Towards Efficient Dominant Relationship Exploration of the Product Items on the Web

Zhenglu Yang and **Lin Li** and **Botao Wang** and **Masaru Kitsuregawa**
Dept. of Info. and Comm. Engineering, University of Tokyo
4-6-1 Komaba, Meguro-ku, Tokyo, Japan 153-8505
{yangzl, lilin, botaow, kitsure}@tkl.iis.u-tokyo.ac.jp

Abstract

In recent years, there has been a prevalence of search engines being employed to find useful information in the Web as they efficiently explore hyperlinks between web pages which define a natural graph structure that yields a good ranking. Unfortunately, current search engines cannot effectively rank those relational data, which exists on dynamic websites supported by online databases. In this study, to rank such structured data (i.e., find the "best" items), we propose an integrated online system consisting of compressed data structure to encode the dominant relationship of the relational data. Efficient querying strategies and updating scheme are devised to facilitate the ranking process. Extensive experiments illustrate the effectiveness and efficiency of our methods. As such, we believe the work in this paper can be complementary to traditional search engines.

Introduction

Suppose you are buying a digital camera from a website (i.e., www.bestbuy.com) and you are looking for one that is cheap and with high sensor resolution to take beautiful pictures. Unfortunately, these two goals are complementary to one another as cameras with more megapixels tend to be more expensive. Moreover, you would appreciate the search if only a few "best" goods are recommended by the website's system with regard to your preference, instead of checking all the items manually. Intuitively, these "best" goods are all cameras that are not worse than any other camera in both attributes, i.e., price and sensor resolution. For instance, Fig. 1 shows some sample cameras and among them, only the items c (PowerShot A630) and d (EOS Digital Rebel XTi) should be the candidates recommended to the user.

This problem of searching for such "best" items, can be traced back to the 1960s in the theory field. The set of these "best" items is called the Pareto set and the objects are called maximal vectors (Bentley *et al.* 1978). However, these main memory algorithms are inefficient for online query processing, due to the frequently updated data.

Recently, the *skyline* operator (Borzsonyi, Kossmann, & Stocker 2001) was proposed to tackle the maximal vector problem in database context. Efficient skyline querying methodologies have been studied extensively (i.e., (Kossmann, Ramsak, & Rost 2002; Papadias *et al.* 2003)). All of these works, however, concerned only the pure binary relationship, i.e., a product item p is whether or not worse than (dominated by) others. Interestingly, Li et al. (Li *et al.* 2006) proposed to analyze a more general dominant relationship in a business model, that users preferred the details of the dominant relationship more, i.e., an item p dominates (and vice versa, is dominated by) how many other items. Here we show an example.

Item ID	Product name	Company	Price ($)	Weight (g)	Sensor resolution (M)
a	PowerShot SD630	Canon	349.99	142	6
b	Cyber-shot DSC-H5/B	Sony	479.99	404	7.2
c	PowerShot A630	Canon	269.99	245	8
d	EOS Digital Rebel XTi	Canon	799.99	509	10.1
e	DSLR-A100K	Sony	899.99	545	10
f	Cyber-shot DSC-M2	Sony	649.99	180	5.1

Figure 1: Digital camera example

Example 1: Consider you are a manager of Sony corporation. You want to know the business position of a digital camera b (Cyber-shot DSC-H5/B) in the current market with regard to your preference, i.e., price and weight, by checking how many other items are better/worse than b and what they are. For the sample cameras shown in Fig. 1, you can deduct the conclusion that item b is better than items d and e but worse than items a and c with regard to your preference[1].

The dominant relationship analysis in Example 1 plays an important role in business decisions. Unfortunately, the existing work cannot tackle the dominant relationship analysis in a dynamic environment (i.e., the Web). Relationships are recomputed naively each time upon update, resulting in extreme inefficiency with the increase of online databases. In this paper we aim to efficiently construct an integrated system to solve the issue in frequently updated environments.

