General Dominant Relationship Analysis based on Partial Order Models

Zhenglu Yang
University of Tokyo
4-6-1 Komaba, Meguro-ku
Tokyo, Japan
yangzl@tkl.iis.u-tokyo.ac.jp

Botao Wang
University of Tokyo
4-6-1 Komaba, Meguro-ku
Tokyo, Japan
botaow@tkl.iis.u-tokyo.ac.jp

Masaru Kitsuregawa
University of Tokyo
4-6-1 Komaba, Meguro-ku
Tokyo, Japan
kitsure@tkl.iis.u-tokyo.ac.jp

ABSTRACT

Due to the importance of skyline query in many applications, it has been attracted much attention recently. Given an N-dimensional dataset D, a point p is said to dominate another point q if p is better than q in at least one dimension and equal to or better than q in the remaining dimensions. Recently, Li et al. [9] proposed to analyze more general dominant relationship in a business model that, users are more interested in the details of the dominant relationship in a dataset, i.e., a point p dominates how many other points. In this paper, we further generalize this problem that, users are more interested in whom these dominated points are. We show that the framework proposed in [9] can not efficiently solve this problem. We find the interrelated connection between the partial order and the dominant relationship. Based on this discovery, we propose efficient algorithms to answer the general dominant relationship queries by querying the partial order representation of spatial datasets. Extensive experiments illustrate the effectiveness and efficiency of our methods.

Categories and Subject Descriptors

H.2 [**Database Management**]: Query processing

General Terms

Algorithms, Performance

Keywords

dominant relationship analysis, partial order

1. INTRODUCTION

Recently, the skyline query has attracted considerable attention because it is the basis of many applications, e.g., multi-criteria decision making [2], user-preference queries [6] and microeconomic analysis [9]. Skyline mining [2] aims to find those points, which are not dominated by others, in a d-dimensional spatial dataset. This problem can be seen as a special class of pareto preference queries [6]. Fig. 1 shows one classic example of skyline query that customers are always interested in those "best" hotels that are better than others at least at one of the two criteria, the distance and the price, with smaller values. The skyline of the example dataset in Fig. 1 consists of a and c.

Efficient skyline querying methodologies have been studied extensively [2] [5] [7] [11] [17] [12] [10] [4] [1]. However, all the above papers concerned only the pure dominant relationship among a dataset, i.e., a point p is whether dominated by others or not, and got those non-dominated ones as results. Recently, Li et al. [9] proposed to analyze more general dominant relationship in a business model that, users are more interested in the detail of the dominant relationship in a dataset, i.e., a point p dominates how many other points and is dominated by how many others. In Fig. 1, although the hotel b is not a skyline, its manager also wants to know how many hotels b dominates (i.e. 2) and how many hotels dominates b (i.e. 2), from where the manager can know the business position of b in the local area.

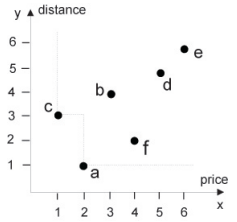

Figure 1: Example of the skyline query

However, in the real world, users are always interested in not only "how many" objects are dominating/dominated by a specific object, but also "whom" they are, which was however, not mentioned in [9]. This problem can be seen as a general dominant relationship analysis to the ones proposed in [9]. It is naively thought, can be easily solved by associating each object with its corresponding cuboid in DADA. So when users query the dominant relationship, these objects will be extracted simultaneously. Nevertheless, due to a huge number of duplicate existence in DADA, the storage overhead and the query time will be unacceptable for users. In this paper, we aim at proposing efficient and effective methods to answer the "whom" problem, based on our discovery that there is an interrelated connection between the general dominant relationship and the partial order.

To illustrate the core idea of this paper, here we show a simple example. Fig. 2 represents the partial order (encoded as DAG format) of the example dataset in Fig. 1 in 2-dimensional space. We can know the point b dominates the points d and e and is dominated by the points a and c, by counting the $out\text{-}link$ and $in\text{-}link$ of d, respectively. It indicates the $Subspace\ Analysis\ Queries$ [9] [1],

$$SAQ(b, D, S') = |dominating(b, D, S')| - |dominated(b, D, S')|$$

$$= 2 - 2 = 0$$

SAC'07 March 11-15, 2007, Seoul, Korea

[1] In fact, the example here is a special case of that described in [9], where the point b can be any point, not necessary in the original dataset. We will discuss it in Section 5.

