Fragments of order

Aristides Gionis*
Dept. of Computer Science
Stanford University
Stanford, CA, 94305, USA
gionis@cs.stanford.edu

Teija Kujala
HIIT Basic Research Unit
Dept. of Computer Science
University of Helsinki
P.O. Box 26, Teollisuuskatu 23
FIN-00014 Helsinki, Finland
Teija.Kujala@cs.helsinki.fi

Heikki Mannila
HIIT Basic Research Unit
Dept. of Computer Science
University of Helsinki
P.O. Box 26, Teollisuuskatu 23
FIN-00014 Helsinki, Finland
Heikki.Mannila@cs.helsinki.fi

ABSTRACT

High-dimensional collections of 0-1 data occur in many applications. The attributes in such data sets are typically considered to be unordered. However, in many cases there is a natural total or partial order $\prec$ underlying the variables of the data set. Examples of variables for which such orders exist include terms in documents, courses in enrollment data, and paleontological sites in fossil data collections. The observations in such applications are flat, unordered sets; however, the data sets respect the underlying ordering of the variables. By this we mean that if $A \prec B \prec C$ are three variables respecting the underlying ordering $\prec$, and both of variables A and C appear in an observation, then, up to noise levels, variable B also appears in this observation. Similarly, if $A_1 \prec A_2 \prec \cdots \prec A_{l-1} \prec A_l$ is a longer sequence of variables, we do not expect to see many observations for which there are indices $i < j < k$ such that A_i and A_k occur in the observation but A_j does not.

In this paper we study the problem of discovering fragments of orders of variables implicit in collections of unordered observations. We define measures that capture how well a given order agrees with the observed data. We describe a simple and efficient algorithm for finding all the fragments that satisfy certain conditions. We also discuss the sometimes necessary postprocessing for selecting only the best fragments of order. Also, we relate our method with a sequencing approach that uses a spectral algorithm, and with the consecutive ones problem. We present experimental results on some real data sets (author lists of database papers, exam results data, and paleontological data).

*Supported by a Microsoft Research Fellowship. Part of this work was done while the author was visiting the HIIT Basic Research Unit, Department of Computer Science, University of Helsinki, Finland.

SIGKDD '03, August 24-27, 2003, Washington, DC, USA

Categories and Subject Descriptors

H.2.8 [**Database management**]: Database applications—*Data mining*; F.2.2 [**Analysis of algorithms and problem complexity**]: Nonnumerical algorithms and problems

Keywords

Novel data mining algorithms, discovering hidden orderings, spectral analysis of data, consecutive ones property

1. INTRODUCTION

High-dimensional collections of 0-1 data occur in many applications, such as in market basket analysis, telecommunications networks, information retrieval, computational biology, and ecology. The attributes in such data sets are typically considered to be unordered. However, in many cases there are ordering dependencies among the attributes. Consider, for example, technical articles about database management systems, and the terms "database system", "query", and "selectivity estimation". A document can contain one of them, all of them, or just the first two or last two. However, a document containing the terms "database" and "selectivity estimation" but not "query optimization" seems somewhat strange.

Indeed, searches on Citeseer[1] using Google give the following numbers of hits:

database system	query	selectivity estimation	hits
1	1	1	49
1	1	0	1930
0	1	1	221
1	0	1	4

We indeed see that one particular pattern of occurrences of terms is very rarely represented. Note that we are not looking at the ordering of terms in individual documents. For each document, we only look at whether the terms occur in it or not. We aim at using this data to obtain information about the ordering relationships between the attributes (variables).

The reason for the asymmetry between the three terms in the above example is that there is an underlying directionality in the terms. The term "selectivity estimation" in the context of "database systems" is dependent on the

[1]http://citeseer.nj.nec.com

concept of "query": selectivity estimation cannot be discussed without mentioning (sooner or later) queries. The chain of concepts could be longer: for example, "histogram technique" would probably fit to the end of the above chain.

What makes the above ordering

"database system" $\prec$ "query" $\prec$ "selectivity estimation"

interesting is that there are enough observations that contain at least two of the terms (the fragment has high frequency) and few observations have the first and last but do not have the second one (the fragment has few violations). We make these criteria specific in the next section.

In this paper we study the problem of discovering this type of fragments of order from unordered 0-1 data. Such orderings between smaller or larger sets of variables occur in various types of data sets. For example, consider a data set where the variables represent courses at a department and rows represent students; a 1 indicates that the student has passed the course. The prerequisites of courses show up in fragments of order: if course A_i is needed for course A_{i+1} for $i = 1, \ldots, l-1$, we assume that no student has passed two courses A_i and A_k but not A_j for some $i < j < k$. Some examples of observations conforming and violating the order $A \prec B \prec C \prec D$ are given in the next table.

A	B	C	D	ok?
1	1	1	0	yes
0	1	1	0	yes
0	1	1	1	yes
1	0	1	1	no
1	0	0	1	no
1	0	1	0	no
0	1	0	1	no

Note that if an observation violates some order, it also violates the order obtained by reversing the direction of all the precedence relationships.

As a third example, consider paleontological data about fossil remains of animals. The variables are different sites (in different locations) from which fossils have been found, and the observations represent the taxa (species or genera). If variable A is set for observation t, this means that taxon t has been found at site A. Each site represents fossils from a relatively brief interval in geological time. The goal for this type of data is to find the order corresponding to the ages of the sites. For a fragment of order $A \prec B \prec C$ relating three sites, an observation t that violates this order corresponds to the so-called Lazarus phenomenon: a taxon that is extant at A, extinct for B, and reappears at C.