The keyword-based search engines (i.e., Google) cannot be employed here because they lose effect when tackling relational data. For the purposes of this paper, we assume the attribute sets of products are available in structured format (i.e., Fig. 1). This assumption is reasonable due to

[1]Note that the analysis here can be further used to determine the price of a new product, which should be competitive in the current market while reserving the most profit.

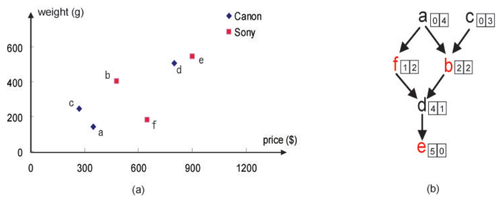


Figure 2: Product attributes in 2-dimensional space and the corresponding partial order

the existence of the techniques that convert unstructured text into structured tables (i.e., wrapper programs (Kushmerick, Weld, & Doorenbos 1997)).

We found the interrelated connection between the dominant relationship and the partial order (Yang, Wang, & Kitsuregawa 2007). To illustrate the idea, here we show a simple example. Fig. 2 (a) represents the product items in two attributes (dimensions) space, i.e., price and weight. Fig. 2 (b) is the corresponding partial order (encoded as DAG format). We can know that item b dominates items d and e and is dominated by items a and c, by checking the $out\text{-}link$ and $in\text{-}link$ of b, respectively. However, (Yang, Wang, & Kitsuregawa 2007) does not deal with dominant relationship analysis in dynamic environments. Furthermore, how to construct partial order representation of spatial datasets is not tackled. In this paper we aim to explore how to efficiently find, update and query such succinct representative partial orders with formal justification.

Different users may have different preferences. To cope with this problem, we propose a compressed data cube to encode all the possibilities in a concise format and devise efficient strategies to extract the information.

There is another methodology, faceted search (Yee *et al.* 2003), that can be employed to support shopping by exploring different attributes of the products. Faceted search combines navigational and direct search to leverage the best of both approaches. However, it does not utilize the property of the numerical attributes, i.e., partial order, to refine the results. Our contributions in this paper are as follows:

- We formally justify the interrelated connection between the dominant relationship and the partial order analysis. We propose effective methods to construct a data cube, $ParCube$, which concisely represents the dominant relationship as partial order representation (DAGs).
- Based on $ParCube$, we efficiently construct an online query system, which employs an effective update scheme to maintain $ParCube$ in a dynamic environment.
- We introduce efficient query processing strategies on top of $ParCube$ to answer the general dominant relationship queries in a dynamic environment.
- We conduct comprehensive experiments to confirm the efficiency of our strategies by using synthetic and real data.

The remainder of this paper is organized as follows. After discussing the related work, we introduce the preliminaries and then propose several strategies to efficiently construct, query and maintain an integrated system for dominant relationship analysis. The performance analysis is reported in Section 5. We conclude the paper in Section 6.

Related work

Deep Web. The framework of this paper can be seen as complementary to the information retrieval systems for $Deep\ Web$ (Bergman 2001), whose information is buried far down on dynamically generated websites, and cannot been found by standard search engines (i.e, Google). Specifically, our work is a postprocess of data extraction (Kushmerick, Weld, & Doorenbos 1997), which uses wrapper softwares to convert the unstructured web data into structured format data.

Maximum vector and Skyline. The maximum vector problem (Kung, Luccio, & Preparata 1975) is a special case of dominant relationship analysis. There are some other related issues, i.e., convex hull (Preparata & Shamos 1985). The $skyline$ query (Borzsonyi, Kossmann, & Stocker 2001) was proposed to tackle the maximal vector problem in database context. However, most existing works concerned only the pure dominant relationship and, output those items which are not "dominated" by others. In contrast, Li et al. (Li *et al.* 2006) proposed to analyze a general dominant relationship from a microeconomic aspect in static environments.

