Efficient Querying Relaxed Dominant Relationship between Product Items based on Rank Aggregation

Zhenglu Yang Lin Li Masaru Kitsuregawa
Info. and Comm. Engineering Department
The University of Tokyo
Tokyo, Japan
{yangzl, lilin, kitsure}@tkl.iis.u-tokyo.ac.jp

Abstract

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, we propose a new model, Relaxed Dominant Relationship (RDR), which extends the state-of-the-art work by incorporating rank aggregation methods. We propose efficient strategies on building compressed data structure to encode the core part of RDR between items. Efficient querying approaches are devised to facilitate the ranking process and to answer the RDR query. Extensive experiments are conducted and the results illustrate the effectiveness and efficiency of our methods.

Introduction

Many e-Business applications, i.e., online shopping system, support users by providing optimal solutions. The typical features of these applications include: (1) the query from users is based on multiple criteria; and (2) different users may prefer different answers based on their personal interests and hence, no single optimal one exists. A practical system should provide all interesting answers that may satisfy a user's demand. One reasonable solution is that only a few "best" goods are recommended by the website's system to the users with regard to their preferences. Here we give an example. Suppose you are looking for one digital camera that is cheap and with light weight on a website. Fig. 1 (a) shows the information of several sample products in 2-dimensional space, in which the items a and b should be the candidates recommended to you because all the other items are worse than these two items with regard the two attributes, price and weight.

The set of these "best" items is called the Pareto set, admissible points (Barndorff-Nielsen & Sobel 1966), maximal vectors (Bentley *et al.* 1978), or skyline points (Borzsonyi, Kossmann, & Stocker 2001). However, as pointed in (Yang *et al.* 2007) all of these works concerned only the pure binary relationship, i.e., a product item p is whether or not worse than (dominated by) others. Interestingly, Yang et al. (Yang *et al.* 2007) proposed an efficient data structure (i.e., $ParCube$) to analyze a more general dominant relationship,

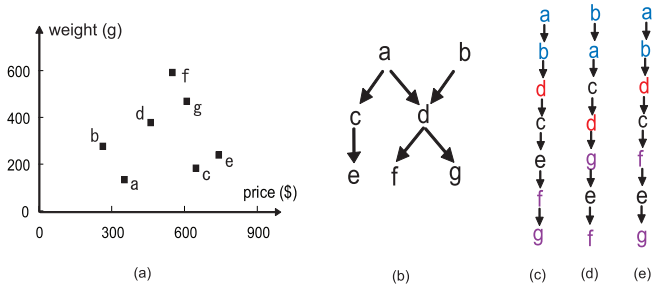


Figure 1: Product attributes in 2-dimensional space, corresponding partial order, and possible fused rank lists

that users preferred the details of the dominant relationship, i.e., an item p is better than (dominates) how many other items. The general dominant relationship between items can be compressed into partial orders. For example, Fig. 1 (b) is the corresponding partial order (encoded as DAG format) of the product items in Fig. 1 (a). We can easily know any product's dominating items by traverse its out-link in the DAG (i.e., c dominates e).

Although the general dominant relationship analysis plays an important role in business decisions, it may fails due to the market adjustment mechanism, that product items are always complementary to one another, i.e., lightweight cameras tend to be more expensive. Therefore, it may be that no products are worse than (dominated by) a given query item, especially when querying in high dimension (attribute) space. As such, users would like to get the items with highest comprehensive scores by fusing the values on different attributes. For example, Fig. 1 (c) shows one fused rank list of the sample dataset in Fig. 1 (a)[1]. Consider you are a manager of a digital camera corporation. You may want to know the business position of a digital camera c in the current market. By checking Fig. 1 (c), you can know that c dominates e, g and f. Note that the result here is different from that discovered by general dominant relationship analysis (i.e., c only dominates e). In this paper, we relax the strict meaning of "dominate" in general dominant relationship analysis by considering on the comprehensive value of items, which incorporates rank aggregation methods.

[1] $Borda\ Count$ rank aggregation method (Borda 1781) is employed here.