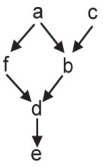

Figure 2: DAG representation in 2-d space

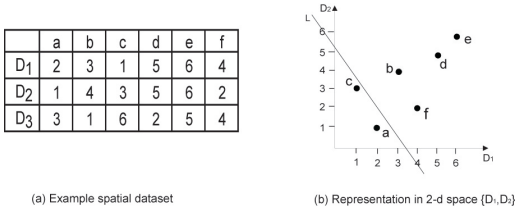

Figure 3: Example dataset

where D is the set of the original points, S' is the 2-dimensional space. The meaning of the result, i.e., 0, indicates the number of the points which are dominated by b is the same as those dominate b. Here we solve not only the $how\ many$ problem, but also the $whom$ problem. From this example, we know that the general dominant relationships of a dataset can be represented into their corresponding partial order representation (i.e., DAGs).

Our contributions in this paper are as follows:

- We generalize the dominant relationship queries proposed in [9], as **General Dominant Relationship Queries** (**GDRQs**). We find the interrelated connection between GDRQs and the partial order analysis.
- We propose efficient and effective algorithms to answer $GDRQs$ based on partial order representation of spatial datasets.
- We conduct comprehensive experiments to illustrate the effectiveness and efficiency of our methods.

The remainder of this paper is organized as follows. In Section 2, we discuss the related work. In Section 3, we present the preliminaries of this paper. A naive method based on existing strategy to answer $GDRQs$ is introduced in Section 4. The query processing strategies for generalized dominant relationship analysis using partial order representation is described in Section 5. The performance analysis are reported in Section 6. We conclude the paper in Section 7.

2. RELATED WORK

Skyline query was first introduced in [2]. The problem comes from some old classic topics, such as convex hull [13] and maximum vectors [8].

Skyline query algorithms can be classified into two categories. The first one is non-index based method, i.e., BNL [2], SFS [5], DC [2]. The second category is index based method, i.e., NN [7], BBS [11]. From the view point of dimension concerned, the existing algorithms can be also classified into two categories, i.e., full space based method [7] [11], and subspace based method [17] [12] [15]. Other related work on skyline mining includes mining skyline in distributed environments [1], skyline query in data stream [10] [14], interesting skyline points in high-dimensional space [4].

All the above works concerned only the pure dominant relationship and, outputted those points which are not "dominated" by others. Note that in addition to the original meaning in [2], "dominated" here can be a variant, i.e., k-dominant [4].

In contrast, Li et al. [9] proposed to analyze a more general dominant relationship from a microeconomic aspect. The users are always interested in not only the binary dominant relation between the points in a dataset, but also the statistical information, i.e., how many other points are dominating/dominated by a specific point. In [9], the authors proposed three basic $Dominant\ Relationship\ Queries$ ($DRQs$) and constructed a data cube, DADA, to efficiently organize the information necessary to $DRQs$. Moreover, a novel data structure, D*-tree, was proposed to fulfill efficient computation for $DRQs$.

However, in the real world, users are always interested in not only "how many" objects are dominating/dominated by a specific object, but also "whom" they are, which was however, not mentioned in [9]. This problem cannot be easily solved by using the methodologies proposed in [9] because of the large duplicate storage cost in DADA. In this paper, we propose efficient data structure and strategies to solve such kind of general dominant relationship queries based on our discovery that GDRQ has interrelated connection with partial order.

3. PRELIMINARIES

Given a d-dimension space S=$\{s_1, s_2, \ldots, s_d\}$, a set of points D=$\{p_1, p_2, \ldots, p_n\}$ is said to be a dataset on S if every $p_i \in D$ is a d-dimensional data point on S. We use $p_i.s_j$ to denote the j^{th} dimension value of point p_i. For each dimension s_i, we assume that there exists a total order relationship. For simplicity and without loss of generality, we assume smaller values are preferred [2] (i.e., MIN operation) in this paper.