In real dynamic environments (i.e., the Web), the attribute values are always updates, which makes it very difficult to analyze the dominant relationship. Moreover, users are always interested in not only "how many" items are dominating/dominated by a specific item, but also "whom" they are. These issues cannot be efficiently tackled by using existing methodologies.

Partial order mining. Partial order has appeared in many computational models (i.e., concurrent models (Lamport 1978)). In this paper, we mainly consider the problem of converting the spatial dataset into partial order representation, which are then queried to get the dominant relationship efficiently. To our best knowledge, there has been no work on this problem. An interesting study investigated the problem of mining a small set of partial orders globally fitting data best (Mannila & Meek 2000). More recently, Casas-Garriga et al. (Casas-Garriga 2005) was intended for discovering several small partial orders from a set of sequences instead of only one that describes all or most of the set. Yet different from this paper, they did not further explore the partial orders for a specific purpose (i.e., dominant relationship extraction).

Preliminaries

Given a d-dimension (attribute) space S=$\{s_1, s_2, \ldots, s_d\}$, a set of product items 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 item on S. We use $p_i.s_j$ to denote the j^{th} dimension value of item 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. Moreover, we assume that the distinct value condition holds (Yuan *et al.* 2005; Pei *et al.* 2005).

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) $x \preceq y$

Table 1: Sample Product Items Dataset

	a	b	c	d	e	f
D_1	2	3	1	5	6	4
D_2	1	4	3	5	6	2
D_3	3	1	6	2	5	4

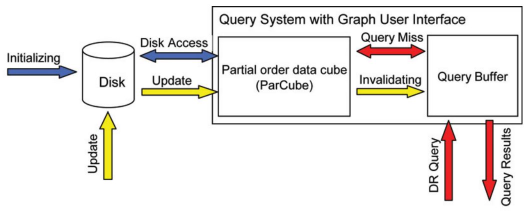


Figure 3: The proposed system based on the partial order data cube (ParCube) (this figure is best viewed in color)

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.

Definition 1 (dominant in ordering context) *A product p is said to dominate another product 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 2 (dominant set, DGS(p, D, S')) *Given a product p, we use DGS(p, D, S') to denote the set of products from D which are dominated by p in the subspace S' of S.*

Definition 3 (dominated set, DDS(p, D, S')) *Given a product p, we use DDS(p, D, S') to denote the set of products from D which dominate p in the subspace S' of S.*

The problem that we want to solve is as follows:

Problem 1 (General Point Query(GPQ)) *Given a dataset D, dimension space S', and a product p, find DGS(p, D, S') and DDS(p, D, S').*

Note that GPQ is the generalized model of Subspace Analysis Queries (SAQ)(Li *et al.* 2006).

Example 1 *Consider the 3-dimensional dataset D = {a, b, c, d, e, f} in Table 1[2]. 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 hereafter.*

Integrated system building, maintaining and querying

In this section, we introduce our online query system, which is illustrated in Fig. 3. There are three parts involved with this system: 1) System construction (symbolized as blue color); 2) Query processing (symbolized as red color); and 3) System updating (symbolized as yellow color).

[2]D_i denotes the ith attribute. For simplicity, we use small integer to simulate items' values on the attributes for convenience of description in this paper.

Table 2: Constructing $ParCube$ algorithm

INPUT:	Spatial dataset D
OUTPUT:	Data cube $ParCube$
1.	Convert D to the corresponding sequence dataset D' // process 1
2.	Apply PrefixSpan (Pei *et al.* 2001) to get the sequential patterns from D', merge them to the $local\ maximal\ sequential\ sequences$ as $SeqCube$ // process 2
3.	Apply the algorithm proposed in (Casas-Garriga 2005) to get the partial order representation (DAGs) as $ParCube$ // process 3

System Construction

We propose to apply strategies from another research context, sequential pattern mining (Agrawal & Srikant 1995), to get the partial order representation cube ($ParCube$) from a spatial dataset. There are three processes for $ParCube$ construction.