We find the relaxed dominant relationship analysis can be efficiently explored based on the partial order. The intuitive idea is that the "strong" ranking in general dominant relationship can be utilized to induce the "weak" ranking in relaxed dominant relationship. With the help of partial order representation (DAG), we can prune some candidate items earlier and do not need to compute their comprehensive value. To illustrate the idea, here we show a simple example. Fig. 1 (c)-(e) shows three possible fused rank lists of the sample product items after applying some rank aggregation method with users' preferences. We can observe that no matter which rank aggregation method is employed, the relaxed dominant relationship between some items do not change. For instance, the item d always dominates f, g, and is dominated by a and b. The stable property between these items can be deduced from the partial order representation (DAG in Fig. 1 (b)) based on the out-link and in-link of d. In this paper we formally justify this discovery and moreover, explore how to efficiently index and query such succinct representative partial orders. For the purposes of this paper, we assume the attribute sets of products are available in structured format.

Our contributions in this paper are as follows:

- We propose a new problem, Relaxed Dominant Relationship Query (RDRQ), which extends the work in (Yang *et al.* 2007) by incorporating rank aggregation. RDRQ is based on a more natural model to rank relational data for business analysis.
- We propose efficient methods to improve the performance of constructing the data cube, $ParCube$(Yang *et al.* 2007), which concisely represents the dominant relationship as partial order representation (DAGs).
- We introduce efficient query processing strategies, which is indeed rank-based fusion method, to answer the general dominant relationship queries with rank aggregation.
- We conduct comprehensive experiments to confirm the efficiency of our strategies.

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 data index and then query it to analyze relaxed dominant relationship. The performance analysis is reported in Section 5. We conclude the paper in Section 6.

Related work

Rank Aggregation. Relaxed Dominant Relationship Analysis is related to the problem of merging rank-ordered lists. There are mainly two kinds of strategies in merging rank-ordered lists. One is score-based, that the score is come from the original score of an item in the rank list, or some transformation of this score (Aslam & Montague 2003; Fagin & Wimmers 1997; Lee 1997). The other strategy is rank-based, that the rank is the original rank of an item assigned to the rank list, or some transformation of this rank (Aslam & Montague 2003; Dwork *et al.* 2001). As another category to classify different approaches, some rank fusion methods rely on training data (e.g., the Bayes-fuse method (Aslam & Montague 2003), the preference rank combination method (Fagin & Wimmers 1997) and the probabilistic model based method (Lillis *et al.* 2006)), while others not.

Dominant relationship analysis with rank aggregation can be seen as a special case, that traditional rank aggregation methods give the order of whole datasets, in contrast, after combining dominant relationship analysis, the relaxed dominant relationship analysis issue becomes finding the order of those items ranked lower than a given item. The two issues coincide when the given query item is the one with highest score (rank) in the dataset.

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) and the $skyline$ query (Borzsonyi, Kossmann, & Stocker 2001). All these works concerned only the pure dominant relationship and, output those items which are not "dominated" by others. In contrast, Yang et al. (Yang *et al.* 2007) proposed to analyze a general dominant relationship from a microeconomic aspect in dynamic environments. The authors devised an efficient data structure, $ParCube$, to compress the general dominant relationship.

However, all these works gave strict definition on "dominate" between two items that, i.e., one item a dominates another item b, if and only if all the attribute values (inferred by users) of a is not worse than those of b. This criteria is too strict and leads to such a result that, it is most likely no items have dominating items, due to the market mechanism of automatic adjustment. Hence, in this paper, we aim to study a ranking-relaxed problem, by incorporating rank aggregation.

Preliminaries

We first introduce some basic notations to present the relaxed dominant relationship analysis in a uniform way. 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, and are ranked higher. A dimension with ranked items will be called a $rank\ list$, denoted as τ.

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.

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

The partial order (D,$\preceq$) can be represented by a DAG $G = (D, E)$, where $(\upsilon, \mu) \in E$ if $\mu \preceq \upsilon$ and there does not exist another value $x \in D$ such that $\mu \preceq x \preceq \upsilon$. For

Table 1: Sample Product Items Dataset

	a	b	c	d	e	f	g
D_1	2	1	6	3	7	4	5
D_2	1	4	2	5	3	7	6
D_3	3	1	4	2	5	6	7

simplicity and without loss of generality, we assume that G is a single connected component.