The rest of this paper is organized as follows. In Section 2 we review work related with the problem we consider here. In Section 3 we describe the formal definitions of fragments of order and give the problem statement for finding all fragments from a dataset. We focus on finding fragments of order, and not total orders, as the real-life data sets seldom allow for nice orderings of attributes that would be compatible with most of the data. Section 4 describes a simple A priori-like algorithm for finding the fragments, and it shows how we can do some postprocessing to select the best fragments. Section 5 is a quick introduction to spectral clustering, a method that can be used to yield total orderings of all the attributes minimizing some stretch measures. Section 6 gives the empirical results, and Section 7 is a short conclusion.

2. RELATED WORK

We are not aware of a lot of related work. Our work is of course heavily influenced by the work on association rules [1]: the idea of looking at *all* possible patterns from a concept class that satisfy certain frequency counts has proved to be very useful (e.g., sequential patterns [2, 15]). Here, however, the concepts are slightly less straightforward: an observation with all 1s does not contribute to all patterns, as it does for normal frequent pattern type of concepts.

Some of the approaches of postprocessing association rules (see, e.g., [10, 19]) aim at finding graphs of attributes from 0-1 data. However, the semantics there typically are quite different from ours. Mannila and Meek [14] provide an algorithm for finding a partial order for the full set of variables in an dataset consisting of observation in each of which the variables are *ordered*; in our setting, the observations are sets, not sequences. The work of Popescul et al. [17] has somewhat similar goals to ours, but in their paper they consider the context of linked documents. In computational biology, e.g., in sequence assembly (see [8]), one encounters the task of totally ordering a set of variables so that so sort of breaks are minimized. This is fairly closely related to the method of spectral clustering. Again, our emphasis is on finding fragments of order, not a single best order, and hence many of the techniques from computational biology are not directly applicable. Our problem is also related to the consecutive ones property [4]; we will discuss the relationship in Section 5.

3. PROBLEM STATEMENT

In this section we introduce some notation and define more formally the problem of discovering the fragments of order. Let $\mathcal{R} = \{A_1, \ldots, A_n\}$ be the set of 0-1 attributes of the dataset. A dataset $\mathcal{D} = \{r_1, \ldots, r_m\}$ consists of m observations each of which is a subset of attributes from $\mathcal{R}$. Equivalently, an observation can be viewed as a record with n attributes $\{A_1, \ldots, A_n\}$ taking values in $\{0, 1\}$. This gives rise to the common data representation as a 0/1 matrix with m rows corresponding to observations and n columns corresponding to attributes.

We are interested in cases where there is an underlying *ordering* of a subset of the attributes of $\mathcal{R}$. We denote such an ordering using the symbol $\prec$, and we write $A_i \prec A_j$ to say that A_i *precedes* A_j in the ordering. A *sequence* of attributes $F = \langle A_1, \ldots, A_l \rangle$ defines a *fragment of order*, or just a *fragment*, if the attributes of F form a *chain* with respect to the ordering $\prec$, i.e., if it is the case that $A_1 \prec \ldots \prec A_l$. We say that fragment F is of *length* l. We sometimes write $F = \langle A_1, \ldots, A_l \rangle$ and in other cases we use the notation $A_1 \prec \cdots \prec A_l$.

Orderings on the data attributes like the above can be implicitly induced by a hidden variable associated with the attributes, or by a semantic interpretation of the attributes.

The central observation in our paper is that there is enough information in the data to recover the underlying ordering of the attributes, or at least to obtain some clues regarding this ordering. This is accomplished by noticing that, given an ordering, certain observations respect it and some violate it.

As an example, consider a potential fragment $F = \langle A_i, A_j, A_k \rangle$ of length 3, i.e., the ordering $A_i \prec A_j \prec A_k$. We say that an observation t *violates* F if $t[A_i] = t[A_k] = 1$ and $t[A_j] = 0$.

An observation t violates a potential fragment F of length $l > 3$, if it violates any fragment F' of length exactly 3 which is *subsequence* of F, i.e., there are three attributes A_i, A_j, and A_k in the ordering such that

$$A_i \prec \cdots \prec A_j \prec \cdots \prec A_k$$

and we have $t[A_i] = t[A_k] = 1$ and $t[A_j] = 0$. We say that an observation *respects* a potential fragment if it does not violate it.

Given a fragment $F = \langle A_1, \ldots, A_l \rangle$, with $l \geq 2$, the *frequency* $f(F, \mathcal{D})$ of F on dataset $\mathcal{D}$ is the fraction of rows r in $\mathcal{D}$ in which at least two attributes of F appear, i.e., for some i and j with $i \neq j$ we have $r[A_i] = r[A_j] = 1$. We denote by $\mathcal{R}(F, \mathcal{D})$ the collection of rows that have the above property; thus $f(F, \mathcal{D}) = |\mathcal{R}(F, \mathcal{D})|$. If fragment F' is a permutation of F, then F and F' have the same frequency. The *violation fraction* $v(F, \mathcal{D})$ of F on $\mathcal{D}$ is the fraction of rows r of $\mathcal{R}(F, \mathcal{D})$ that violate F. For a fragment F of length 2 we define $v(F, \mathcal{D}) = 0$ for all $\mathcal{D}$. Note that the frequency of F is the fraction from all rows, while the violation fraction is the fraction of violating rows from the $\mathcal{R}(F, \mathcal{D})$ rows contributing to $f(F, \mathcal{D})$.