The first process is to convert the spatial dataset to the sequence dataset. With a k-dimensional dataset, we simply get a k-customer sequence dataset, by sorting the objects in each customer (dimension) according to their value in ascending order. For example, Fig. 4 (b) shows the converted sequence dataset of the spatial dataset in Fig. 4 (a). Note the specific values of these points on k-dimension are resident on the disk. Efficient management methods (i.e., R-tree (Guttman 1984)) are employed, as will be explained in this paper.

Theorem 1 *The converted sequence dataset records all the dominant relationships of the points in the spatial dataset.*

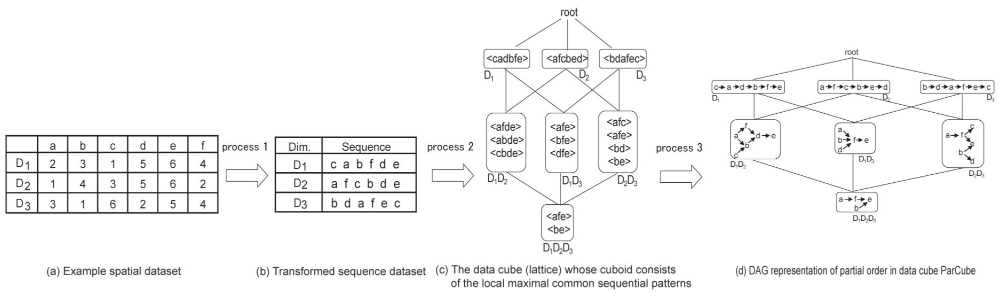
Proof [Sketch of Proof] Trivial because the small-large pair (dominant) relationship in the spatial dataset is equivalent to the early-late pair (dominant) relationship in the converted sequence dataset. □

The second and the third processes aim to determine a partial order that describes the point set in the subspace S' of data space S in D'. In this paper, we apply the approach in (Casas-Garriga 2005) with a minor modification, that we mine general sequential patterns (Agrawal & Srikant 1995). In the second process, we discover the sequential patterns from the transformed sequence dataset by applying PrefixSpan algorithm (Pei *et al.* 2001)[3], which is the state-of-the-art method. To save space, we merge these sequential patterns as $local\ maximal\ sequential\ sequences$, which are not the subsequence of other sequential sequences. For example, in subspace {D_1, D_2}, although $\langle afd \rangle$, $\langle afe \rangle$, $\langle ade \rangle$, and $\langle fde \rangle$ are length-3 patterns, we merge them as length-4 $local\ maximal\ sequential\ sequences$, as $\langle afde \rangle$. The resultant data cube ($SeqCube$) from the second process is shown in Fig. 4 (c).

Theorem 2 *SeqCube records all the dominant relationship of the points in the sequence dataset D.*

Proof [Sketch of Proof] (Proof by Contradiction.) For simplicity, we only prove for a specific subspace of

[3]We skip the details of PrefixSpan here. Interested users can refer to (Pei *et al.* 2001).

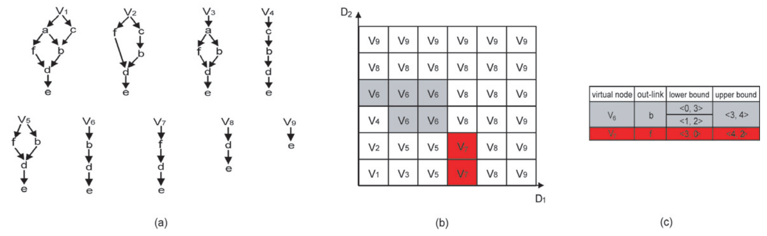


	a	b	c	d	e	f
D_1	2	3	1	5	6	4
D_2	1	4	3	5	6	2
D_3	3	1	6	2	5	4

Dim.	Sequence
D_1	c a b f d e
D_2	a f c b d e
D_3	b d a f e c

process 1
process 2
process 3
root
<cadbfe>
<afcbed>
<bdafec>
D1
D2
D3
<afde> <abde> <cbde>
<afe> <bfe> <dfe>
<afc> <afe> <bd> <be>
D1D2
D1D3
D2D3
<afe> <be>
D1D2D3
(a) Example spatial dataset
(b) Transformed sequence dataset
(c) The data cube (lattice) whose cuboid consists of the local maximal common sequential patterns
(d) DAG representation of partial order in data cube ParCube

