Finding Partial Orders from Unordered 0-1 Data

Antti Ukkonen
Helsinki University of Technology
Laboratory for Computer and Information Science
P.O. Box 5400
FI-02015 TKK
antti.ukkonen@hut.fi

Mikael Fortelius
University of Helsinki
Department of Geology
P.O Box 64
FI-00014 University of Helsinki
mikael.fortelius@helsinki.fi

Heikki Mannila
Department of Computer Science
University of Helsinki &
Helsinki University of Technology
P.O. Box 68, FI-00014
University of Helsinki
mannila@cs.helsinki.fi

ABSTRACT

In applications such as paleontology and medical genetics the 0-1 data has an underlying unknown order (the ages of the fossil sites, the locations of markers in the genome). The order might be total or partial: for example, two sites in different parts of the globe might be ecologically incomparable, or the ordering of certain markers might be different in different subgroups of the data. We consider the following problem. Given a table over a set of 0-1 variables, find a partial order for the rows minimizing a score function and being as specific as possible. The score function can be, e.g., the number of changes from 1 to 0 in a column (for paleontology) or the likelihood of the marker sequence (for genomic data). Our solution for this task first constructs small totally ordered fragments of the partial order, then finds good orientations for the fragments, and finally uses a simple and efficient heuristic method for finding a partial order that corresponds well with the collection of fragments. We describe the method, discuss its properties, and give empirical results on paleontological data demonstrating the usefulness of the method. In the application the use of the method highlighted some previously unknown properties of the data and pointed out probable errors in the data.

Categories and Subject Descriptors

H.2.8 [**Information Systems**]: Data mining

General Terms

Algorithms

Keywords

partial order, hidden ordering, consecutive ones property

KDD'05, August 21–24, 2005, Chicago, Illinois, USA.

1. INTRODUCTION

The prototypical examples of data mining include the analysis of large masses of 0-1 data. The typical assumption for such data is that there is no underlying order or partial order for the attributes (variables, columns) or for the observations (rows). This assumption is at least approximately true in many applications, such as market basket analysis, where it is difficult to assume that the variables (items people buy) would have any clear ordering. Likewise, for market basket data the order of the rows might not be very significant.

However, in many other applications the concept of an underlying ordering of the rows or columns (or both) is extremely important. Consider the problem of paleontological data modeling. In such data the columns correspond to species or genera. Each row corresponds to a *site*, i.e., a collection of fossil remains collected from the same location and the same layer in rock. That is, a site is a (very incomplete) snapshot of the set of species that lived at a certain location at approximately the same time. The ages or evolutionary stages of the sites (rows) form a natural ordering for the sites. This order need not be a total order, as two sites of approximately the same age but widely separated in space might not be comparable. The ages of the sites are only rarely known, and the problem of *seriation* is to find good estimates for the ages or ordering of the sites [17, 3, 4, 5]. For the columns, the species or genera, there is also a natural partial ordering, the evolutionary ordering in which the species arise.

How could one find the ordering of the sites from the observed presence or absence of genera? In a good ordering species evolve and become extinct, indicating that the occurrences of the species form an interval in the ordering; as we will see, this basic property can be used in finding good orderings. The ordering can be done, e.g., by using a spectral method [6, 23, 15, 13]. The weakness in these methods is that they produce a total order, regardless of whether the data supports it or not. The method presented in this papers finds the partial order shown in Figure 2, which yields much more information about the true evolutionary relationships between the sites and the genera.

The problem described above is not limited to paleontological applications. Similar situations occur also in genome analysis, i.e., the problem of finding genes that predispose to some specific disease. In this case the rows of the data corre-

spond to *markers*, i.e., base pairs in the genome where there is variation between individuals, and columns correspond to individuals. The value at position j of row i tells what is the nucleotide in the genome of person j at the position of the genome corresponding to row i. Since typically only two nucleotides occur at a marker, the data can be coded with 0s and 1s. For such data, the ordering of the markers corresponds to the location of the points of variation in the genome. While the locations are fairly often well known, there are situations in which the order is uncertain. For example, part (e.g., 20 %) of the population can have large (> 1 M base pairs) regions of the genome inverted; this implies that there is no global total order among the markers.

What data could one use to find the ordering among the rows from such genetic data? The underlying processes producing the variation are mutations and recombinations, and they lead to higher (absolute values of) correlation between markers that are close to each other. This information can be used to obtain good orderings. For this type of data, the question of finding a good partial order comes close to finding a good dependency graph for the variables.

The two applications described above have the following characteristics in common. First, there is a notion of an underlying partial order among the rows. Second, finding the best ordering for the whole set of data is difficult. Third, given a small subset of rows, we can find the best ordering for the subset.

In this paper we give a method for finding good partial orders P from 0-1 data. The method proceeds in stages. It first constructs a collection of fragments of order, i.e., small collections of rows for which there is a clear total ordering. This task is done by selecting random subsets of, say, 5 rows and testing all the permutations.

Second, we orient the fragments. As the methods for evaluating the quality of an ordering are typically symmetric with respect to ordering P and the reverse ordering $P^R = \{ (B, A) \mid (A, B) \in P \}$, we next have to divide the set of fragments into two subsets, one corresponding to P and the other to P^R. The division is done by using a graph partitioning method: we consider the graph having the fragments as vertices. The weight of an edge between two fragments indicates how many pairs of rows are ordered in the same way in the fragments. We use a simple local search method for finding a reasonably good bisection of the graph.