We want to search for orders that are good in the sense that they have few violations. This condition, however, is not sufficient for the ordering to be interesting. A database in which all observations are empty (i.e., there are no 1s) has violation fraction of 0 for all orderings. This, obviously, is not what we want: we want to also have positive evidence that the attributes occurring in the order are used.

As an example, consider the dataset

A	B	C	D
1	1	0	0
0	1	1	0
0	0	1	1

All the rows in this table contribute to the frequency of the ordering $A \prec B \prec C \prec D$, and this ordering has no violations The same is true for the reverse ordering $D \prec C \prec B \prec A$. All the other 22 orderings of the four attributes have also frequency 1, but each has at least one violation.

A frequency threshold σ is used in combination with the violation threshold τ in order to obtain fragments that appear frequently in the dataset and have a small number of violations. In particular, given thresholds $\sigma, \tau \in [0, 1]$ we define the set $\mathcal{T}_0(D, \sigma, \tau)$ to consist of the fragments that are σ-frequent and not τ-violated, i.e.,

$$\mathcal{T}_0(\mathcal{D}, \sigma, \tau) = \{F \text{ fragment of } \mathcal{R} \mid f(F, \mathcal{D}) \geq \sigma,\ v(F, \mathcal{D}) \leq \tau\}.$$

Note, however, that this definition does not require that all attributes contribute to the frequency of the fragment. Consider the dataset

A	B	C	D
1	1	1	0
1	1	1	0
1	1	1	0

The frequency of the ordering $A \prec B \prec C \prec D$ is 1 and there are no violations. Obviously, D is not contributing to the order. Thus we augment our definition by requiring that all subfragments of F also have to belong to the collection:

$$\begin{aligned} \mathcal{T}(\mathcal{D}, \sigma, \tau) \quad = \quad & \{F \mid f(F, \mathcal{D}) \geq \sigma,\ v(F, \mathcal{D}) \leq \tau, \\ & \text{all subfragments } F' \text{ of } F \text{ of size } \geq 2 \\ & \text{satisfy } f(F', \mathcal{D}) \geq \sigma,\ v(F', \mathcal{D}) \leq \tau\}. \end{aligned}$$

According to this definition, the fragment $A \prec B \prec C \prec D$ does not belong to $\mathcal{T}(\mathcal{D}, \sigma, \tau)$ for any $\sigma > 0$, as the frequency of, e.g., fragment $A \prec D$ is 0.

One can worry whether our criteria for interesting fragments is too weak. Consider the following data set.

A	B	C	D
1	1	1	1
1	1	1	1
1	1	1	1

Every ordering of the four attributes has frequency 1 and violation fraction 0, so they would belong to the set $\mathcal{T}(\mathcal{D}, \sigma, \tau)$ for all choices of σ and τ. In Section 4.2 we show how we select between the orderings among the same set of attributes. Basically, the strategy is to look at the number of violations: if there is an ordering F that has clearly less violations than the other orderings then that implies that F is more informative than other orderings among the same attributes.

In Section 4 we describe an efficient algorithm to compute the set $\mathcal{T}(D, \sigma, \tau)$, and we show how the best orderings can be found in the case where there are several permutations of the same subset of variables in the collection $\mathcal{T}(D, \sigma, \tau)$.

4. DISCOVERING FRAGMENTS OF ORDER

In this section we describe an algorithm for computing the set $\mathcal{T}(\mathcal{D}, \sigma, \tau)$ for given $\mathcal{D}$, σ and τ.

The algorithm consists of two phases. The first is a fairly standard *levelwise* computation of the set $\mathcal{T}(\mathcal{D}, \sigma, \tau)$. This can be done efficiently in fashion similar to A priori algorithm, since the collection $\mathcal{T}(\mathcal{D}, \sigma, \tau)$ is downward closed by definition.

One characteristic of the set $\mathcal{T}(\mathcal{D}, \sigma, \tau)$ is that it might be possible (and indeed this is typically the case) to contain many fragments of the same subset of attributes. First, for any fragment $\langle A_1, \ldots, A_l \rangle$ that belongs in the set $\mathcal{T}(\mathcal{D}, \sigma, \tau)$ it is true that the *reverse* fragment $\langle A_l, \ldots, A_1 \rangle$ belongs in the set, as well. This is true because the two fragments have exactly the same frequency and violation fraction. In other words, the data $\mathcal{D}$ is *oblivious* with respect to the *direction* of attribute orderings.

In addition to reversed fragments, it is possible that many more permutations of the same subset of attributes belong in the set $\mathcal{T}(\mathcal{D}, \sigma, \tau)$. Imagine, for example, a dataset $\mathcal{D}$ for which there is a subset S of attributes such that in each row of $\mathcal{D}$ either all of the attributes of S appear together or none. In this case, it is feasible that all fragments that are derived as permutations of S are members of $\mathcal{T}(\mathcal{D}, \sigma, \tau)$.