DEFINITION 1 (DOMINATE). *A point p is said to dominate another point q on S if and only if $\forall s_k \in S$, $p.s_k \leq q.s_k$ and $\exists s_t \in S$, $p.s_t < q.s_t$.*

A partial order on D is a binary relation $\preceq$ on D such that, for all x,y,$z \in D$, (i) $x \preceq x$ (reflexivity), (ii) $x \preceq y$ and $y \preceq x$ imply x=y (antisymmetry), (iii) $x \preceq y$ and $y \preceq z$ imply $x \preceq z$ (transitivity). We use (D,$\preceq$) to denote the partial order set (or poset) of D. We denote by $\prec$ the strict partial order on D, i.e., $x \prec y$ if $x \preceq y$ and $x \neq y$. Given x,$y \in D$, x and y are said to be comparable if either $x \prec y$ or $y \prec x$; otherwise, they are said to be incomparable.

The Definition 1 can be translated into the ordering context as follows:

DEFINITION 2 (DOMINATE IN ORDERING CONTEXT). *A point p is said to dominate another point q on S if and only if $\forall s_k \in S$, $p.s_k \preceq q.s_k$ and $\exists s_t \in S$, $p.s_t \prec q.s_t$.*

The partial order (D,$\preceq$) can be represented by a DAG $G = (D, E)$, where $(v, \omega) \in E$ if $\omega \preceq v$ and there does not exist another value $x \in D$ such that $\omega \preceq x \preceq v$. For simplicity and without loss of generality, we assume that G is a single connected component.

DEFINITION 3 (DOMINATING SET, DGS(P, D, S')). *Given a point p, we use DGS(p, D, S') to denote the set of points from D which are dominated by p in the subspace S' of S.*

DEFINITION 4 (DOMINATED SET, DDS(P, D, S')). *Given a point p, we use DDS(p, D, S') to denote the set of points from D which dominate p in the subspace S' of S.*

The problems that we want to solve are as follows:

PROBLEM 1 (GENERAL POINT QUERY(GPQ)). *Given a dataset D, dimension space S' and a point p, find DGS(p, D, S') and DDS(p, D, S').*

Note that GPQ is the generalized model of subspace analysis queries (SAQ)[9].

EXAMPLE 1. *Consider the 3-dimensional dataset D = {a, b, c, d, e, f} in Fig. 3 (a). Given a query point b, dimension space S'=$\{D_1, D_2\}$, the dominating set DGS(b, D, S') = {d, e} and the dominated set DDS(b, D, S') = {a, c}. We will use this dataset as a running example in the rest of this paper.*

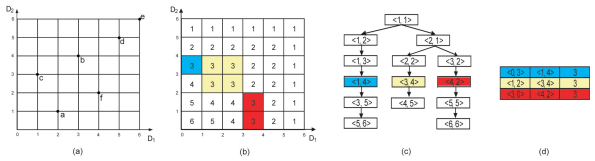

Figure 4: The strategy of DADA for GPQ (this figure is best viewed in color)

PROBLEM 2 (GENERAL LINEAR OPTIMIZATION QUERY). *Given a dataset D, dimension space S' and a plane L, find the sets of points from D which have the aggregate maximum, max(|DGS(p, D, S')|), where p is any point in the plane L, in the subspace S'.*

PROBLEM 3 (GENERAL COMPARATIVE DOMINANT QUERY). *Given two sets of points A and B in a dataset D, and dimension space S', find the sets of points from B which are dominated by some point from A in the subspace S'.*

The first problem statement is merely a special case of the second and the third. For clarity of presentation we deal first with the task of determining the dominating/dominated set based on a random point. The general linear optimization query and the general comparative dominant query are then reduced to the first problem by considering the line L and the dataset A/B as a set of points.

4. A NAIVE METHOD

To solve the problems defined in Section 3, a natural idea is to extend the framework proposed in [9]. In this section, we briefly introduce this naive strategy and then, illustrate its weak points.

The authors in [9] partition the data space by using griding strategy. For example, Fig. 4 (a) shows a dataset in 2-dimensional space (i.e., $\{D_1, D_2\}$). In Fig. 4 (b), each grid records the number of the points which current grid dominates. For instance, the colorful grids are all those which dominates three points. Instead of recording each grid information, [9] proposed D*-tree to record the compressed information (upper/lower bound of a region that dominates the same number of points). For example, the colorful grids can be partitioned into three regions, i.e., blue, yellow and red, which are represented by their upper bound, i.e., {1, 4}, {3, 4} and {4, 2}, respectively. The whole D*-tree is shown in Fig. 4 (c), which is constructed based on the rule defined in [9] (Definition 4.7). Fig. 4 (d) shows the compressed information about the three colorful regions, i.e., the lower bound (1st column), the upper bound (2nd column) and the number of points dominated (3rd column). Given a point P_{query}, to get the number of the points P_{query} dominates, it needs to start from the root of the D*-tree and move down the node with the upper bound that can dominate P_{query}. Once it knows that P_{query} is contained in a region that the node dominates, the desired number of the points which are dominated by P_{query} can be output.