The third step in our method is the construction of a partial order from the fragments from one of the classes produced by the previous step. For this we apply a simple greedy algorithm that looks again at all pairs of rows from the data. For each pair (A, B) we compute the number of fragments in which A and B occur and A precedes B or B precedes A. If the number of times A precedes B is clearly higher than the number of times B precedes A, then we have evidence for $(A, B) \in P$. All such pairs cannot necessarily be added to the partial order, as the order might become cyclic; a greedy approach is used to select the most useful pair to be added.

We demonstrate the usefulness of our method by applying it to paleontological data. The resulting partial orders provide a very clear overview of the underlying structure of the data set, and the details of the result correspond well with the known eras of the last 25 million years. Furthermore, some aspects of the results point out probable errors in the seriation data in existing literature; thus the method for finding partial orders yields useful scientific information.

The rest of this paper is organized as follows. We start in Section 2 by discussing the motivation for searching for partial orders. We also describe the paleontological application in some more detail, and formally present the problem and describe some simplistic and infeasible methods for solving it. The next three sections describe the three parts of the solution. First, in Section 3, we consider the problem of generating small totally ordered subsets of rows. Second, in Section 4 we consider the question of orienting the fragments by graph partitioning. Third, we describe the construction of the partial order from the totally ordered fragments in Section 5. Empirical results are given in Section 6. Section 7 is a short conclusion.

2. TOTAL AND PARTIAL ORDERS

In this section we consider the motivation for finding partial orders from 0-1 data. Let R be the set of attributes (variables) of our 0-1 dataset. Let M be a set of 0-1 rows over the attributes R. The value at column A of row t is denoted by $t(A)$. In our main application the data sets typically have hundreds or thousands of rows; thus quadratic or cubic time is acceptable.

A *partial order* on M is a binary relation P such that $(t, t) \in P$ for all rows $t \in M$ (reflexivity), if $(t, u) \in P$ then $(u, t) \notin P$ (symmetry), and if $(t, u) \in P$ and $(u, v) \in P$, then $(t, v) \in P$ (transitivity). The partial order is *trivial*, if $(t, u) \in P$ only if $t = u$, and P is a *total order*, if for all u and v either $(u, v) \in P$ or $(v, u) \in P$. Partial orders are typically drawn as directed acyclic graphs, where the edges deducible by transitivity are omitted (i.e., one draws only the edges in the transitive reduction of P). The size of a partial order is the number of ordered pairs it contains and is denoted by $|P|$.

As an example, consider paleontological data. Recall that the columns correspond to species or genera, and each row corresponds to a site. In a good ordering species evolve and become extinct, indicating that the occurrences of the species form an interval in the ordering. Thus, the ordering of the sites below on the left is biologically not as natural as the one on the right.

0	0	1		0	0	1
1	1	0		0	1	1
0	1	1		1	1	1
0	1	0		1	1	0
1	1	1		0	1	0

On the left side, there are several cases in which a species is first present, then absent, and then present again; for example, in the first column there are two zeros between the ones. Such zeros are known as *Lazarus events*. The left matrix has four Lazarus events. On the right side, there are none.

Given a total order T on the row set M, the *Lazarus count* of M with respect to T is defined as

$$L(M \mid T) = \big|\{ (u, A) \mid u(A) = 0 \wedge \exists v \in M : v(A) = 1 \wedge (v, u) \in T \wedge \exists w \in M : w(A) = 1 \wedge (u, w) \in T \}\big|.$$

That is, the Lazarus count of a 0-1 dataset under a certain ordering of the rows is the number of zeros in the data that

are between the first and last ones in their column.

The question of determining whether for some total order T we have $L(M|T) = 0$ (finding an ordering of the rows of a 0-1 matrix with no Lazarus events) is the problem of determining whether a binary matrix has the *consecutive ones property*. A binary matrix with the rows M is said to have the consecutive ones property if there exists a permutation π of the rows, such that in $\pi(M)$ the ones are aligned to consecutive rows on every column. A well known application of this problem is physical mapping of chromosomes [2], while matrix envelope reduction [21] is a related problem.

In the case where $L(M|T) = 0$ for some total order T, the order T can be determined in linear time [9, 18]. In practice, however, the data M has $L(M|T) > 0$ for all total orders T. The reason is that the data contains noise in form of false positives (false ones) and false negatives (false zeros); especially false zeros (cases where a genus is not observed when it was actually present) are abundant: their number can be about as high as the number of true ones.

A number of methods [2, 6, 13] have been proposed for finding good ordering. In general the objective of these algorithms is to find a single total order of the rows that is optimal according to the used criteria. A typical approach is to minimize the number of Lazarus events by some means. However, in many cases determining a strict total order is not useful. The available data may be insufficient to yield a precise order on the rows. As an example, consider the following simple matrix with one Lazarus event:

1:	1	0	0
2:	1	1	0
3:	1	**0**	1
4:	0	1	1
5:	0	0	1

No permutation would yield a completely error-free ordering, thus this matrix does not have the consecutive ones property. But it is not hard to see that there are two other permutations of the rows that result in one Lazarus event as well, namely:

1:	1	0	0		1:	1	0	0
2:	1	1	0		3:	1	0	1
4:	**0**	1	1	and	2:	1	1	**0**
3:	1	0	1		4:	0	1	1
5:	0	0	1		5:	0	0	1

Every other permutation except the reversals of these three have a higher Lazarus count. If we consider the indices of the matrix rows, let the first one correspond to the order (1 2 3 4 5). Then the second one is given by (1 2 4 3 5) and the third one by (1 3 2 4 5). Forcing a total order on the rows of the above matrix by minimizing the Lazarus count is thus bound to contain some randomness, since all three orders and their reversals are in fact equally good. For example, spectral ordering [6] gives (1 2 3 4 5) as a solution. There is no reason to prefer this over the two others if the only criteria is number of Lazarus events.