This motivates the second phase of our algorithm, which is a *peer selection* phase. We would like to distinguish between fragments that correspond to a true ordering of attributes, versus fragments that appear in $\mathcal{T}(\mathcal{D}, \sigma, \tau)$ just because they are frequent *itemsets*. The intuition of the algorithm for discovering true ordering is to test how a fragment stands out among its *peers*, where peer is any other fragment which is a permutation of the same attributes. This is explained in detail in Section 4.2

4.1 Levelwise phase

As we mentioned in the previous section, the algorithm for computing the set $\mathcal{T}(\mathcal{D}, \sigma, \tau)$ is an A priori type method based on the monotonicity property of the collection $\mathcal{T}(\mathcal{D}, \sigma, \tau)$. Note that the frequency of a fragment is not monotonic with

respect to subsequences: the frequency of $\langle A, B, C, D\rangle$ can be higher than the frequency of $\langle A, B, C\rangle$.

The algorithm starts by forming all fragments consisting of exactly two attributes from $\mathcal{R}$. These are marked as *candidate* fragments of length 2. At the k-th step, candidate fragments of length k have been formed. The algorithm proceeds by dropping all fragments that do not satisfy the frequency and violation fraction thresholds, and by forming candidate fragments of length $k+1$. The inbuilt monotonicity of $\mathcal{T}(\mathcal{D}, \sigma, \tau)$ guarantees that no fragment will be missed.

The next three functions provide a detailed description of the levelwise algorithm.

```
LevelWise(D, σ, τ)
  C_2 = R × R
  ▷ C_2 now contains all ordered pairs
  k = 2
  while C_k ≠ 0
        do L_k = Evaluate(D, σ, τ, C_k)
           C_{k+1} = Candidates(L_k)
           k = k + 1
```

```
Candidates(L)
  ▷ L is a set of equal-length sequences from R
  for X, Y ∈ L
      do if X = Az and Y = zB
            then AzB is a potential candidate
  for all potential candidates W = B_1 ... B_{k+1}
      do for i ← 1 to k + 1
            do S = B_1 ... B_{i-1} B_{i+1} ... B_{k+1}
               ▷ S is subsequence of W of length k
               if S ∈ L
                  then add W to the result
```

```
Evaluate(D, σ, τ, C)
  ▷ C is a collection of sequences over R
  for all F ∈ C
      do compute f(F, D)
         compute v(F, D)
         if f(F, D) ≥ σ and v(F, D) ≤ τ
            then add F to the result
```

The complexity of the algorithm is $O(mn(|\mathcal{T}|+C))$, where m is the number of observations in the data, n is the number of variables, and C is the number of candidate fragments considered which turn out not to belong to $\mathcal{T}$. Thus the method is linear in the size of the data and scales nicely to large data sets.

The frequency and violation fraction for a fragment or a set of fragments can be computed from the data by simple sequential scan. The counts can also be written by using the inclusion-exclusion principle in the form $f(F, \mathcal{D}) = \sum_i c_i fr(X_i, \mathcal{D})$, where the sets X_i are subsets of the fragment, $fr(X_i, \mathcal{D})$ is the frequency of the *set* X_i in the association rule sense [1], and c_i is a positive or negative integer. However, these expressions can be exponentially large in the number of elements in F, so we in our experiments compute the frequencies directly from the data. However, the use of expressions of the above form gives the possibility of taking advantage of the wealth of fast association rules algorithms [10] for our problem, as well.

4.2 Selection among peers

The second phase of the algorithm for discovering the best fragments of order is to test whether a fragment is a true order that stands by its own. As we mentioned before, if a set of attributes occur always together we do not want to take a single or all permutations of this set as a result. In this case we would probably like to say that the trivial partial order is the right answer.

Suppose that a fragment F belongs to $\mathcal{T}(\mathcal{D}, \sigma, \tau)$. This means that the ordering of attributes given by F is not violated too often. We still have to test that the actual order is significantly different from other possible orders of the attributes occurring in F.

If, for example, for three attributes A, B, and C we have that the association rules $AB \to C$, $AC \to B$, and $BC \to A$ all have accuracy 1 in the data, then all 6 permutations of ABC have the same frequency and violation fraction. (Note the similarity to concepts such as closed sets.)

Next we describe a simple probabilistic model for estimating the likelihood of orders (somewhat in the spirit of [14]). The model gives in the end the expected result: the best ordering is the one that has the fewest violations.

Given a fragment $F = \langle A_1, \ldots, A_l\rangle$, consider the set of all possible permutations of the attributes of F, that is

$$\mathcal{S}(F) = \{Z \mid Z \text{ is a permutation of } A_1 \ldots A_l\}$$

For a fragment F, the likelihood $L(\mathcal{D}|F)$ expresses the probability that the dataset $\mathcal{D}$ comes from a model where F is a true order on the attributes. We assume that

$$L(\mathcal{D}|F) \propto exp(-v(F, \mathcal{D})), \tag{1}$$

i.e., the logarithm of the likelihood is proportional to the violation fraction. The reason for this assumption can be obtained, e.g., from MDL type of arguments.