Figure 4: The result representation of each process for the example spatial dataset



virtual node	out-link	lower bound	upper bound
V_6	b	<0, 3> <1, 2>	<3, 4>
V_7	f	<3, 0>	<4, 2>

Figure 5: All virtual nodes representation in 2-dimensional space $\{D_1, D_2\}$ (this figure is best viewed in color)

$SeqCube$. Assume to the contrary there is a dominant relationship between two points, a dominates b in a subspace S', that is not represented in the cuboid S' of $SeqCube$. This means the sequential pattern $\langle ab \rangle$ is not listed in S' of $SeqCube$, which contradicts our assumption the sequential pattern mining can find all the sequential patterns. □

In the third process, the combinations of the *local maximal sequential sequences* are enumerated to generate partial orders with DAG representation, by applying the method proposed in (Casas-Garriga 2005). The resultant data cube ($ParCube$) for the example dataset is shown in Fig. 4 (d).

Theorem 3 *ParCube records all the dominant relationships of the points in the spatial dataset D.*

Proof [Sketch of Proof] Proof can be deduced based on Theorem 1 and Theorem 2 in this paper, and in (Casas-Garriga 2005). □

Complexity analysis: The cost of the $ParCube$ construction is mainly dependent on the (maximal) sequential pattern mining process, which is #P-complete (Yang 2004).

The pseudo code of constructing $ParCube$ is shown in Table 2, where the three lines describe the three processes and can be justified based on Theorem 1, 2 and 3, respectively.

Query Processing

We differentiate the two cases based on whether the query point P_{query} is in the dataset D.

- $P_{query} \in D$

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. As an example, Fig. 2 (b) shows the DAG representation in subspace $\{D_1, D_2\}$. To facilitate the counting, the numbers of points dominating/dominated by current nodes are inserted into each node.

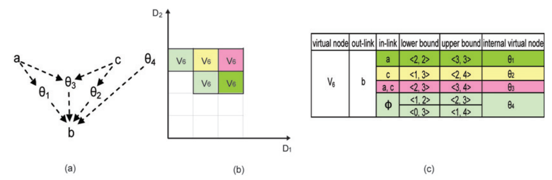
- $P_{query} \notin D$

We proposed to solve this problem by traversing the partial order representation (DAGs) with semantic cutting (Yang, Wang, & Kitsuregawa 2007). To illustrate this, we parse the example DAG in Fig. 2 (b) by adding some $Virtual\ Node$s which act as real nodes that dominate different sets of points in the original dataset. Fig. 5 (a) shows all the virtual nodes (denoted as V_i). We can see that for the nodes which dominate three points, there are two virtual nodes (i.e., V_6 and V_7). Fig. 5 (b) illustrates the virtual nodes effect in grid model.

To store the information of these virtual nodes, we use a more compressive method that, with the help of the DAG of the local maximal sequential sequence shown in Fig. 2 (b), 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. 5 (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. 2 (b) will be first visited according to the out-link of V_6. The remainder of the process will be the same as that in the first case, when $P_{query} \in D$.

Complexity analysis: For the scenario when $P_{query} \in D$, the time complexity is $O(1)$, and when $P_{query} \notin D$, the time complexity is $O(d \cdot \log n)$ in the worst case, where d is the dimension of the dataset, and n is the number of the virtual nodes. The space complexity is $O(n)$.