Instead of selecting one of the orderings, we can gather all common features of the three orders together and based on these construct a partial order on M. In this example it is relatively easy to see merely by looking at the orders that in each case row 1 precedes and row 5 follows every other row. The order changes only with respect to the three middle rows. Restricting our attention to these, we see that in every case row 2 precedes row 4. In this case there are no more common features to be found. These features together form the partial order which is depicted in Figure 1 as a directed acyclic graph.

Note that we did not consider the reversals of the three orders when constructing the partial order even though they result in only one lazarus event as well. In general it is not possible to know the correct direction of the order based on the consecutive ones property alone. When constructing the partial order from a set of total orders we must make sure that at least a sufficient number of the orders "point to the same direction". We will address problems caused by this in Section 4.

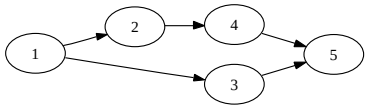

Figure 1: A directed acyclic graph that represents the partial order on the rows (their indices) of the example matrix. If there is no path between two nodes then their mutual order is undetermined.

Thus it makes sense to search for partial orders instead of total orders. Consider now a partial order P among the rows a 0-1 matrix with the row set M. The number of Lazarus events in M with respect to P can be defined in exactly the same way as for the case of a total order:

$$L(M \mid P) = \big|\{\, (u, A) \mid u(A) = 0 \\ \wedge \exists v \in M : v(A) = 1 \wedge (v, u) \in P \\ \wedge \exists w \in M : w(A) = 1 \wedge (u, w) \in P \,\}\big|.$$

That is, we count the number of zeros $t(A)$ in the data such that there are rows u and v before and after t with respect to the partial order P such that $u(A) = v(A) = 1$.

Our problem can now be posed as follows: given the data M, find a partial order P such that $L(M|P)$ is small. However, this formulation is incomplete: we can easily make $L(M|P)$ to be 0 by having P to be the trivial partial order. Thus, our task is to find a partial order P minimizing the quantity

$$L(M|P) + \alpha(P)$$

where $\alpha(P)$ is some function of P that is small for total orders and becomes larger as P approaches the trivial partial order. The choice of the function α is an interesting question. Some possible choices for $\alpha(P)$ are the logarithm of the number of linear extensions of P (computing this, however, is $\sharp$P-complete [10]), or the size of P, which for n rows varies between n for the trivial partial order and $n(n+1)/2$ for a total order.

Finding the optimal partial order P from scratch is, however, quite difficult. There are several reasons for this. One is that individual elements of the partial order P do not have an immediate effect on $L(M|P)$; whether a 0 in the data contributes to $L(M|P)$ depends on at least two elements of P. The second reason is that it is not easy to construct a good set of operations that would allow one to do heuristic search over the set of all partial orders. In [22] the approach used was limited to constructing series-parallel partial orders, and still the set of operations is more complex than one would wish.

For these reasons we did not develop an algorithm that would directly aim at a search over the set of all partial orders. Rather, we construct the partial order by first collecting information about the precedence relationships between rows. We construct a set of fragments, total orders over small subsets of the data, and use the fragments to obtain good partial orders. The quality of the solutions is still evaluated by using, among others, the quantity $L(M|P)$.

3. COMPUTING FRAGMENTS

In the first stage of our algorithm we compute a set of small total orders from the data. Given the data M, a *fragment* f of size k is a totally ordered subset of the rows of M:

$$f = (u_1, u_2, \ldots, u_k),$$

where u_i is a row of M. Such fragments of order were considered in [15], where an algorithm for discovering fragments from unordered data is presented. The approach is similar to the Apriori algorithm [1] for discovering frequent itemsets. However, generating longer fragments with the method is cumbersome because computation of fragments of length n requires the computation of all fragments of length $n-1$ first. To obtain enough information about the order we need fragments containing at least 3 rows, as the notion of a Lazarus event does not make sense for subsets of 2 rows. In practice the methods work far better when they operate on fragments of, say, 5 rows. If there is an underlying total order in the data, there are $\binom{n}{5}$ such fragments, and the algorithm of [15] would compute all of them; this is clearly infeasible even for moderate values of n.

The solution we use for generating fragments is simple. We randomly select a subset M' containing k rows from the dataset M, and evaluate $L(M'|T)$ for all $k!$ total orders T on M'. A fragment is created by a total order T together with the set M' when

$$L(M'|T) \leq \mu,$$

where μ is the maximum number of Lazarus events allowed. One M' will thus result in a number of different fragments depending on μ. For any μ we will always get an even number of fragments. If we get the fragment $f = (u_1, u_2, \ldots, u_k)$ we will obtain its *reversal* $f^R = (u_k, u_{k-1}, \ldots, u_1)$ as well, since they both have the same Lazarus count. Should the total number of fragments produced by a subset M' be very high, the rows in M' fail to provide useful information about the underlying order of the complete dataset. For example, if the sites in M' have no common genera, all the $k!$ permutations are equally good with a Lazarus count of zero. To prevent massive amounts of such useless fragments being generated, we output the set of fragments produced by M' only if it contains at most two (reversals excluded) fragments; otherwise no fragments are produced. This procedure is repeated until the desired number of fragments is obtained.