Assume that $L(\mathcal{D}|Z)$ is computed for all $Z \in \mathcal{S}(F)$, and furthermore assume that the fragments in $\mathcal{S}(F)$ is the complete set of models for which we are interested in distinguishing the likelihood of the data. Then, by applying Bayes' rule we can write

$$Pr(F|\mathcal{D}) = \frac{Pr(Z)L(\mathcal{D}|Z)}{Pr(\mathcal{D})} = \frac{Pr(Z)L(\mathcal{D}|Z)}{\sum_{W\in\mathcal{S}(F)} Pr(W)L(\mathcal{D}|W)}$$

We are assuming that $Pr(Z) = \frac{1}{|\mathcal{S}(F)|}$ for all $Z \in \mathcal{S}(F)$, since we have no reason to favor a particular permutation. For the choice of $L(Z|\mathcal{D})$ as specified in Equation (1) we have

$$Pr(Z|\mathcal{D}) = \frac{exp(-v(Z, \mathcal{D}))}{\sum_{W\in\mathcal{S}(F)} exp(-v(W, \mathcal{D}))}$$

One should immediately observe that the denominator in the above formula does not depend on the permutation Z, thus, for selecting the best permutation, we only have to consider the term $exp(-v(Z, \mathcal{D}))$. The log of the latter is proportional to $v(Z, \mathcal{D})$, i.e., the number of violations of fragment Z in the dataset $\mathcal{D}$.

Motivated by the above discussion, we suggest that the selection among peers (say, $\langle ABC\rangle$ and all permutations of it) should be based on the number of violations for each permutation. We consider the number of violations just as a (e.g., binomially distributed) random variable. Given such a set of random variables we select the (one or several) permutations with the less number of violations as candidate

best fragments. To decide if these candidates are indeed best fragments we need to verify that the values of their violation fractions cannot be attributed to randomness and noise in the data. This can be done easily with a standard ANOVA test. The selection phase algorithm is summarized below.

SELECTAMONGPEERS($\mathcal{T}$)
▷ $\mathcal{T}$ is set of fragments computed at the previous phase
▷ We assume that for each $F \in \mathcal{T}$ we know $v(F, \mathcal{D})$
Partition $\mathcal{T}$ into groups of peers $\{P_1, \ldots, P_d\}$
for each group of peers P_i
  do for each fragment $F \in P_i$
    do Compute $Pr(F, \mathcal{D})$ using $v(F, \mathcal{D})$
  S fragments in P_i using ANOVA

We assume that the violation fractions $v(F, \mathcal{D})$ are known since they have been computed in the previous phase of the algorithm. The complexity of the selection phase is $O(|\mathcal{T}| \log |\mathcal{T}|)$. This is because the most expensive step – partitioning $\mathcal{T}$ into groups of peers – can be accomplished by lexicographic sorting of all fragments in $\mathcal{T}$.

5. SPECTRAL ALGORITHMS FOR ORDERING

In this section, we describe an ordering algorithm based on spectral methods. Spectral algorithms are important tools for solving graph partitioning problems, and they have been used in a wide range of applications, such as solving linear systems [18], domain decomposition [6], scientific numerical algorithms [20], and clustering problems [16]. Spectral algorithms have also been used for ordering vertices in a graph, and in particular for the linear arrangement problem, as discussed in [13].

The spectral algorithm we discuss in this section is attractive because of its simplicity and its intuitive appeal. However, one should note that formally the spectral algorithm does not correspond to the same problem that we defined in Section 3. The main difference is that the spectral algorithm provides a single *total* ordering, instead of many, perhaps overlapping, fragments of order. Similarly, the concepts of frequency and violation fraction thresholds are not used by the algorithm. In the rest of the section, we give some background for the spectral method in general and then we describe our spectral ordering algorithm.

Consider an undirected graph $G = (V, E)$ where each $(i, j) \in E$ has weight w_{ij}. Let A be a matrix whose (i, j)-th entry is equal to w_{ij} if $(i, j) \in E$ and 0 otherwise. The *Laplacian* of graph G is defined to be the symmetric and zero-sum matrix $L = D - A$, where D is a diagonal matrix whose (i, i)-th entry is $d_i = \sum_{(i,j)\in E} w_{ij}$. Let v be the eigenvector of L that corresponds to the second smallest eigenvalue. One can show (e.g., [7]) that v is the vector that minimizes the quadratic form

$$\sum_{(i,j)\in E} w_{ij} \cdot (v_i - v_j)^2 \qquad (2)$$

$$\text{subject to: } ||v||_2 = 1$$

Vector v is also known in the literature as the *Fiedler vector*. It can be viewed as a mapping from a node $i \in V$ to the value v_i, that is, as an embedding from graph nodes to the 1-dimensional line. If we use Equation (2) to define a "stretch" energy function associated with this embedding, then the Fiedler vector has the property that it minimizes this energy function.

This has clearly an ordering interpretation. However, it goes beyond simple ordering since it also assigns real number values to nodes and therefore distances between them. If one is looking for an ordering it is more natural to search for a bijection $\pi : V \to \{1, \ldots, n\}$ that minimizes the *arrangement cost*

$$\sum_{(i,j)\in E} w_{ij} \cdot |\pi(i) - \pi(j)|$$

This NP-hard problem is studied in [13] and various heuristics are proposed. In this paper, we use such a heuristic to obtain an approximate total orderings on the dataset attributes. This is done as follows:

We consider each attribute $A \in \mathcal{R}$ to be a node in a graph $G_{\mathcal{D}}$. For each pair of attributes (A_i, A_j), we consider the weight of this edge in $G_{\mathcal{D}}$ to be the frequency of the itemset $\{A_i, A_j\}$. The Laplacian matrix of $G_{\mathcal{D}}$ is formed and the Fiedler vector is computed. Then, the algorithm outputs all the attributes in order of ascending (or descending) value of their Fiedler coordinates.