To relax the strict meaning of $dominate$ in Definition 1, we need to compute each product's comprehensive value by fusing product values (scores) on multiple dimensions (rank lists) into one rank list. Since the values in different dimensions may be not comparable, normalization is usually applied before merging dimensions (rank lists) in order to uniform value distributions to capture within an unique framework. We can apply any normalization method here[2]. Because $Borda\ Count$ (Borda 1781) is one of the seminal rank aggregation methods, we employ its normalized weight. Consider a set S of rank lists (dimensions), a set D of product items, and let a rank list $\tau \in S$. $\tau(p_i)$ is the rank of product p_i in τ.

Definition 2 (normalized weight (Borda rank)) *The normalized weight, $\omega^\tau(p_i)$, of a product $p_i \in \tau$ is defined as follows:*

$$\omega^\tau(p_i) = \begin{cases} 1 - \frac{\tau(p_i)-1}{|D|} & p_i \in \tau \\ \frac{1}{2} + \frac{|\tau|-1}{2\cdot|D|} & otherwise \end{cases} \quad (1)$$

Definition 3 (Borda score) *The fused rank list $\hat{\tau}$ is ordered with regard to the $Borda\ score\ s^{\hat{\tau}}$, where the $Borda\ score$ of an item $p_i \in D$ in $\hat{\tau}$ is defined as*

$$s^{\hat{\tau}(p_i)} = \sum_{\tau \in S} \omega^\tau(p_i) \quad (2)$$

Definition 4 (relaxed-dominate) *A product p is said to relaxed-dominate another product q on S if and only if the $Borda\ score$ of p is larger than the $Borda\ score$ of q.*

Definition 5 (relaxed-dominating set, RDS(p, D, S')) *Given a product p, we use RDS(p, D, S') to denote the set of products from D which are relaxed-dominated by p in the subspace S' of S.*

The problem that we want to solve is as follows:

Problem 1 (Relaxed Dominant Relationship Query (RDRQ)) *Given a dataset D, dimension space S', and a product p, find RDS(p, D, S').*

Example 1 *Consider the 3-dimensional dataset D = {a, b, c, d, e, f, g} in Table 1*[3]*. Given a query point c, dimension space S'={D_1, D_2}, the relaxed-dominating set RDS(c, D, S') = {e, f, g}. We will use this dataset as a running example hereafter.*

[2]The aim of this paper is to incorporate rank aggregation mechanism, rather than comparing different rank aggregation methods.

[3]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: Finding (k+1)-length frequent patterns with optimization

INPUT:	DB = the converted sequence DB
OUTPUT:	$FreMaxPatterns$ = frequent maximal sequential patterns
Function:	$Gen\_Pattern(S)$
Parameters:	S = Set of k-length frequent patterns
Goal:	Generate (k+1)-length frequent sequential pattern
Main():	
1.	F_2 = Scan DB to find 2-length sequential patterns;
2.	Call $Gen\_Pattern$ (F_2);
3.	$FreMaxPatterns$ = Merge all the atoms in F_i;
Function:	$Gen\_Pattern(S)$
4.	For all atoms $A_i \in F_2$
5.	$T_i = \emptyset$;
6.	For all atoms $A_j \in F_2$, with $j \geq i$
7.	$R = A_i \vee A_j$;
8.	$T_i = T_i \cup R$;
9.	$F_{\|R\|} = F_{\|R\|} \cup R$;
10.	For all $T_i \neq \emptyset$
11.	Call $Gen\_Pattern$ (T_i);

As illustrated in the Introduction section, with the help of partial order representation, the Relaxed Dominant Relationship can be extracted efficiently. In the next sections, we will first propose several optimized strategies on discovering partial orders (DAGs), and then give our effective algorithm to tackle RDRQ problem.

