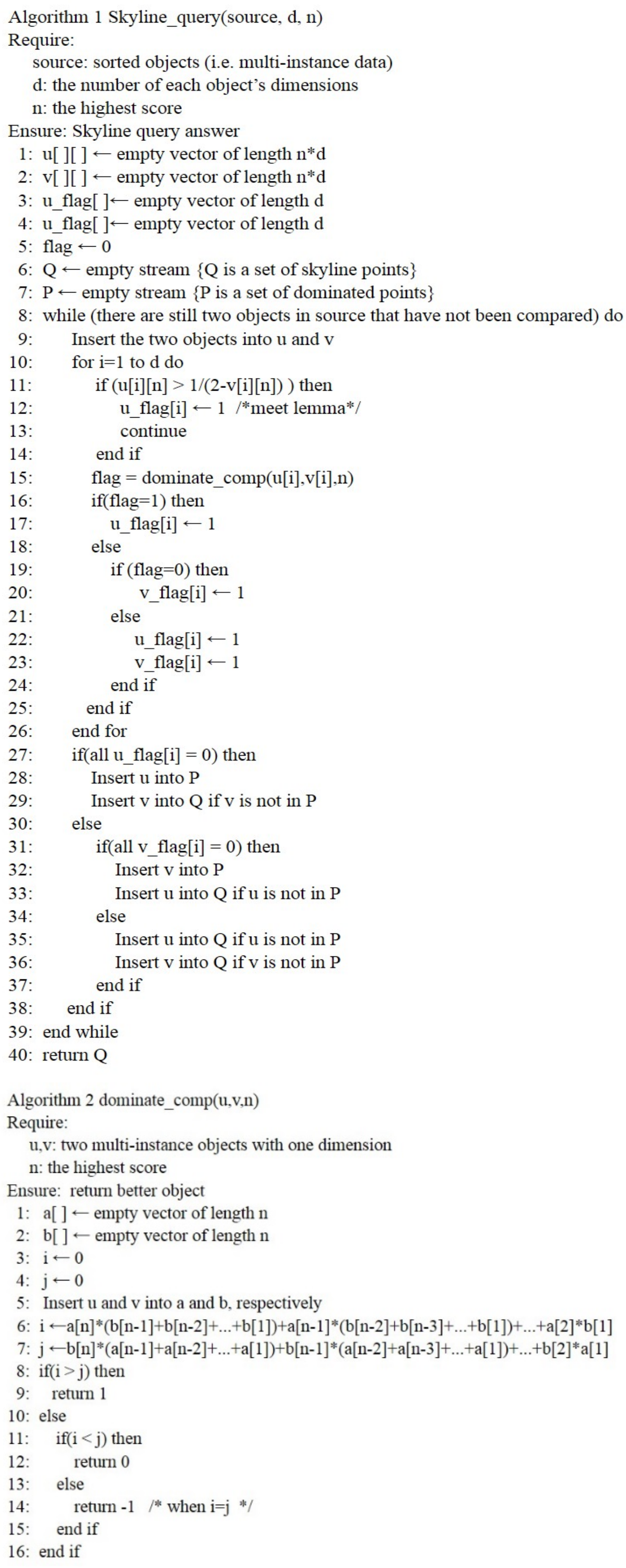
LEMMA 2. For n = 2, let A = <(1, a_1), (2, a_2)>, B = <(1, b_1), (2, b_2)> are two objects. If $a_2 > b_2$, then A is *better* than B.
Proof: For each dimension of an object, the sum of these probability values is 1. For two 1-dimension objects A=<(1, a_1), (2, a_2)> and B=<(1, b_1), (2, b_2)>, we know that 1 and 2 are rating dimensions and that $a_1 + a_2$ =1 and $b_1 + b_2$ =1. If A is *better* than B, then Pr[$A > B$]-Pr[$B > A$] > 0. a_2 =1-a_1 and b_2=1-b_1. From Definition 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, $a_2 - a_2 b_2 - b_2 + b_2 a_2 > 0$, and $a_2 - b_2 > 0$. If $a_2 > b_2$, then A is *better* than B.

Figure 2 presents two algorithms. The first algorithm is to do skyline query processing, and the second algorithm is to do dominant relation comparison.

4. EXPERIMENTS

The goal of experiments is to show that using the aggregation values to presort objects can improve the efficiency of the skyline query processing discussed in this paper and that using the above lemmas can reduce the amount of calculation for dominant relation comparison. We use the C programming language to implement our algorithms and conduct experiments on a general PC.