The spectral algorithms have interesting connections to the *consequtive ones* property. A matrix of 0s and 1s has the consequtive ones property, if there is a way of permuting the attributes so that for each row all the 1s are consequtive. Testing whether a matrix has the consequtive ones property can be done in linear time using PQ-trees [4, 5, 11]. The relationship of spectral methods and the consequtive ones property is studied in more detail in [3].

The relationship between spectral clustering and finding fragments of order is interesting. Consider the case when the input data has the consequtive ones property. Then by the results of [3] the spectral method (used for the Laplacian matrix of $G_{\mathcal{D}}$) produces an ordering which satisfies the consequtive ones property. Furthermore, in this case this ordering has in our terms frequency 1 and no violations. Thus in the case of consequtive ones property the output from our method would be of exponential size, as all subsequences of the order would qualify. Thus one can see our technique as more applicable to situations in which the underlying order does not have long paths.

6. EXPERIMENTS

In this section we briefly describe some of the experiments we have done on real data using the fragment discovery algorithm.

We report some results from a students exam results data set, from author lists of database papers, from titles of database papers, and from paleontological databases.

6.1 Generated data

We also ran experiments on generated data, by generating random orderings of subsets of the set of attributes and generating data by selecting consequtive attributes from these orderings. The data was corrupted by adding noise. The results on these data sets showed that the method was able to find the generating orderings up to a fairly high level of noise, and that the number of superfluous fragments remained small. We omit the details.

σ (in %)	τ (in %)	$Max\ l$	$\|\mathcal{T}\|$	α (in %)	β (in %)
20	0	3	2	96.3	99.5
20	2.5	5	578	48.6	70.5
20	5	6	1528	40.0	66.0
20	10	7	4476	29.3	59.3
15	0	3	28	89.9	98.6
15	2.5	6	1934	46.8	78.2
15	5	7	5158	38.9	72.3
15	10	8	16884	28.0	64.9
10	0	3	222	76.9	97.7
10	2.5	6	6390	45.1	81.1
10	5	8	19716	35.0	73.9
10	10	9	82116	18.7	65.1

Table 1: Results on the exam results data. σ: the frequency threshold; τ: the violation fraction threshold. $Max\ l$ is the maximum length of a fragment in the result. $|\mathcal{T}|$ is the total number of fragments found. α, β: see text.

6.2 Exam results data

We considered a data set from the Department of Computer Science at the University of Helsinki. The data set has 2953 observations (corresponding to students), 5684 variables (corresponding to courses), and 72395 exam results. The average number of 1s per row is 24.5, with a minimum of 1 and maximum of 131.

For this data, we ran experiments on the fragment finding algorithm. In this case, we have two ways of relating the resulting fragments to the data. First, we know the recommended ordering among the courses; see Figure 1 for a partial view of this. Second, we have in the original data set information about the times in which the students have taken each course. So we can compare how well the discovered fragments correspond with official policy and with the real data.

Table 1 gives some results on how the fragments are related to the actual known order in which each student has passed the courses. To compare how well the fragments correspond with the actual order of courses, we computed two coefficients. The score α is computed by looking at each fragment and each observation that contains at least all the variables occurring in that fragment. We test whether the order of the courses in the observations is the same (or reverse) from that in the fragment; α is the fraction of cases in which this happens. The score β is computed by looking at each fragment and each observation containing at least 2 courses appearing in the fragment, and by checking whether the order in the observation corresponds to the order in the fragment (or its reverse); β is the fraction of cases in which this happens. (Note that β has the bias that no observation with at most 2 courses can cause a violation.)

Visual comparison of the discovered fragments against the ordering shown in Figure 1 shows that the correspondence is good, especially for small thresholds of the violation fraction. When up to 40 % of violations are allowed, one sees also fragments that are not compatible with the partial ordering in the figure.

As a single example, we found the fragment

⟨Programming,
Computer Organization,

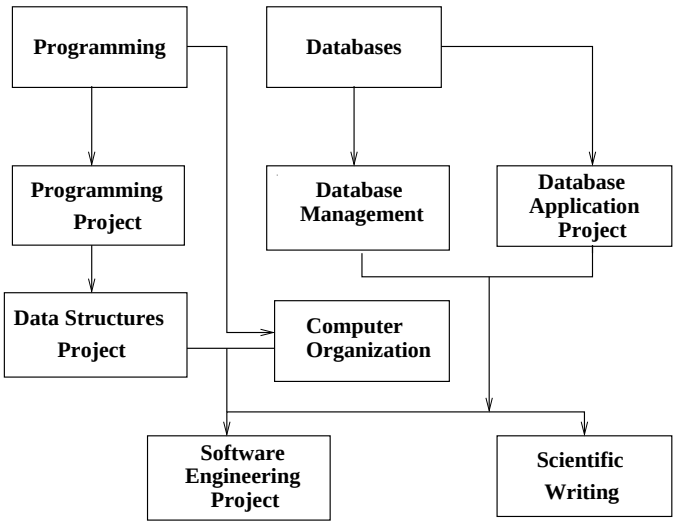

Figure 1: Recommended ordering among 9 courses in the exam results data.

Programming Project,
Data Structures Project,
Scientific Writing⟩

which had frequency 1361, and violation fraction 3.2%. There were 464 students who had attended all these 5 courses, and 278 had passed them in exactly the above order. When the course Computer Organization took away from the above order, the frequency was 1222 and violation fraction 0.9%. The same 464 students who had attended all those 4 courses, but 439 (94.6%) had passed them in exactly the above order.