Optimization of $ParCube$ Construction

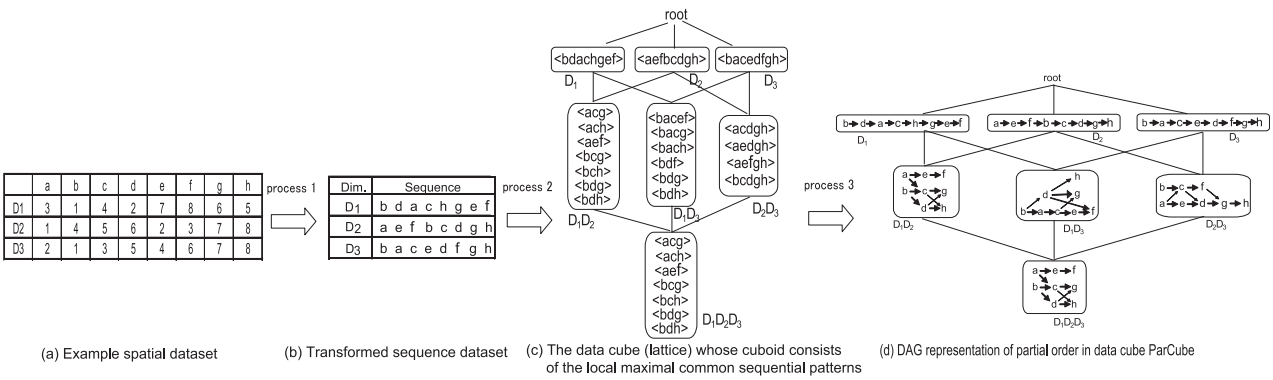
In this section, we will propose our strategies on optimizing the partial order data cube ($ParCube$) construction. We first introduce the methods proposed in (Yang *et al.* 2007) and then present our optimized approaches.

Naive $ParCube$ Building

To get partial orders from spatial datasets, Yang et al. (Yang *et al.* 2007) proposed to apply strategies from another research context, sequential pattern mining (Agrawal & Srikant 1995). There are three processes for $ParCube$ construction, as illustrated in Fig. 2.

The first process is to convert the spatial dataset to the sequence dataset. With a k-dimensional dataset, it is easy to get a k-customer sequence dataset by sorting the objects in each customer (dimension) according to their value in ascending order. In process 2, the sequential patterns from the transformed sequence dataset are discovered by applying PrefixSpan algorithm (Pei *et al.* 2001), which is the state-of-the-art method. The patterns are then merged as $local\ maximal\ sequential\ sequences$, which are not the subsequence of other sequential sequences. The resultant data cube ($SeqCube$) from process 2 for the sample dataset is shown in Fig. 2 (c). In process 3, 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. 2 (d).

Optimization of Sequential Pattern Mining Among the three processes of partial orders finding as illustrated in Fig. 2, the second one, sequential pattern mining, is the slowest



(a) Example spatial dataset

	a	b	c	d	e	f	g	h
D1	3	1	4	2	7	8	6	5
D2	1	4	5	6	2	3	7	8
D3	2	1	3	5	4	6	7	8

(b) Transformed sequence dataset

Dim.	Sequence
D1	b d a c h g e f
D2	a e f b c d g h
D3	b a c e d f g h

(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 2: The result representation of each process for the example spatial dataset

process although the state-of-the-art algorithm is used. To improve the efficiency of the whole work, we aim to develop a new algorithm to fasten the mining process by considering the special property of the converted sequence datasets.

We find that the converted sequence dataset has one important characteristic: for each customer sequence (dimension), one item appears and only appears once. In other words, there is no two same items existing in the same customer sequence (dimension). This is very different from general sequence, i.e., Web log sequence, customer shopping history or DNA sequence. Based on this discovery, we have the following two lemmas:

Lemma 1 (Transitivity) *Let AB and BC be two sequential patterns in k-customer sequences, then AC should also be a frequent sequence in the k-customer sequences.*

Lemma 2 (Pattern Growth) *Let AB and BC be two sequential patterns in k-customer sequences, then ABC should also be a frequent sequence in the k-customer sequences.*

These two lemmas can be easily proofed and we avoid detail here. Based on Lemma 1 and Lemma 2, we can develop a much more efficient algorithm to find the sequential patterns. The pseudo code of the algorithm is shown in Table 2.