The complexity of this step is $O(k!kd)$ per fragment, where k is the number of rows in the fragment and $d = |R|$ is the dimension of the data. In practice the running time is influenced by μ as well. When μ is set too low fragments are formed less frequently (or possibly not at all), which results in slower sampling.

4. ORIENTING THE FRAGMENTS

As stated in the previous sections, using the Lazarus count to determine good orderings for the rows has a small drawback. For every discovered fragment f we also find its reversal f^R. This problem may occur also with other types of data when the criterion used for sampling fragments is invariant with respect to the direction of the underlying order P. The next task is to divide the set of fragments into two classes, one corresponding to P and the other to the reverse of P. We call this problem *orienting* the fragments.

Our solution is based on graph partitioning. Given a set S of fragments, let G be a weighted graph having as vertices the fragments $f \in S$. For a fragment $f = (u_1, u_2, \ldots, u_k)$, denote $T(f) = \{(u_i, u_j) | 1 \leq i < j \leq k\}$, i.e., the set of pairs of entries occurring in f in their order. Two vertices (fragments) f_1 and f_2 with $f_1 \neq f_2$ are connected in G by an edge if they share at least two elements and order them in the same way, i.e., if $C = \{\ T(f_1) \cap T(f_2)\ \} \neq \emptyset$. The weight $w(f_1, f_2)$ of edge (f_1, f_2) is given by $|C|$. This weight is essentially a measure of similarity between two fragments that simply counts the number of their common orderings. We define $w(f, f) = 0$ for all f. Obviously there is no edge between f and f^R in G. Our objective is to partition the vertices of G to sets V_1 and V_2 so that f and f^R end up in different sets, and that fragments in the same set are oriented to the same direction. As a cost function we use the sum

$$\sum_{f_1 \in V_1, f_2 \in V_2} w(f_1, f_2) \qquad (1)$$

of weights of edges crossing the boundary of V_1 and V_2.

A straightforward way of doing the graph partitioning would be to use some spectral techniques, as in [12, 23, 20]. An alternative is to use simple local search methods, and in these one can use some specific information about the fragments (f and f^R should end up in different sets V_i). In our experiments we used a local search technique. The algorithm starts by assigning for each $f \in S$ one of f and f^R at random to V_1 and the other to V_2. Then local moves swap f and f^R until no improvement can be achieved. This way a fragment and its reversal remain in separate sets at every step.

The method works extremely well in practice. Constructing G can be done in time $O(|S|^2 k^2)$ where k is the length of a fragment.

5. DETERMINING THE PARTIAL ORDER

The previous sections described the discovery of good fragments of order and the orientation of the orders. The final step in the algorithm is to produce a partial order P, given a set V_1 of total orders that are oriented in the same direction.

In this section we consider the general problem of determining a partial order P that describes the set of total orders in the best possible way. The same problem is addressed in [22] and more recently in [11]. The approach of [22] is computationally demanding and restricted on a certain class of partial orders, the series-parallel ones. The method of [11] is intended for discovering several small partial orders from a set of sequences instead of only one that describes all or most of the set. For our purpose a simple approach is sufficient.

Given a collection V_1 of fragments, consider two rows u and v from the data M. Let $F(u, v)$ denote the number of

times u and v occur in the same fragment with u preceding v, i.e.,

$$F(u,v) = \big|\{f \in V_1 \mid (u,v) \in T(f)\}\big|.$$

Suppose $F(u,v) \approx F(v,u)$. In such a case the data gives no reason to prefer either of the orderings and neither (u,v) or (v,u) should not be included to P. On the other hand, if $F(u,v)$ is considerably larger than $F(v,u)$ the data has more evidence of the order (u,v) and we should include (u,v) to P. We define the score function

$$D_\epsilon(P) = \sum_{(u,v)\in P} \big(F(u,v) - (1+\epsilon)F(v,u)\big), \qquad (2)$$

where ϵ is a positive real valued parameter that determines how much more evidence we must have for $F(u,v)$ in order to prefer it over $F(v,u)$.

Our method of discovering P is based on maximizing this function. Each pair (u,v) with

$$\Delta(u,v) = F(u,v) - (1+\epsilon)F(v,u) > 0$$

is a potential candidate for inclusion into the partial order P. It is not hard to show that in fact all such pairs (u,v) are candidates for inclusion for which $\frac{F(u,v)}{F(u,v)+F(v,u)} > \frac{\epsilon+1}{\epsilon+2}$ holds. However, all such pairs cannot be blindly added to P, as the transitivity requirement might lead to violation of the antisymmetry conditions. We use a simple greedy approach, by adding the pairs (u,v) in descending order of $\Delta(u,v)$ and discarding the pairs that lead to violations of antisymmetry (cycles). To make the check for antisymmetry fast we maintain the transitive closure of P at all times. The complexity of the method is $O(n^2)$ per added pair in P; as there are $O(n^2)$ potential pairs (u,v), the running time is $O(n^4)$. In practice the method works much faster.