The results show that the fragment discovery algorithm manages to capture a surprising amount of the order actually existing in the data. In [14] an algorithm was given that reconstructed a partial order for the courses, given the *ordered* observations; what we show here is that the fragments describing the same ordering to a high degree can be reconstructed from *unordered* data.

6.3 Authors of database papers

We studied the bibliographies of VLDB, SIGMOD, and PODS available at the Collection of Computer Science Bibliographies[2], and extracted the lists of authors from each paper. A selection of the papers with at least 2 authors yielded 3109 rows (papers) with a total of 9538 authors, i.e., 3.1 authors per paper. There are 3398 authors in total. The numbers of orderings obtained for various thresholds are shown in Table 2.

From the results in Table 2 we note that the number of elements in the answer set increases rapidly with decreasing frequency threshold σ and increasing violation fraction threshold τ. One should observe, however, that it is easy to find values of σ and τ that produce outputs of desired size.

Recall that for each fragment also the reverse fragment has the same frequency and violation fraction. Thus the algorithm for finding fragments find produces all fragments basically twice. The column N in Table 2 shows how many distinct subsets there are. We discussed previously the case

[2]http://liinwww.ira.uka.de/bibliography/

σ (in %)	τ (in %)	$Max\ l$	$\lvert\mathcal{T}\rvert$	N	P
0.3	0	6	58	29	0
0.3	10	6	96	41	7
0.3	20	8	132	56	9
0.3	30	8	134	56	9
0.3	40	8	134	56	9
0.2	0	8	268	133	1
0.2	10	9	684	273	19
0.2	20	11	1466	681	47
0.2	30	14	2546	1153	110
0.2	40	16	4928	1955	394

Table 2: Number of elements in $\mathcal{T}(\mathcal{D}, \sigma, \tau)$ for various values of σ and τ for the authors of database papers data set. $Max\ l$ is the maximum length of a fragment in the result. $|\mathcal{T}|$ is the total number of fragments found. N is the number of different subsets of attributes present in the output, and P is the number of subsets for which more than two different orderings belong to $\mathcal{T}$.

of a tightly connected set of attributes. This case occurs fairly rarely in the database. One nice example is the triple

⟨R.Agrawal, T.Imielinski, A.Swami⟩

which satisfies the above condition![3]

A converse case is observed for the triplet

⟨D.Florescu, A.Levy, D.Suciu⟩

which has a frequency of 15. This ordering has no violations, while the other orderings have either 1 or 4 violations. Thus we can say that the above ordering is the best among these three authors.

As an example of a long fragment, we have

⟨P.Deshpande, J.Naughton, D.DeWitt, M.Carey, M.Livny, R.Ramakrishnan, D.Srivastava S.Sudarshan, S.Seshadri, R.Rastogi, M.Garofalakis⟩

which has a frequency of 3.3 %: at least two of these authors are authors in 104 papers, while there are 18 violations (17 %). Many of the long fragments, as the one above, are very intuitive for readers of the database literature.

We also tested the spectral clustering method on this data set. The technique cannot be applied to the whole data set, as there are several disconnected components in the underlying graph, i.e., authors who have no chain of coauthorship connections with the other authors. Thus we had to limit the data set by taking the 20 most common authors and including all papers in which at least one of them was an author. The spectral ordering for the top 20 authors produced was

⟨C.Faloutsos, R.Rastogi, A.Silberschatz, Y.Ioannidis, S.Sudarshan, H.Jagadish, M.Stonebraker, M.Livny, M.Carey, R.Agrawal, D.Srivastava, D.DeWitt, R.Ramakrishnan, H.Pirahesh, J.Naughton, S.Chaudhuri, J.Widom, S.Abiteboul, J.Ullman, D.Agrawal⟩

[3]The authors are joint authors in 5 papers in the database, but there is no paper in which only two of them would be authors. There are lots of papers in which one of them is an author, of course.

One of the longer orders produced by the fragment finding algorithm is

⟨K.Shim, R.Rastogi, S.Sudarshan, D.Srivastava, R.Ramakrishnan, M.Livny, M.Carey, M.Franklin, S.Zdonik, S.Acharya, V.Poosala, P.Gibbons, Y.Matias⟩

It has a frequency of 3.2 % (99 observations) and only 3 violations (3%). We see close similarity between the above fragment and the spectral clustering result.

We also tested the similarity of the discovered fragments against the spectral ordering. The results show, as expected, that the fragments mostly capture the same type of ordering information as the spectral method. However, there are many cases in which the fragments yield a more intuitive collection of orders against the forced single order of the spectral method. We omit the details.

We also used the algorithm to find fragments from the titles of the papers in the bibliography. An example fragment is

⟨association, mining, data, management⟩

which has a frequency of 116, and 3 violations.

6.4 Paleontological data sets

In the paleontological dataset (see [12]) the variables are different sites (in different locations) from which fossils have been found; there are 674 of these. As observations, we used the presence or absence of genera, of which there are about 300. The sites have been classified into so-called MN classes; the MN class of a site corresponds roughly to the age of the site. We ran the fragment discovery algorithm to see (1) how well are the fragments compatible with the MN classification, and (2) whether any exceptions can be found. Again, we found that the fragments correspond relatively well with the existing order of the variables. Of the approximately 2000 triplets found with $\sigma = 0.01$ and $\tau = 0.2$, about 1800 where compatible with the MN classes, about 200 had a difference of 1 (e.g., the fragment was $A \prec B \prec C$, but the MN classes were 7,8, and 7, respectively). One triplet had a difference of 2: for the fragment $A \prec B \prec C$ the MN classes were 15, 17, and 15. The existence of the fragment indicates, e.g., that no genera was present in A and C but absent in B, and thus the assignment of MN class 17 to B is suspect. This was confirmed by an investigation of the background for the assignment [9].