Because every item (point) must exist in each customer sequence (dimension), we do not need to find 1-length patterns. In line 1, we thus directly find the 2-length sequential patterns. We scan each item's suffix database to accumulate the support of 2-length candidate sequences. Then in line 2, we recursively call the function $Gen\_Pattern$ to get those patterns whose length are larger than 2. We just merge two atoms together based on their prefix sequences. For example, when merging two patterns, i.e., ab and ac, we need to check the existence of bc or cb in the frequent pattern list. The pattern abc could be claimed if bc is found. By this way, we do not need to do DB projecting and scanning operation in PrefixSpan, which largely reduce the computation cost. The experimental results illustrates the improvement of this strategy. In line 3, we merge these sequential patterns to get maximal ones.

Compression of the $ParCube$ Data Cube The local maximal sequential sequences compress the data to some extent, we can further improve the compression by employing the technique of closed sequence (Yan, Han, & Afshar 2003; Casas-Garriga 2005). If a local maximal sequence l exists in two subspaces, S_1 and S_2 where $S_1 \subset S_2$, then l is only recorded in S_2. For instance, although a sequence, $b \rightarrow d \rightarrow g$, exists in three subspaces $\{D_1,D_2\}$, $\{D_2,D_3\}$ and $\{D_1,D_2,D_3\}$, we only record it in the super-subspace, i.e., $\{D_1,D_2,D_3\}$. This technique can largely reduce the space consumed to store the DAGs. While querying the relaxed dominant in the local subspace, we should also check the super-subspace because some local sequences are absorbed by their super-subspace ones.

Efficient Strategies on Relaxed Dominant Relationship Query

We assume that the query point P_{query} is in the dataset D. With the help of the partial order models, DAGs, we have the following lemmas:

Lemma 3 *Let A be a node on the path from the root to the query point, P_{query}, in the DAG, then P_{query} cannot relaxed-dominate A, no matter which rank aggregation method is used.*

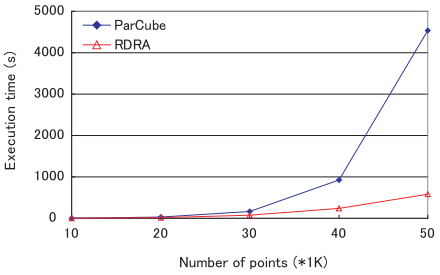
Lemma 4 *Let A be a node on the path from the query point, P_{query}, to any leaf node in the DAG, then P_{query} relaxed-dominates A, no matter which rank aggregation method is used.*

Lemma 5 *If the aggregate score of a node A is smaller than the aggregation score of the query point, P_{query}, with regard to some rank aggregation method (i.e., $Borda\ Count$), then all the child nodes of A in the DAG are relaxed-dominated by A with regard to the same rank aggregation method.*

These three lemmas can be easily proofed and we avoid detail here. The intuition idea is that the general dominant relationship has a stronger rank meaning compared with relaxed dominant relationship. Therefore the semantic meaning compressed in the partial order representation, DAG, can be utilized to deduce relaxed dominant relationship with the help of rank aggregation methods. We next introduce our efficient algorithm based on Lemma 3, Lemma 4 and Lemma 5. Notice that $Borda\ score$ is counted as the aggregation

Table 3: Relaxed Dominant Relationship Query Processing

INPUT:	Partial order representation DAG, a query point P_{query}, a subspace S'
OUTPUT:	RDRQ(P_{query}, D, S')
1.	Insert all points with sorting into the candidate list, L_c, based on their level value in DAG (from top to bottom)
2.	for each parent node c_p of P_{query} in DAG and its subspace S'
3.	remove c_p from the candidate list L_c
4.	for each child node c_p of P_{query} in DAG and its subspace S'
5.	put c_i in the result set RDRQ(P_{query}, D, S')
6.	remove c_p from the candidate list L_c
7.	compute $Borda$ score of P_{query}, s_q (based on Equation 2)
8.	for each candidate point c_p in the candidate list L_c (start from the root node of L_c)
9.	compute $Borda$ score of c_p, s_c (based on Equation 2)
10.	if ($s_c < s_q$)
11.	put c_p in the result set RDRQ(P_{query}, D, S')
12.	for each child node c_n of c_p in DAG with subspace S'
13.	put c_n in the result set RDRQ(P_{query}, D, S')