Yet there are two issues arising when processing the general dominant relationship queries by using DADA's strategy. Firstly, the "whom" problem can not be efficiently solved. For example, although the colorful regions in Fig. 4 (b) all dominate three points, they have different dominating sets, i.e., {b, d, e} for blue and yellow regions and {f, d, e} for red region. Although by adding the dominating set into each node of the D*-tree can naively answer the question (as shown in Fig. 5), this simple solution will introduce serious burden of data duplication problem. Therefore, the strategy of DADA is not appropriate for the general dominant relationship analysis problem. Another issue is that the search strategy in DADA while traversing the D*-tree is inefficient.

5. A PARTIAL ORDER BASED METHOD

In this section, we propose to efficiently apply the properties of the partial order to analyze the general dominant relationship. We

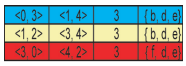

Figure 5: The extension of DADA for general dominant relationship analysis (this figure is best viewed in color)

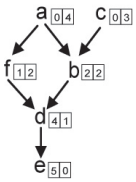

Figure 6: DAG representation of the example dataset in 2-dimensional space $\{D_1, D_2\}$

assume that the partial orders of a spatial dataset are available. In practice, we use the techniques proposed in [16] to discovery all the partial orders, which are represented in DAG format.

5.1 Querying Partial Orders

We propose several strategies to efficient answering different kinds of queries. The semantic meaning kept in the partial orders is the key used to extract the general dominant relationship.

5.1.1 General Point Query (GPQ)

Given a dataset D, a query point P_{query} and a subspace S', the most basic GPQ is to compute the points dominate or dominated by P_{query}. We can further differentiate two cases based on whether the point P_{query} is in the dataset D or not. Note that in [9], the authors did not mention the difference of these two cases, $P_{query} \in D$ and $P_{query} \notin D$. Hence, DADA deals with the two cases in the same manner.

- $P_{query} \in D$

An important observation in this case is that, if P_{query} is in D, all the general dominant relationship related to P_{query} can be easily discovered by traversing the DAG in a specific subspace. No disk access is performed because we don't need to check their individual value of each dimension.

As an example, Fig. 6 shows the DAG representation in subspace $\{D_1, D_2\}$. To facilitate the counting process, the numbers of points dominating/dominated by current node (point) are inserted into each node. This process is executed in the precomputed-mode. Suppose the query point is b, we can get the points dominated by b immediately, which is 2. Upon users are interested in whom these two points are, it goes downward following the out-link of b, and gets the dominating set of b as $\{d, e\}$. In DADA [9] framework, however, it needs to traverse the D*-tree to get the corresponding class. For example, assume the query point is b, the order of the traversed nodes in D*-tree, as shown in Fig. 4 (c), is $\{\langle 1,1\rangle,\langle 2,1\rangle,\langle 2,2\rangle,\langle 3,4\rangle\}$. Then it finds the dominating set of b by checking the class of $\{\langle 3,4\rangle\}$. Obviously, DADA consumes more time compared with our strategy when the query point p is in D.

- $P_{query} \notin D$

As explained in Section 4, DADA [9] can not efficiently distinguish different dominating sets, which have the same number of points dominated. We propose to solve this problem by traversing the partial order representation (DAGs). To illustrate this, we parse the example DAG in Fig. 6, by adding some $Virtual\ Node$s, which act as real nodes that dominates different set of points in the original dataset. Fig. 7 (a) shows all the virtual nodes (denoted as V_i). We can see that for the nodes which dominate three points,

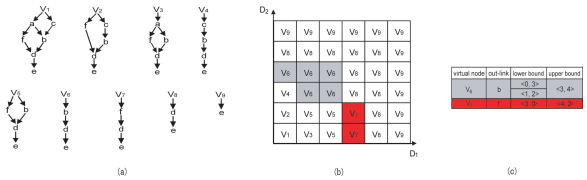

Figure 7: All virtual nodes representation in 2-dimensional space $\{D_1, D_2\}$ (this figure is best viewed in color)