Proof [Sketch of Proof] When $P_{query} \in D$, the number of dominated nodes can be got by following the out-links of P_{query} in DAGs. Hence, the time complexity is $O(1)$. When



virtual node	out-link	in-link	lower bound	upper bound	internal virtual node
V_6	b	a	<2, 2>	<3, 3>	θ_1
		c	<1, 3>	<2, 4>	θ_2
		a, c	<2, 3>	<3, 4>	θ_3
		ϕ	<1, 2>	<2, 3>	θ_4
			<0, 3>	<1, 4>	

Figure 6: Inserting a node (this figure is best viewed in color)

$P_{query} \notin D$, the time complexity is mainly dependent on the time used to determine the virtual node corresponding to P_{query}, which is indeed search in R-tree. The time complex is $O(d \cdot \log n)$ and the space complex is $O(n)$. □

System Maintenance

The complete $ParCube$ is recomputed naively upon each update. Such "blind" updating method is extremely inefficient. We propose our effective update scheme for $ParCube$. We differentiate three cases when update happens: node deleting, inserting and modifying.

- Deleting a node θ.

To tackle this case is straightforward: if we want to delete θ from a DAG, we only need to link the parents of θ and the children of θ, respectively. Moreover, we need another traverse of the updated DAG to remove those collision edges and update the tuples of each node (i.e., the number of points dominating/dominated current node).

- Inserting a node θ.

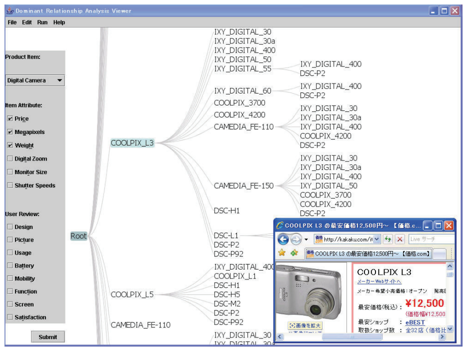
Inserting a node θ into a DAG can be naively treated as the query processing when $\theta \notin D$. However, we can only get those points which are dominated by θ (i.e., downward of θ). For instance, if we know a new node is located in the region of the virtual node V_6 in Fig. 5 (b), we still cannot determine its exact position in the corresponding DAG in Fig. 2 (b) (i.e., between a and b, or between c and b, etc). Therefore, we need to further explore the topology of the DAG to locate the exact position of θ.

Fig. 6 shows all the four possibilities when a new node θ exists in the region of the virtual node V_6. The nodes, $\theta_1 \sim \theta_4$, can be deemed as the internal virtual nodes of V_6, whose positions are illustrated in Fig. 6 (a). Given a new node θ, we use the same strategies as those used to find the corresponding internal virtual node of θ, by range search based on lower/upper bound in Fig. 6 (c). For example, if a new node θ is tested to be existing in the region $\{\langle 2,2 \rangle, \langle 3,3 \rangle\}$, then we have θ=θ_1 and its parent (in-link) should be a and child (out-link) should be b.

- Modifying a node θ.

Modifying a node can simply be tackled as a sequel execution of node deleting and node inserting.

Complexity analysis: The worst case updating time complexity is $O(d \cdot n_{iv} \cdot \log n_{iv})$, where d is the number of dimensions involved, and n_{iv} is the number of node intervals in R-tree. The space complexity is $O(n_{in})$, where n_{in} is the number of internal virtual nodes.