Execution time on buliding index against number of points(dimensionality=5)

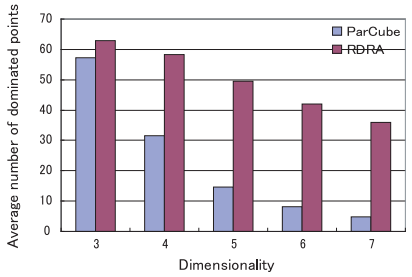
Figure 3: Execution time comparison between $ParCube$ and $RDRA$ on data index construction

score of a point. Other rank aggregation method can be also applied. The pseudo code is shown in Table 3.

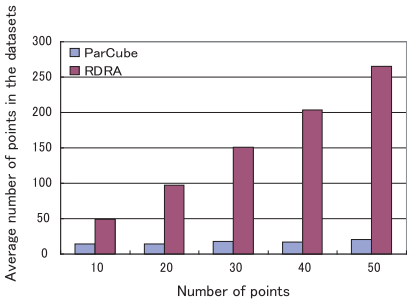
In Line 1, we first insert all points into the candidate list based on their level value in DAG. In fact, this step can be executed in the pre-processing. The reason why we sort the points is based on Lemma 5. We fasten the process if the root of a subgraph is relaxed-dominated by the query point, and hence, all the nodes in the subgraph can be extracted immediately.

Line 2 and Line 3 is based on Lemma 3, which prunes candidates as soon as possible, before we compute their $Borda\ scores$. Lines 4-6 are based on Lemma 4, which extracts the results before we compute their $Borda\ scores$.

From Line 7 to Line 13, we compute and compare the $Borda\ score$ of each candidate point with the query point. This is similar to the traditional rank aggregation process. The difference is that we give the order of the candidate points used to compare, that those located in the higher layer of the DAG will be tested first. By this way, we can enlarge the probability of pruning points earlier. As the experimental results illustrate, which will be introduced shortly, our proposed algorithm largely improves the efficiency of extracting the relaxed-dominant relationship.



(a) Average number of result points when varying dimensionality (number of points=10K)



(b) Average number of result points when varying number of points in the dataset (dimensionality=5)

Figure 4: Querying result comparison between $RDRA$ and $ParCube$ against dimensionality and number of points in the dataset

Performance Analysis

To evaluate the efficiency and effectiveness of our strategies, we conducted extensive experiments. We performed the experiments using a Intel(R) Core(TM) 2 Dual CPU PC (3GHz) with a 3G memory, running Microsoft Windows XP. All the algorithms were written in C++, and compiled in an MS Visual C++ environment. We conducted experiments on both synthetic and real life datasets. However, due to space limitation, we will only report results on synthetic datasets here. Results from real life datasets mirror the results of the synthetic datasets closely. We employ the synthetic data generator (Borzsonyi, Kossmann, & Stocker 2001) to create our synthetic datasets. They have $independent$ distribution, with dimensionality d in the range [3, 7] and data size in the range [10k, 50k]. The default values of dimensionality were 5. The default value of cardinality for each dimension was 50k.

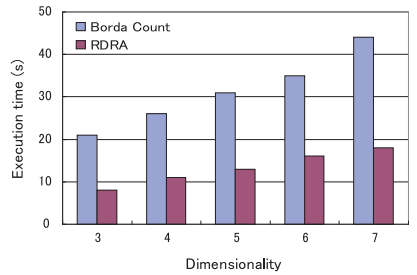
Detailed implementation of the algorithms used to compare is described as follows:

1. *ParCube.* *ParCube* was implemented as described in (Yang *et al.* 2007).
2. *RDRA*[4]. *RDRA* was implemented as described in this paper.