Algorithm 1: Creating Virtual Nodes

Input: Partial order representation DAG
Output: Virtual nodes set V
1. for each partial order DAG and its subspace S' do
2. initialize i=1
3. traverse GAG based on breadth first order
4. delete all the nodes and pathes traversed
5. if the left part (LP) is still a single connected component
6. insert a new virtual node V_i into the virtual node table
7. merge top nodes of LP to find smallest value of each dimension in S', record it as upper bound U_i of V_i
8. between the original point and U_i, find blank regions not occupied by other virtual nodes, record left bottom corners of these blank regions as the low bounds of V_i
9. i++

Figure 8: Creating Virtual Nodes

there are two virtual nodes (i.e., V_6 and V_7). Fig. 7 (b) illustrates the virtual nodes effect in grid model. It can be deemed as a semantic format of that used in DADA (i.e., as shown in Fig. 4 (b)).

To store the information of these virtual nodes, a naive solution is to store all the DAGs shown in Fig. 7 (a), which is obviously impractical because of the data duplication. We propose to use a more compressive method that, with the help of the DAG shown in Fig. 6, we only need to save the out-link node and the occupied region (represented by its upper bound/lower bound) of each virtual node. For example, Fig. 7 (c) shows the outlinks of the virtual nodes V_6 and V_7 and their occupied regions. Suppose we know that a query point P_{query} is in the region occupied by V_6, then the point b in Fig. 6 will be first visited according to the out-link of V_6. The remain process will be the same as that in the first case, when $P_{query} \in D$. Note that here the number of the points in the dominated set of V_6 is 2+1=3 because b itself is dominated by V_6.

The pseudo code of creating the virtual nodes is shown in Fig. 8. To create the virtual nodes, we traverse the DAG (i.e., Fig. 6) following the Breadth First Order (line 3). We delete all the nodes and the pathes traversed, and judge if the left part of the DAG is a single connected component because a virtual node can not exist if there is a gap appears in the left part of the DAG. For example, suppose after we successfully get the virtual node V_4, the next traverse order should be $\{a, b\}$. If we delete the nodes a and b, then the node c should be also deleted. The proof can be easily got by contradiction. Because we do not traverse the path which cover the node c, we terminate the routing path. This is the reason why we need to check that the left part of the DAG is a single connected component or not (line 5). We insert a new virtual node V_i into a table (line 6), and record the occupied regions of V_i by using the upper bound and lower bound of the regions (line 7 and line 8). Note that the virtual node creation algorithm is executed in the preprocess model for the case when a query point $p \notin D$. The pseudo code of General Point Query (GPQ) processing algorithm is shown in Fig. 9. It is self explainable and due to limited space, we skip the detail.

5.1.2 General Linear Optimization Query (GLOQ)

The GLOQ problem in this section and the GCDQ problem in the next section can be solved by extending the methods used in the last section.

Algorithm 2: GPQ Processing

Input: Partial order representation DAG, a query point P_{query}, a subspace S'
Output: GPQ(P_{query}, D, S')
1. if $P_{query} \in D$
2. for each child node c_i of P_{query} in DAG and its subspace S'
3. put c_i in the result set GPQ(P_{query}, D, S')
4. else // when $P_{query} \notin D$
5. search P_{query} to find its corresponding representative virtual node V_i
6. if (the out-link node o_i of V_i exists)
7. put o_i in the result set GPQ(P_{query}, D, S')
8. for each child node c_i of P_{query} in DAG and its subspace S'
9. put c_i in the result set GPQ(P_{query}, D, S')

Figure 9: GPQ processing algorithm

We deal with the GLOQ processing based on such a fact: if the upper bound of an occupied region and the origin point fall on the same side of L, then all the representative virtual nodes enclosed by the the upper bound of an occupied region and the origin point can not intersect L.

Note that there may be existing multiple representative virtual nodes which dominate the most points in D, so we need to store the intermediate result. The whole work flow of the original version algorithm is similar to the one proposed in [9] (Section 5.1). We need first get all possible virtual nodes which dominate the most points in D. Once we get these virtual nodes, then the remaining task is the same as that in GPQ processing.