6. EXPERIMENTAL RESULTS

We tested the method on a paleontological dataset stemming from [19]. Recall that in a 0-1 matrix the rows correspond to fossil discovery sites and columns to different genera. The number of sites and genera were 124 and 139, respectively. Average number of genera at a site was 15.95 with a minimum of 10 and a maximum of 39 (standard deviation 5.6). The data has lots of false zeros, because there are several natural reasons why a species or genus is not found even though the species actually occurred at the site. The probability of a false zero is estimated to be roughly 0.5. In addition, a small fraction of the ones are incorrect due to human error (approximately 1 percent).

The sites have been classified to 14 temporal classes (the MN classes) based on their approximate age. The MN numbers increase as the sites get more recent with the youngest ones in class 17. It should be emphasized that the classification of sites to MN classes is a difficult task, and the criteria are known to vary (see, e.g., [7, 3]). One of our goals is to look for sites whose classification should be reevaluated.

Still, a comparison with the MN system is an important evaluation criterion for our method. We denote the partial order our algorithm gives by P. Clearly the MN classes define a partial order on the sites as well; we call it P_{MN}. Given P it is interesting to see how well it correlates with P_{MN}, that is, how many of the pairwise orderings present in P are compatible with the MN numbers. Denote the MN number of site u by u_{MN}. Similarly, one can look at the pairs

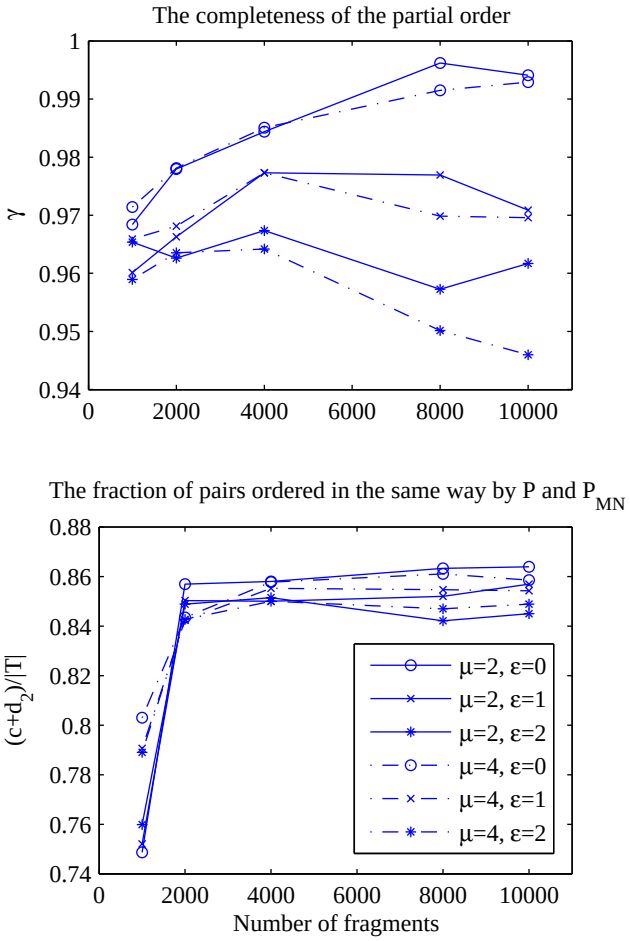

Figure 3: Upper panel: the completeness γ of the resulting partial order for different values of μ and ϵ as a function of the number of sampled fragments. Lower panel: the compatibility of P with respect to P_{MN} for different values of μ and ϵ as a function of the number of sampled fragments.

of sites for which P makes no judgment, and check how many are unordered or near to each other in P_{MN} as well. More precisely, let

$$c = \big|\ \{\ (u,v) \in P \mid u_{\text{MN}} < v_{\text{MN}}\ \}\ \big|,$$

$$\bar{c} = |P| - c,$$

$$d_x = \big|\ \{\ \{u,v\}\big|\ (u,v) \notin P \wedge (v,u) \notin P \wedge |u_{\text{MN}} - v_{\text{MN}}| \le x\ \}\ \big|,$$

and

$$\bar{d_x} = \big|\ \{\ \{u,v\}\big|\ (u,v) \notin P \wedge (v,u) \notin P \wedge |u_{\text{MN}} - v_{\text{MN}}| > x\ \}\ \big|$$

with either $x = 1$ or $x = 2$. Thus c and d_x are measures of similarity between P and P_{MN} when ordered and unordered elements are considered. Should it happen that P orders the sites the "wrong" way (recent to old), we simply use the reverse P^R. Note that