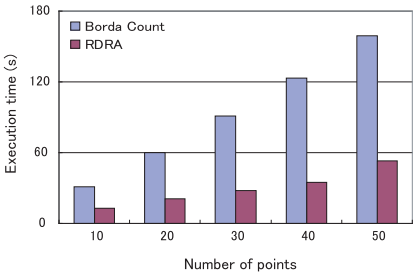
Index Data Structure Construction Performance

In this section, we show the comparison between $RDRA$ and $ParCube$ on building the partial orders (DAGs). Fig. 3 illustrates the execution time for index building against the

[4]Relaxed Dominant Relationship Analysis.



(a) Execution time comparison when varying dimensionality (number of points=10K)



(b) Execution time comparison when varying number of points in the dataset (dimensionality=5)

Figure 5: Execution time comparison of querying between $RDRA$ and $Borda\ Count$ against dimensionality and number of points in the dataset (number of query points=500)

number of points in the dataset. We can see that the $RDRA$ is much faster than $ParCube$. The reason why $RDRA$ performs so well, is because we can prune many candidates earlier and avoid to compare the positions of every item.

Query Performance

In this section, we evaluated the query answering performance of $RDRA$ compared with $ParCube$. Note that in this paper, our major purpose is to provide more natural candidate items that users may favor, rather than compare the precision of results between different rank aggregation methods. To test the effect of RDR query, we randomly selected 500 different points from D and the result is the average value. For $RDRA$, given the randomly selected point p, we queried p's relaxed dominating points. For $ParCube$, we queried p's general dominating points.

The result is shown in Fig. 4, from where we can see that $RDRA$ always extracts more dominated points than $ParCube$. This is not surprising because we relax the strict meaning of $dominate$ in $ParCube$ and thus, can give users more favorable candidate items. When dimensionality increases, as shown in Fig. 4 (a), the size of the result set in $ParCube$ decreases quickly, since $ParCube$ is sensitive to the dimensionality. In contrast, $RDRA$ is relative stable on output result set. When changing the number of points in the dataset, the result set of $RDRA$ proportionally varies. However, the result set of $ParCube$ keeps stable. In summary, compared with $ParCube$, $RDRA$ outputs more reasonable candidates items.

The comparison of the execution time on querying relaxed dominant relationship between $RDRA$ and traditional rank aggregation ($Borda\ Count$) is shown in Fig. 5. The reason why we compared with $Borda\ Count$ is that we want to demonstrate the efficiency of partial orders on querying relaxed dominant relationship, rather than comparing two rank aggregation methods themselves. The latter issue is beyond the scope of this paper. We can know that $RDRA$ is much efficient than its competitor for the two cases (varying dimensionality and number of points) because of the effect of partial orders we used.

Conclusions

We have introduced Relaxed Dominant Relationship Query ($RDRQ$), which is an extension model based on general dominant relationship by incorporating rank aggregation. We found that $RDRQ$ can provide more natural candidates that users may favor. We have proposed efficient strategies to build partial order models and to answer $RDRQ$. The performance study confirmed the efficiency of our strategies.

References

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

Aslam, J. A., and Montague, M. 2003. Models for metasearch. In *SIGIR*, 276–284.

Barndorff-Nielsen, O., and Sobel, M. 1966. On the distribution of the number of admissable points in a vector random sample. *Theory of Probability and its Application* 11(2):249–269.

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.

Borda, J. C. 1781. Mémoire sur les élections au scrutin. In *Histoire del'Académie Royal des Sciences*.

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.

Dwork, C.; Kumar, R.; Noar, M.; and Sivakumar, D. 2001. Rank aggregation methods for the web. In *WWW*, 613–622.

Fagin, R., and Wimmers, E. L. 1997. Incorporating user preferences in multimedia queries. In *ICDT*, 247–261.

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

Lee, J. H. 1997. Analysis of multiple evidence combination. In *SIGIR*, 267–276.

Lillis, D.; Toolan, F.; Collier, R.; and Dunnion, J. 2006. Probfuse: a probabilistic approach to data fusion. In *SIGIR*, 139–146.

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.

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

Yan, X.; Han, J.; and Afshar, R. 2003. Clospan: mining closed sequential patterns in large datasets. In *SDM*, 166–177.

Yang, Z.; Li, L.; Wang, B.; and Kitsuregawa, M. 2007. Towards efficient dominant relationship exploration of the product items on the web. In *AAAI*, 1483–1488.