Product Item: Digital Camera

Item Attribute:
- [✓] Price
- [✓] Megapixels
- [✓] Weight
- [ ] Digital Zoom
- [ ] Monitor Size
- [ ] Shutter Speeds

User Review:
- [ ] Design
- [ ] Picture
- [ ] Usage
- [ ] Battery
- [ ] Mobility
- [ ] Function
- [ ] Screen
- [ ] Satisfaction

COOLPIX L3
メーカーWebサイトへ
メーカー希望小売価格：オープン
最安価格(税込)：¥12,500
(価格幅¥12,500
最安ショップ ： eBEST
取扱ショップ数 ： 全32店（価格比

Figure 7: Visualization of our system

Performance Analysis

We performed the experiments using an Intel Core 2 Duo 2.33GHz PC with 2G memory. All the algorithms were written in C++. We conducted experiments on both real and synthetic datasets by comparing two algorithms, $ParOrder\_Web$ and $Naive$[4]. Detailed of the datasets used to test is described as follows:

1. *Real Data.* The real dataset was extracted from the web sites of several stores[5]. The retrieved products include Note Computer (1419 items, 16 attributes), Digital Camera (683 items, 14 attributes), TV (710 items, 10 attributes), DVD Player (432 items, 12 attributes).
2. *Synthetic Data.* We employ the three most popular synthetic benchmark datasets in skyline query research context, $independent$, $correlated$ and $anti\text{-}correlated$ (Borzsonyi, Kossmann, & Stocker 2001), with dimensionality d in the range [3, 6] and cardinality as 100K.

Effectiveness of Our System

To convenient users to find the dominant relationship between items with regard to their preferences, we built a system with a graphical user interface based on $Prefuse$ toolkit[6]. Fig. 7 illustrates the snapshot for one user query (Product = "Digital Camera" $\wedge$ skyline attributes: Price $\vee$ Sensor Resolution $\vee$ Weight). The dominant relationship tree shown in the middle of the page clearly illustrates the business status of each product item. The node on the left part of the tree dominates (is better than) the node on the right. Furthermore, it is easy to check each item's information by clicking on its represented node and meanwhile, browse its dominating nodes.

Query Performance

In this section, we evaluated the query answering performance of $ParOrder\_Web$. There are two cases while querying, the query point p is in the original spatial dataset D or not. Note that the difference of the two methods can be

[4]*ParOrder_Web* was implemented as described in this paper and *Naive* was tested with the extension of DADA (Li *et al.* 2006)

[5]www.kakaku.com, www.rakuten.co.jp and www.biccamera.com

[6]http://www.prefuse.org

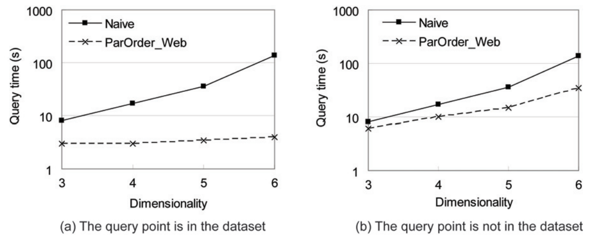


Figure 8: Query processing (anti-correlated) of GPQ queries

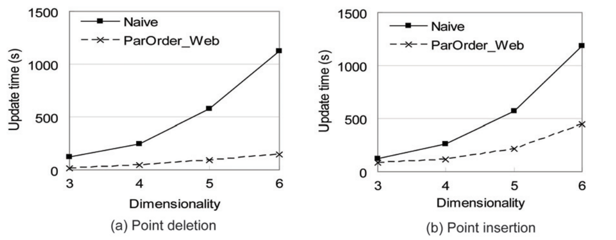


Figure 9: System maintenance on the anti-correlated dataset

finely illustrated on large datasets and hence, we show the result on large synthetic data.

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 items in all subspaces. Due to limited space, we present only the result on the anti-correlated dataset.

Fig. 8(a) and Fig. 8 (b) show the query time against dimensionality when the query point $p \in D$ and $p \notin D$, respectively. We can see that the $ParOrder\_Web$ algorithm outperforms the $Naive$ in both cases. This is because the compact representation of partial orders can lead to faster routing. From Fig. 8, we can also know that the performance difference between the $Naive$ method and the $ParOrder\_Web$ in the first case is much larger than that in the seconde case. The reason is that we can traverse the DAGs in the first case, instead of judging the virtual nodes in the second case.