5.1.3 General Comparative Dominant Query (GCDQ)

To handle the GCDQ problem, we need to pre-construct two DAGs for both A and B, respectively. Because of the definition of skyline, all the non-skyline points in A are dominated by some skyline points in the same dataset, A. Hence, we first need to extract the skyline points of A. This can be easily done by checking the first level of DAG representation of A because the nodes there are not dominated by any other points, i.e., a and c in Fig. 6. Then the problem reduces to find GPQ(P_{query}, D, S'), where P_{query} is the skyline point of A. This simplified problem has been solved in the former part of Section 5.2. The final result set is the union of both.

6. PERFORMANCE ANALYSIS

We performed the experiments using a 1.6GHz Intel Pentium(R)M PC with a 3G memory, running Windows XP. All the algorithms were written in C++, and compiled in an MS Visual C++ environment. We employ the three most popular synthetic benchmark datasets, *independent*, *correlated* and *anti-correlated* [2], with dimensionality d in the range [3, 7] and cardinality in the range [100k, 500k]. The default values of dimensionality and cardinality were 5 and 100k, respectively. Detailed implementation of the algorithms used to compare is described as follows:

1. *Naive. Naive* was tested with the extension of DADA [9], by storing the dominated/dominating points in the corresponding class, as explained in Section 4.

2. *ParOrder. ParOrder* was implemented as described in this paper (as explained in Section 5).

6.1 Effectiveness of Compression

In this experiment, we explored the compression benefits of $ParOrder$ compared with $Naive$ while processing general dominant relationship queries. Following the definition used in [9], we use a metric called compression ratio, which is the size of the $ParOrder$ as a proportion of $Naive$. Hence, a smaller ratio indicates better compression.

Fig. 10 shows the compression ratio for each type of data distribution with varying dimensionality and cardinality. Among the three types of data, correlated data is always the winner which

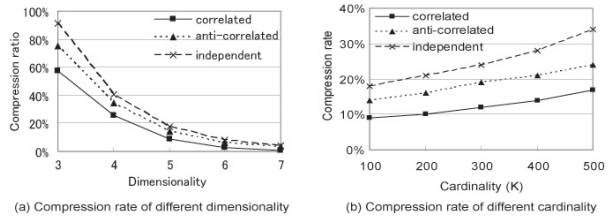

Figure 10: Compression ratio of varying dimensionality and cardinality

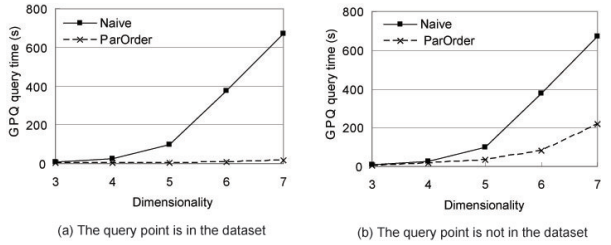

Figure 11: Query processing (anti-correlated) of GPQ queries

has the most compressive effect. The reason is that many cells in $ParOrder$ dominates the same set of points for correlated data. Independent data has the least compression as it is more difficult for cells to dominate the same set of points and hence, $ParOrder$ can not efficiently summarize the data into partial order representation. The performance of anti-correlated data is between those of the correlated and independent data on compression rate. As illustrated in Fig. 10 (a), the higher the dimensionality is, the better the compression ratio will be. The reason is due to the more sparse data cube as dimensionality increasing. Fig. 10 (b) illustrates the compression effect with different cardinality.

6.2 Query Performance

In this section, we evaluated the query answering performance of $ParOrder$. There are two cases while testing the GPQ queries, the query point p is in the original spatial dataset D or not.

To test the effect of GPQ query, we randomly selected 10,000 different points from D for the first case and generated 10,000 different points for the second case. For each point p, we queried its dominating number in all subspaces. Due to limited space, we present only the result on the anti-correlated dataset.

Fig. 11 (a) and Fig. 11 (b) show the query time against dimensionality when the query point $p \in D$ and $p \notin D$, respectively. We can see that the $ParOrder$ algorithm outperforms the $Naive$ in both cases. This is because the compact representation of partial orders can lead to faster routing. From Fig. 11 (a) and Fig. 11 (b), we can also know that the performance difference between the $Naive$ method and the $ParOrder$ in the first case is much larger than that in the seconde case. The reason is that we can directly traverse the DAGs in the first case, instead of judging the virtual nodes in the second case.