We generate 10 groups of objects by using the random distribution. Each group has 100,000 objects with 1, 3, 5, and 10 dimensions. Dimensions are not totally independent. So, we additionally generate 10 other groups by using the Gaussian distribution. 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 the ratings given to an object or a product by customers. Customers give ratings to a product in each of the dimensions. The dimension being 10 (D=10) means that customers give ratings to a product in each of the 10 dimensions. So, a product has 50 probability values that are transformed from rating data. The dominant values of two objects are then calculated by converting the corresponding populations of the ratings for each dimension into probabilities. With the naïve method, calculating 3 objects requires 6 runs of dominant relation computation. When the number of dimensions increases, the chance of one point dominating another point can possibly be very low, and the skyline query may return a large number of points. The complexity of the processing increases when the number of dimensions increases. Nevertheless, as the number of dimensions increase, the above lemmas can decrease the number of comparisons of dominant relation for some dimensions.



```
Algorithm 1 Skyline_query(source, d, n)
Require:
    source: sorted objects (i.e. multi-instance data)
    d: the number of each object's dimensions
    n: the highest score
Ensure: Skyline query answer
 1: u[ ][ ] ← empty vector of length n*d
 2: v[ ][ ] ← empty vector of length n*d
 3: u_flag[ ]← empty vector of length d
 4: u_flag[ ]← empty vector of length d
 5: flag ← 0
 6: Q ← empty stream {Q is a set of skyline points}
 7: P ← empty stream {P is a set of dominated points}
 8: while (there are still two objects in source that have not been compared) do
 9:     Insert the two objects into u and v
10:     for i=1 to d do
11:         if (u[i][n] > 1/(2-v[i][n]) ) then
12:             u_flag[i] ← 1  /*meet lemma*/
13:             continue
14:         end if
15:         flag = dominate_comp(u[i],v[i],n)
16:         if(flag=1) then
17:             u_flag[i] ← 1
18:         else
19:             if (flag=0) then
20:                 v_flag[i] ← 1
21:             else
22:                 u_flag[i] ← 1
23:                 v_flag[i] ← 1
24:             end if
25:         end if
26:     end for
27:     if(all u_flag[i] = 0) then
28:         Insert u into P
29:         Insert v into Q if v is not in P
30:     else
31:         if(all v_flag[i] = 0) then
32:             Insert v into P
33:             Insert u into Q if u is not in P
34:         else
35:             Insert u into Q if u is not in P
36:             Insert v into Q if v is not in P
37:         end if
38:     end if
39: end while
40: return Q

Algorithm 2 dominate_comp(u,v,n)
Require:
    u,v: two multi-instance objects with one dimension
    n: the highest score
Ensure: return better object
 1: a[ ] ← empty vector of length n
 2: b[ ] ← empty vector of length n
 3: i ← 0
 4: j ← 0
 5: Insert u and v into a and b, respectively
 6: i ←a[n]*(b[n-1]+b[n-2]+...+b[1])+a[n-1]*(b[n-2]+b[n-3]+...+b[1])+...+a[2]*b[1]
 7: j ←b[n]*(a[n-1]+a[n-2]+...+a[1])+b[n-1]*(a[n-2]+a[n-3]+...+a[1])+...+b[2]*a[1]
 8: if(i > j) then
 9:     return 1
10: else
11:     if(i < j) then
12:         return 0
13:     else
14:         return -1  /* when i=j */
15:     end if
16: end if
```

Figure 2. Algorithms for skyline query processing and dominant relation comparison

First, we use these two aggregation values to assign a better ranking order to each object. Lemma 1 is used to reduce the amount of calculation for dominant relation comparison. In experiments, we compare the execution time of using the naïve method and that of using the presorting method with each of the two aggregation values. For rating generated by using the random distribution, Figures 3 and 4 show the result in the execution time and that in the number of comparisons, respectively. For *D*=1 and *D*=3, using the presorting method with one of the two aggregation values is faster than using the naïve method, and on average there is a reduction of 15% in execution time. For *D*=5, using the presorting method with one of the two aggregation values can save us an average of 25% in execution time when compared to using the naïve method, and this indicates that our algorithms can enhance efficiency under more dimensions.