$$c + \bar{c} + d_x + \bar{d_x} = |T|,$$

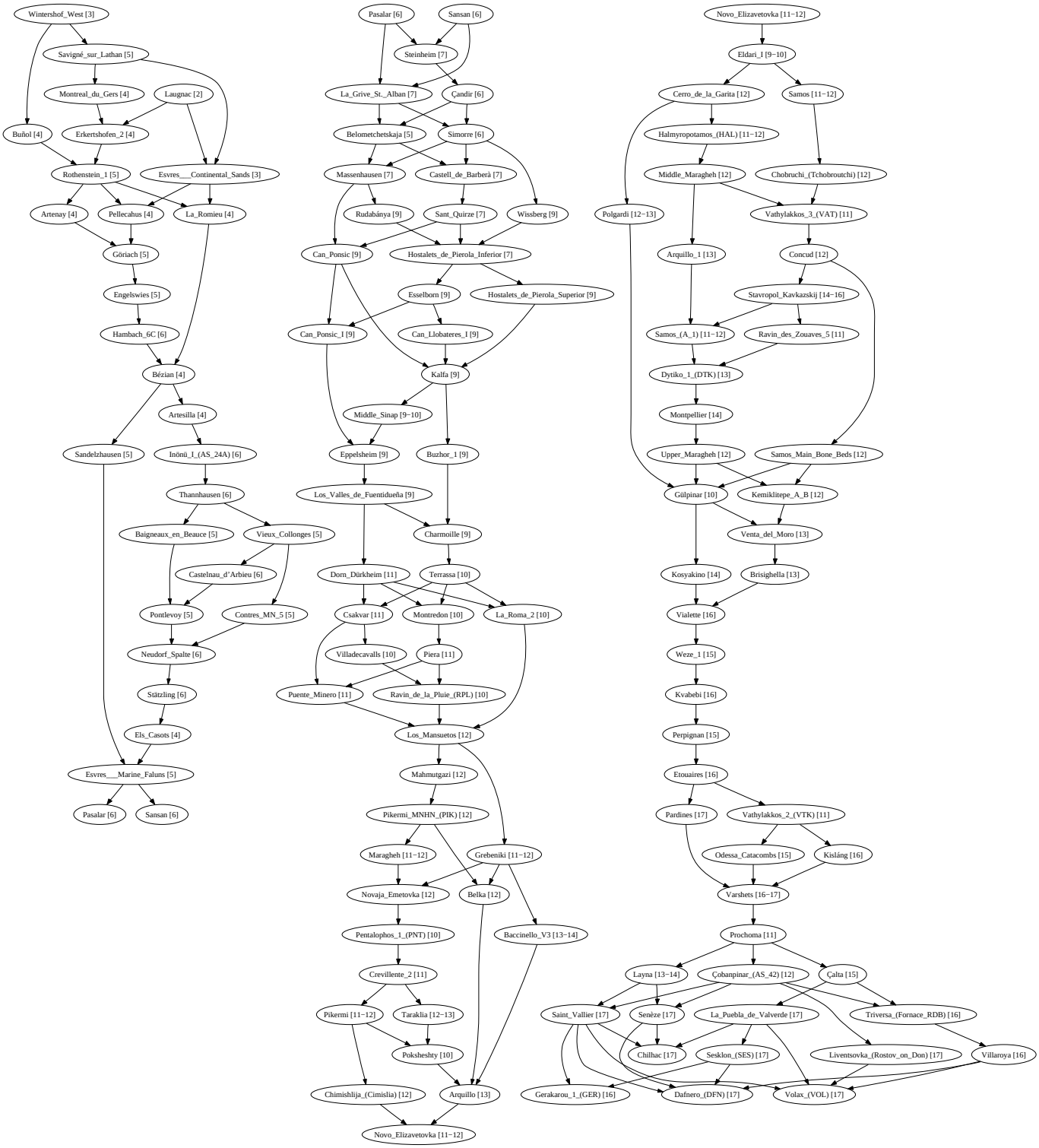

Figure 2: The discovered partial order for the sites presented in three parts ($n = 4000$, $l = 5$, $\mu = 2$ and $\epsilon = 1$). The ordering progresses from top to down and left to right. The number in brackets next to the name of a site is the MN class of the site. The original order is connected but has been split to pieces for presentational purposes; nodes in the bottom and top are repeated for ease of understanding. The image was generated with the *tred* and *dot* utilities provided by the Graphviz graph visualization software [14]. See also `www.graphviz.org`.

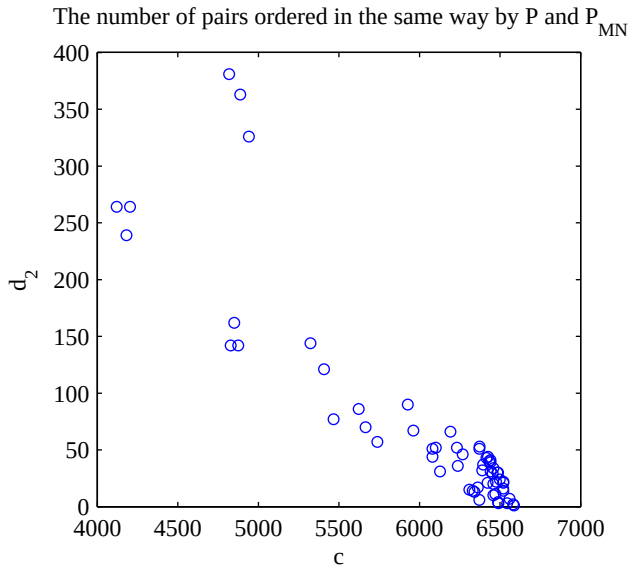

Figure 4: The number of pairs ordered in P and P_{MN} versus the number of pairs unordered in P and P_{MN}.