7. CONCLUSIONS

We have defined the concept of a fragment of order. Such a fragment is an ordering of a subset of variables in a 0-1 dataset. We described the criteria, frequency and violation fraction, to be used for finding potentially interesting fragments of order. We gave a simple A priori-like algorithm for discovering all fragments of order that satisfy the thresholds, and describe how further pruning can be done by selection among peers, if necessary.

We gave preliminary empirical results on several datasets. The results on the course enrollment data set showed that the method is able to find large fractions of the underlying structure of the curriculum. The results on the bibliographic database demonstrated that the method also yields intuitively appealing results in the case where there is no known

underlying order. We also studied briefly the relationship of the fragments to an ordering of the variables obtained by spectral methods. Even the first results on paleontological data were strong enough to lead to the discovery of an error in the original data set.

Obviously, a lot remains to be done. We are currently conducting a much larger set of experiments, and also devising methods for estimating the recall and precision of a set of fragments with respect to a known partial order.

A possible extension of the above framework is to consider the discovery of partial orders. This leads to interesting issues. Consider for example a partial order in which $A \prec B, A \prec C, B \prec D$ and $C \prec D$. What constitutes a violation of this order? If we see A and D in an observation, do we require that both or at least one of B and C is also seen? The first interpretation is conjunctive, and it yields a fairly well behaving concept class, which has several monotonicity properties. The second, disjunctive interpretation, on the other hand, gives a concept class which fails many monotonicity test both in theory and practice. Finding a well-behaving class of partial orders would be of interest.

8. REFERENCES

[1] R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In P. Buneman and S. Jajodia, editors, *Proceedings of ACM SIGMOD Conference on Management of Data (SIGMOD'93)*, pages 207 – 216, Washington, D.C., USA, May 1993. ACM.

[2] R. Agrawal and R. Srikant. Mining sequential patterns. In *Proceedings of the Eleventh International Conference on Data Engineering (ICDE'95)*, pages 3 – 14, Taipei, Taiwan, Mar. 1995.

[3] J. E. Atkins, E. G. Boman, and B. Hendrickson. A spectral algorithm for seriation and the consecutive ones problem. *SIAM Journal on Computing*, 28(1):297–310, Feb. 1999.

[4] K. S. Booth and G. S. Lueker. Linear algorithms to recognize interval graphs and test for the consecutive ones property. In ACM, editor, *Conference record of Seventh Annual ACM Symposium on Theory of Computing: papers presented at the Symposium, Albuquerque, New Mexico, May 5–May 7, 1975*, pages 255–265, New York, NY, USA, 1975. ACM Press.

[5] K. S. Booth and G. S. Lueker. Testing for the consecutive ones property, interval graphs, and graph planarity using P-Q tree algorithms. *J. of Comp. and Syst. Sci.*, 13:335–379, 1976.

[6] T. F. Chan and D. C. Resasco. A framework for the analysis and construction of domain decomposition preconditioners. Technical Report CAM-87-09, UCLA, 1987.

[7] F. R. K. Chung. *Spectral Graph Theory*. CBMS Regional Conference Series in Mathematics, 1997.

[8] R. Durbin, S. Eddy, A. Krogh, and G. Mitchison. *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press, Cambridge, 1998.

[9] M. Fortelius. Private communication. 2003.

[10] J. Han and M. Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann, 2000.

[11] Hsu. A simple test for the consecutive ones property. *Journal of Algorithms*, 43, 2002.

[12] J. Jernvall and M. Fortelius. Common mammals drive the evolutionary increase of hypsodonty in the neogene. *Nature*, 417:538–540, 2002.

[13] Y. Koren and D. Harel. Multi-scale algorithm for the linear arrangement problem. Technical Report MCS02-04, Faculty of Mathematics and Computer Science, The Weizmann Institute of Science, 2002.

[14] H. Mannila and C. Meek. Global partial orders from sequential data. In *Proceedings of the sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), Boston, MA*, pages 161–168. ACM Press, 2000.

[15] H. Mannila, H. Toivonen, and A. I. Verkamo. Discovery of frequent episodes in event sequences. *Data Mining and Knowledge Discovery*, 1(3):259 – 289, Nov. 1997.

[16] A. Ng, M. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *In Advances in Neural Information Processing Systems*, 2001.

[17] A. Popescul, G. W. Flake, S. Lawrence, L. H. Ungar, and C. L. Giles. Clustering and identifying temporal trends in document databases. In *ADL 2000*, pages 173–182, 2000.

[18] A. Pothen, H. Simon, and L. Wang. Spectral nested dissection. Technical Report CS-92-01, Pennsylvania State University, Department of Computer Science, 1992.

[19] R. Ramakrishnan and J. Gehrke. *Database Management Systems (2nd ed.)*. McGraw-Hill, 2001.

[20] H. D. Simon. Partitioning of unstructured mesh problems for parallel processing. *Computing Systems in Engineering*, 2, 1991.