Efficiency of System Maintenance

In this experiment, we evaluated the efficiency of our scheme to system maintenance. We randomly selected 10,000 different points from D to simulate the deletion case, and generated 10,000 different points to simulate the insertion case. From Fig. 9, we can see that the $ParOrder\_Web$ algorithm outperforms the $Naive$ in both cases. The reason is due to the compact representation of partial orders, which can lead to less modification. Due to the curse of dimensionality, both of these two methods become inefficient when the dimensionality increases. Another issue we observed is that the update cost is relative high due to the precise exploration of dominant relationship for both algorithms.

We have also explored the compression benefits of $ParOrder\_Web$ compared with $Naive$. Because we used a more concise format, i.e., partial order, to record the dominant relationship among the items, our methodology showed superior performance compared the naive one (Yang, Wang, & Kitsuregawa 2007).

Conclusions

We have proposed effective methods to construct a data cube, $ParCube$, which concisely represents the dominant relationship as partial orders. We constructed an online query system, which employs an effective update scheme to maintain $ParCube$. We introduced efficient query processing strategies in a dynamic environment. The performance study confirmed the efficiency of our strategies.

References

Agrawal, R., and Srikant, R. 1995. Mining sequential patterns. In *ICDE*, 3–14.

Bentley, J. L.; Kung, H.; Schkolnick, M.; and Thompson, C. 1978. On the average number of maxima in a set of vectors and applications. *Journal of ACM* 25(4):536–543.

Bergman, M. K. 2001. The deep web: Surfacing hidden value (white paper). *Journal of Electronic Publishing* 7(1):536–543.

Borzsonyi, S.; Kossmann, D.; and Stocker, K. 2001. The skyline operator. In *ICDE*, 421–430.

Casas-Garriga, G. 2005. Summarizing sequential data with closed partial orders. In *SDM*, 380–391.

Guttman, A. 1984. R-trees: A dynamic index structure for spatial searching. In *SIGMOD*, 47–57.

Kossmann, D.; Ramsak, F.; and Rost, S. 2002. Shooting stars in the sky: An online algorithm for skyline queries. In *VLDB*, 275–286.

Kung, H.; Luccio, F.; and Preparata, F. 1975. On finding the maxima of a set of vectors. *JACM* 22(4).

Kushmerick, N.; Weld, D. S.; and Doorenbos, R. B. 1997. Wrapper induction for information extraction. In *IJCAI*, 729–737.

Lamport, L. 1978. Time, clocks, and the ordering of events in a distributed system. *Communications of the ACM* 21(7):558–564.

Li, C.; Ooi, B. C.; Tung, A. K.; and Wang, S. 2006. Dada: A data cube for dominant relationship analysis. In *SIGMOD*.

Mannila, H., and Meek, C. 2000. Global partial orders from sequential data. In *KDD*, 161–168.

Papadias, D.; Tao, Y.; Fu, G.; and Seeger, B. 2003. An optimal and progressive algorithm for skyline queries. In *SIGMOD*, 467–478.

Pei, J.; Han, J.; Mortazavi-Asl, B.; and Pinto, H. 2001. Prefixspan:mining sequential patterns efficiently by prefix-projected pattern growth. In *ICDE*, 215–224.

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

Preparata, F. P., and Shamos, M. I. 1985. *Computational Geometry: An Introduction*. Springer-Verlag.

Yang, Z.; Wang, B.; and Kitsuregawa, M. 2007. General dominant relationship analysis based on partial order models. In *SAC*.

Yang, G. 2004. The complexity of mining maximal frequent itemsets and maximal frequent patterns. In *KDD*, 344–353.

Yee, K.-P.; Swearingen, K.; Li, K.; and Hearst, M. 2003. Faceted metadata for image search and browsing. In *CHI*, 401–408.

Yuan, Y.; Lin, X.; Liu, Q.; Wang, W.; Yu, J. X.; and Zhang, Q. 2005. Efficient computation of the skyline cube. In *VLDB*, 241–252.