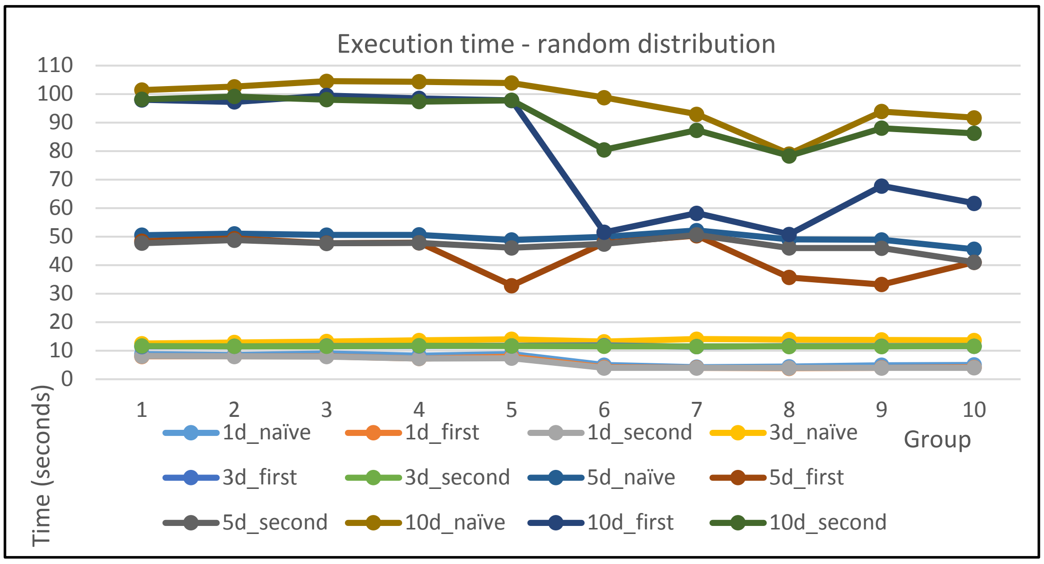


Figure 3. *Running time on different dimensions and aggregation values*

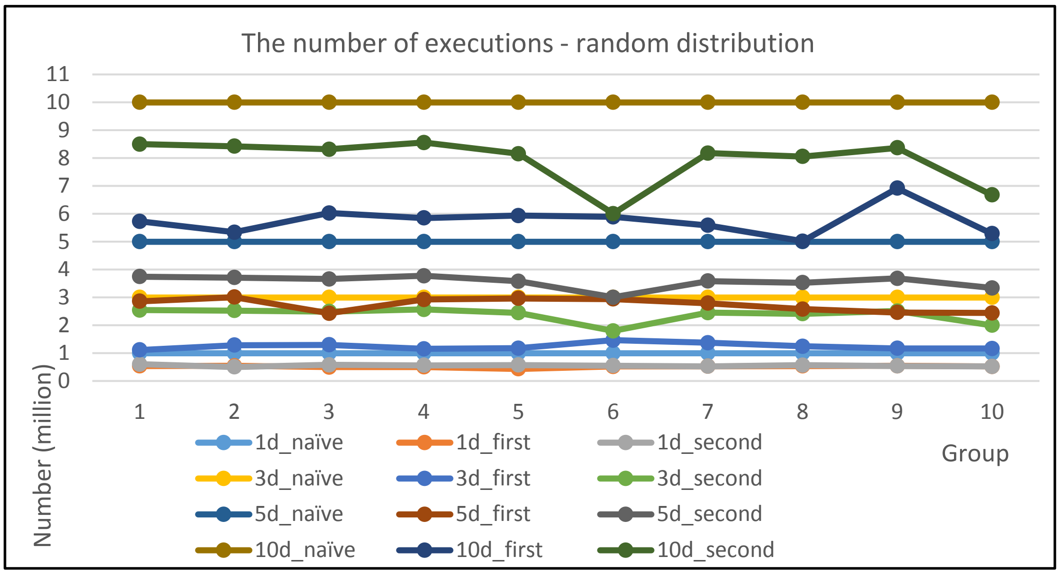


Figure 4. *The number of comparisons on different dimensions and aggregation values*