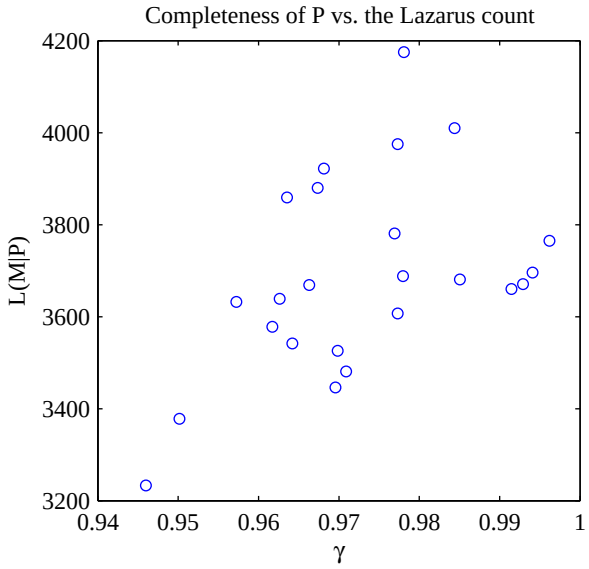

Figure 5: The Lazarus count versus the completeness γ of the partial order P.

where T is some total order on the sites. The fraction $\frac{c+d_x}{|T|}$ can be considered a measure of compatibility of P with respect to P_{MN}. Another informative property of P is its *completeness* γ, defined as $|P|/|T|$.

To evaluate the method's sensitivity with respect to the length of a fragment l, total number of sampled fragments n, the value of ϵ and μ, we computed P varying the parameter values as follows: $n \in \{1000, 2000, 4000, 8000, 10000\}$, $l \in \{4, 5\}$, $\epsilon \in \{0, 1\}$ and $\mu \in \{2, 4\}$.

The results are shown in Table 1 and Figures 3, 4, 5 and 6. The values of γ, c, d_2 and $L(M|P)$ are provided as well for P_{MN}. Obviously P_{MN} is compatible with itself, hence the high values of c and d_2. The low Lazarus count of P_{MN} is largely explained by its low completeness. All partial orders found by the algorithm are more specific than P_{MN}.

A number of observations can be made from Figure 3. First of all, the number of fragments has in practice only a minor effect both on the completeness γ and on the compatibility of P and P_{MN}. Thus, a high coverage of the sites is achieved already with a low fragment count. The completeness increases slightly as n increases when $\epsilon = 0$; for larger ϵ the effect is less clear. More obvious is the effect of ϵ itself on γ. This is to be expected as the number of candidate pairs for inclusion to P must decrease for larger ϵ. Increasing μ from 2 to 4 seems to cause a slight decrease in the completeness.

Figure 4 plots c on the horizontal and d_2 on the vertical axis. Clearly c and d_2 exhibit a inversely proportional relationship, that is, as the number of ordered pairs in common to P and P_{MN} increases, the number of common unordered elements decreases.

$L(M|P)$ is minimized when P is the trivial partial order with $\gamma = 0$. Conversely, $L(M|T)$ is maximized for some total order T. $L(M|P)$ might be maximized for some partial order as well, but completing P to a total order can not decrease the Lazarus count. It follows that $L(M|P)$ increases as γ does. Figure 5 is in accordance with this proposition. The Lazarus count of the total order given by spectral ordering is 3792; most of the resulting partial orders have a smaller Lazarus count, indicating that the partial orders tend to omit precedence relations that cause Lazarus events. Also partial orders with a high completeness ($\gamma > 0.99$) tend to perform well in terms of $L(M|P)$: this supports our claim that the partial orders describe relevant orderings of the data.

The probability $Prob((u, v) \in P_{\text{MN}}|(u, v) \in P)$ is given by $c/(c + \bar{c})$. It determines the probability of a pair $(u, v) \in P$ being correct with respect to P_{MN}. Figure 6 indicates a reasonably strong correlation with this probability and $L(M|P)$. Orders resulting in less Lazarus events are also more "correct" and vice versa.

The results show that the algorithm produces partial orders that are in quite good agreement with the partial order given by the MN system. The correspondence is by no means full; there are interesting variations between the orderings.

One partial order generated by our method was also qualitatively analyzed. The analysis was based on the visual description of the partial order given in Figure 2. Of every site is given its name and previously known MN class (in brackets). The visualized graph is the transitive reduction of the original one and has been split to three parts for ease of presentation. Note that only the arrows are significant, the nodes are positioned to minimize edge crossings. Thus, even though e.g. Sandelzhausen is placed right next to Inönü_I (see left part of Figure 2) one should not consider these two more similar than Sandelzhausen and Stätzling, for instance. Constructing the transitive reduction for the visualization can in practice be very slow.

The overall pattern generated is in remarkable agreement with current understanding of the evolutionary history of European land mammal faunas [7, 24]. For example, the first cluster (Wintershof-West to Artesilla) comprises early Miocene localities. (The one apparent exception, the site Hambach_6C, has a faunal list entirely compatible with an MN 5 age.) As another example, the partial order shows

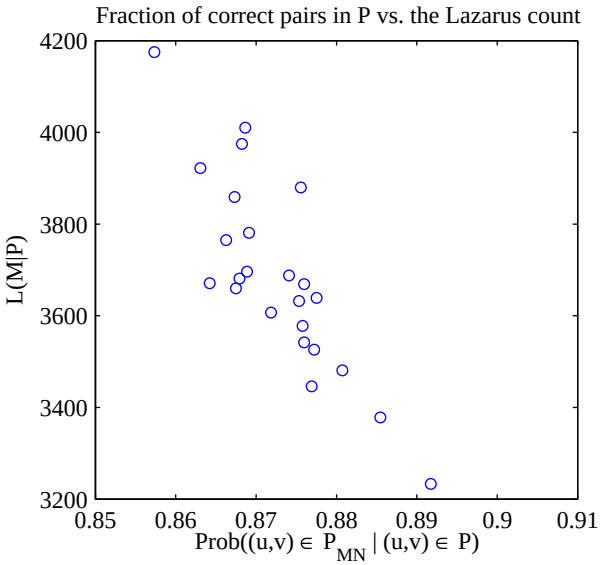

Figure 6: The Lazarus count versus the the fraction of correct pairs in P with respect to P_{MN}.

clearly the transition to the late Miocene, defined by the entry of the horse genus *Hipparion* from North America. The ordering of these sites shows an interesting pattern. Three *Hipparion*-localities (Massenhausen, Wissberg and Rudabanya) cluster with last middle Miocene localities, most likely owing to their high content of relictual faunal elements [8]. Otherwise the early (MN 9) late Miocene localities form a homogeneous group (Hostalets de Pierola Superior/Esselborn/Can Ponsic to Charmoille).