To test the effect of GLOQ query, we randomly generated 10,000 different GLOQs based on the synthetic dataset. Fig. 12 (a) show the query time against dimensionality. We can see that the $ParOrder$ approach is always better than the $Naive$.

To test the effect of GCDQ query, we randomly generated 10,000 different GCDQs based on the synthetic dataset with in A and B containing 100 points each. Fig. 12 (b) show the query time against dimensionality. As expected, the $ParOrder$ approach performs the best.

7. CONCLUSIONS

In this paper, we have introduced General Dominant Relationship Analysis, which could not be easily solved by existing strategies. Due to the interrelated connection between the partial order and the dominant relationship, we have proposed efficient data structures and strategies to answer the general dominant relationship queries. The experimental results and performance study confirmed the efficiency and effectiveness of our strategies.

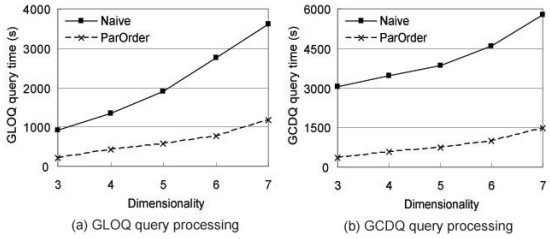

Figure 12: Query processing (anti-correlated) of GLOQ and GCDQ queries

8. REFERENCES

[1] W.T. Balke, U. Guentzer and J.X. Zheng. Efficient distributed skylining for web information systems. In *EDBT*, pages 256-273, 2004.

[2] S. Borzsonyi, D. Kossmann and K. Stocker. The skyline operator. In *ICDE*, pages 421-430, 2001.

[3] G. Casas-Garrica. Summarizing sequential data with closed partial orders. In *SDM*, pages 380-391, 2005.

[4] C.Y. Chan, H.V. Jagadish, K.L. Tan, A.K.H. Tung and Z. Zhang. Finding k-Dominant skylines in high dimensional space. In *SIGMOD*, pages 503-514, 2006.

[5] J. Chomicki, P. Godfrey, J. Gryz and D. Liang. Skyline with presorting. In *ICDE*, pages 717-719, 2003.

[6] W. Kieβling. Foundations of preferences in database systems. In *VLDB*, pages 311-322, 2002.

[7] D. Kossmann, F. Ramsak and S. Rost. Shooting stars in the sky: An online algorithm for skyline queries. In *VLDB*, pages 275-286, 2002.

[8] H.T. Kung, F. Luccio and F.P. Preparata. On finding the maxima of a set of vectors. In *JACM*, 22(4): pages 469-476, 1975.

[9] C. Li, B.C. Ooi, A.K.H. Tung and S. Wang. DADA: A data cube for dominant relationship analysis. In *SIGMOD*, pages 659-670, 2006.

[10] X. Lin, Y. Yuan, W. Wang and H. Lu. Stabbing the sky: Efficient skyline computation over sliding windows. In *ICDE*, pages 502-513, 2005.

[11] D. Papadias, Y. Tao, G. Fu and B. Seeger. An optimal and progressive algorithm for skyline queries. In *SIGMOD*, pages 467-478, 2003.

[12] J. Pei, W. Jin, M. Ester and Y. Tao. Catching the best views of skyline: A semantic approach based on decisive subspaces. In *VLDB*, pages 253-264, 2005.

[13] F. Preparata and M.I. Shamos. Computational geometry: In introduction. *Springer-Verlag*, 1985.

[14] Y. Tao and D. Papadias. Maintaining sliding window skylines on data streams. In *TKDE*, 18(3): pages 377-391, 2006.

[15] T. Xia and D. Zhang. Refreshing the Sky: The compressed skycube with efficient support for frequent updates. In *SIGMOD*, pages 491-502, 2005.

[16] Z. Yang, B. Wang, and M. Kitsuregawa. ParCube: A Partial Order Model Based Data Cube for General Dominant Relationship Analysis. *Technical Report*, University of Tokyo, Japan, 2006.

[17] Y. Yuan, X. Lin, Q. Liu, W. Wang, J.X. Yu and Q. Zhang. Efficient computation of the skyline cube. In *VLDB*, pages 241-252, 2005.