Next, we consider the data generated by using the Gaussian distribution. After using one of the two aggregation values for presorting, we use Lemma 1 for preliminary filtering. Figure 5 and 6 show the execution time and the numbers of comparisons performed by using the naïve method and the presorting method with two aggregation values for different numbers of dimensions. When *D*=1, our method can significantly reduce the number of comparisons. For *D*=3 and *D*=5, using the presorting method with the second aggregation value requires a smaller average number of comparisons, and this indicates that the presorting method based on using the expected values needs a small number of comparisons when there are more dimensions. For *D*=3, 5, and 10, the average number of comparisons required by using the presorting method with the first aggregation value is somewhat similar to that required by using the presorting method with the second aggregation value. The results from our experiments show that the execution time for the situations where the attribute values are drawn from the Gaussian distribution is reduced by 20-40%, when it is compared with that for the situations where the attributes are drawn from the random distribution. In terms of the number of comparisons, when the attribute values are drawn from the Gaussian distribution, it is reduced by 10-30%, compared with that when the attribute values are drawn from the random distribution. Because the Gaussian distribution is closer to what generates the data in many real-world situations, the results suggest that our algorithms have a great potential to be applied to many real-world applications.

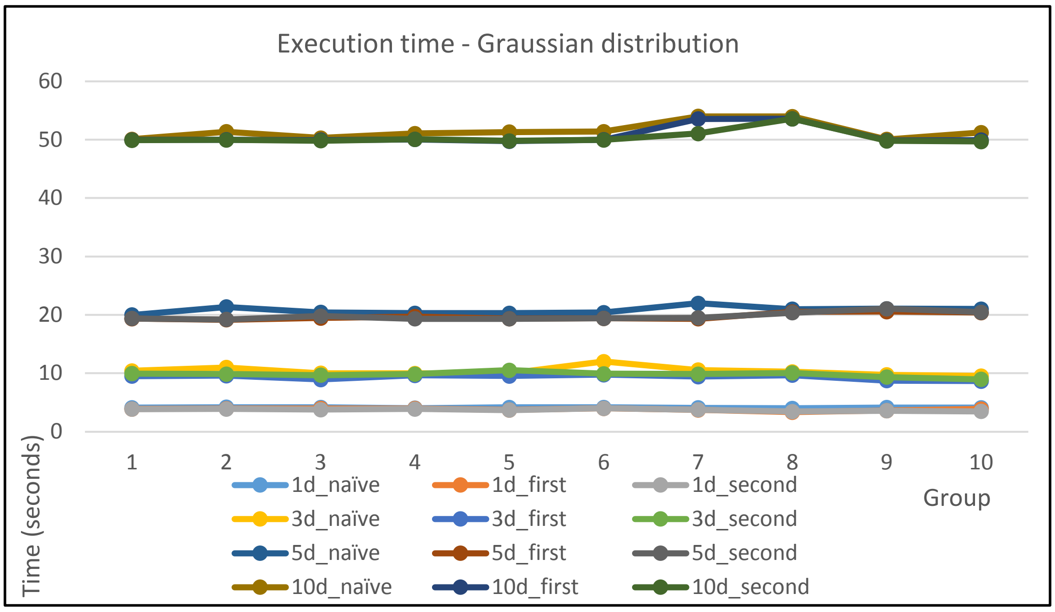


Figure 5. Running time on different dimensions and aggregation values

Both the execution time and the number of comparisons can be reduced when our method is used for situations where the random distribution or the Gaussian distribution is used to generate attribute values. The reason is that the distributions of attribute values are dependent. For example, if an object has been given poor ratings in any of the dimensions, it would be eliminated according to the Lemmas. This means that no more calculation is required for dominant relation comparison and then the execution time is less. For the situations where the Gaussian distribution is used to generate the attribute values, the execution time decreases about 40% for *D*=10, and the number of comparisons decreases 20-30%.

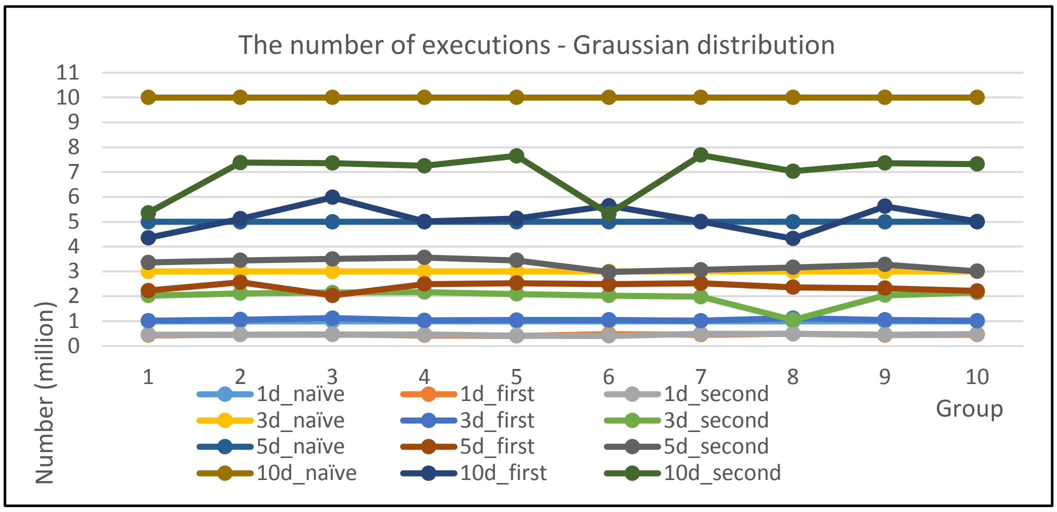


Figure 6. The number of comparisons on different dimensions and aggregation values

5. CONCLUSIONS

With the rapid development of the Internet, electronic commerce started to evolve, expand, and effect many important changes in the world of business. How to help customers efficiently find the products that they want? The problem remains, however. Many websites are offering customers experience-sharing services. When customers share their experiences regarding the quality of certain products and rate them based on the different attributes, the generated data set can be viewed as a set of objects. An object is a product (e.g. a restaurant), and there are many ratings given by customers to it. Our approach uses these ratings to provide more suitable products to customers who made the search. We propose a method to process skyline query on these ratings. We use a presorting method with two aggregation values to speed up the query processing. The results from experiments indicate that using the presorting method with any of the proposed aggregation values can reduce the execution time. The presorting method based on the expected value (the second proposed aggregation value) can generate more favorable outcomes for objects with more dimensions, and this results in a fewer number of calculation required for dominant relation comparisons. For future studies, we plan to propose new aggregation values for presorted objects. We hope that the first tuple of presorted objects by new aggregation values is one of skyline tuples. On the other hand, the rating data belongs to dynamic data, and thus these objects including probability values are variable. We plan to provide a novel approach to solve this condition in the future. Moreover, using real world data to evaluate our algorithms will be part of our future work.

ACKNOWLEDGMENT

The authors would like to thank the Ministry of Science and Technology, R.O.C. for the support under grant number 102-2420-H-004-045-MY3. The authors would also like to thank anonymous reviewers for their valuable time and comments.

References

Borzsonyi, S, Kossmann, D. and Stocker, K. (2001). The Skyline Operator. International Conference on Data Engineering, 421-430.

Byers, J. W., Mitzenmacher, M. and Zervas, G. (2012). The Groupon Effect on Yelp Ratings: A Root Cause Analysis. Proceedings 13th ACM Conference Electronic Commerce, 248-265.

Chomicki, J., Godfrey, P., Gryz, J. and Liang, D. (2003). Skyline with Presorting. International Conference on Data Engineering, 717-719.

Chomicki, J., Godfrey, P., Gryz, J. and Liang, D. (2005). Skyline with Presorting: Theory and Optimizations. Intelligent Information Systems, 595-604.

Jiang, B., Pei, J., Lin, X, and Yuan, Y. (2012). Probabilistic skylines on uncertain data: model and bounding-pruning-refining methods. Journal of Intelligent Information Systems, 38(1), 1-39.

Liu, X., Yang, D. N., Ye, M. and Lee, W. (2013). U-Skyline: A New Skyline Query for Uncertain Databases. IEEE Transactions on Knowledge and Data Engineering, 25(4), 945-960.

Mikhail J. A., and Yinian Q. (2009). Computing All Skyline Probabilities for Uncertain Data. Proceedings of the twenty-eighth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems, 279-287.

Mikhail J. A., Yinian Q. and Hao, Y. (2011). Asymptotically Efficient Algorithms for Skyline Probabilities of Uncertain Data. ACM Transactions on Database Systems, 36(2), 12:1-28.

Pei, J., Jiang, B., Lin, X. and Yuan, Y. (2007). Probabilistic Skylines on Uncertain Data. Proceedings of the 33rd International Conference on Very Large Data Bases, 15-26.

Soliman, M.A., Ilyas, I.F. and Chang, K. C.-C. (2007). Top-k query processing in uncertain databases. In International Conference on Data Engineering, 896-905.

Wu, P., Zhang, C., Feng, Y., Zhao, B. Y., Agrawal, D. and Abbadi, A.E. (2006). Parallelizing Skyline Queries for Scalable Distribution. Proceedings of the 10th international conference on Advances in Database Technology, 112-130.