Similar comments can be made on the whole of the partial order. Altogether the order shows a very good correspondence between the previously known ages of the sites, and approximately 6–7 MN classes of sites will be re-evaluated on the basis of the partial order.

While the worst-case running times of the algorithm are quite large, they run reasonably fast in practice. The bottleneck of the process is the sampling of fragments, which is largely affected by the fragment length l, the maximal Lazarus count μ and obviously the size of R (number of genera in our case). Decreasing μ or increasing l or $|R|$ makes the sampling slower, because the probability of a random set of l rows having a permutation with a Lazarus count less than μ clearly decreases as l increases and/or μ decreases.

7. CONCLUDING REMARKS

We have described a method for discovering a partial order from 0-1 data. The approach first constructs short total orders, fragments, from the data, then orients them, and finally uses the precedence information in the patterns to form the partial order. We have applied the method to paleontological data, and the results yield novel information about several fossil sites.

As mentioned in the introduction, the approach can also be used for other types of data. The crucial features of the data are the following. First, there is a notion of an underlying partial order among the rows. Second, finding the best ordering for the whole set of data should be difficult. Third, given a small subset of rows, we can find the best ordering for the subset. We have made preliminary experiments on marker data from medical genetics, and the method yields promising results also there. For that application, the question of finding a good partial order resembles the problem of finding a good dependency graph for the variables.

n	l	μ	ϵ	γ	c	d_2	$L(M\|P)$
2000	5	2	0	0.977	6519	16	3688
2000	5	2	1	0.966	6455	29	3669
2000	5	2	2	0.962	6442	31	3639
2000	5	4	0	0.978	6395	37	4175
2000	5	4	1	0.968	6372	51	3922
2000	5	4	2	0.963	6373	53	3859
4000	5	2	0	0.984	6521	22	4010
4000	5	2	1	0.977	6471	11	3975
4000	5	2	2	0.967	6459	34	3880
4000	5	4	0	0.985	6520	21	3681
4000	5	4	1	0.977	6498	24	3607
4000	5	4	2	0.964	6441	41	3542
8000	5	2	0	0.996	6581	2	3765
8000	5	2	1	0.976	6475	22	3781
8000	5	2	2	0.957	6390	32	3632
8000	5	4	0	0.991	6559	7	3660
8000	5	4	1	0.969	6488	30	3526
8000	5	4	2	0.950	6416	43	3378
10000	5	2	0	0.994	6587	1	3696
10000	5	2	1	0.970	6521	14	3481
10000	5	2	2	0.961	6423	21	3578
10000	5	4	0	0.992	6544	3	3671
10000	5	4	1	0.969	6484	30	3446
10000	5	4	2	0.945	6433	40	3233
MN system (P_{MN})				0.917	6995	631	2957

Table 1: Results of experiments. n: the number of fragments; l: the length of a fragment; μ: the maximum number of Lazarus events allowed per fragment; ϵ: the parameter of Equation 2; c, d_2: see discussion above; $L(M|P)$: the Lazarus count of the partial order P.

The proposed method contains interesting topics for further research. Determining good criteria for accepting a set of fragments produced by the subset M' is crucial for the performance of the sampling. How μ should be set clearly depends on l and the size of R, but also on the estimated number of false zeros in the data. This dependence should be investigated further, not only because it affects the running time of sampling, but also because it has an effect on the quality of the fragments. Unsuitable values of μ can cause certain rows (sites) to have a higher frequency than others in the produced fragments.

The problem of fragment orientation is not restricted to a case where the fragments are generated by minimizing the Lazarus count. Suppose we construct fragments by using a simple likelihood function that is based on pairwise similarities of the observations. If the similarity measure is symmetric, the likelihoods of fragments f and f^R are the same.

Perhaps the most interesting open question is what type of an algorithm could be used to find a partial order P minimizing

$$L(M|P) + \alpha(P)$$

for some suitable chosen functions α describing the complexity of the partial order. As mentioned in Section 2, this problem seems fairly difficult: the space of potential partial orders is large and it seems difficult to do even local search in it. We are currently pursuing two approaches to this problem: one is based on the idea of using MCMC techniques over the set of partial or total orders. The other is based on randomized algorithms [16].

8. REFERENCES

[1] R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In *Proceedings of the ACM SIGMOD Conference on Management of Data*, pages 207–216, 1993.

[2] F. Alizadeh, R. M. Karp, D. K. Weisser, and G. Zweig. Physical mapping of chromosomes using unique probes. In *Proceedings of the fifth annual ACM-SIAM symposium on Discrete algorithms*, pages 489–500, 1994.

[3] J. Alroy. Appearance event ordination: a new biochronologic method. *Paleobiology*, 20:191–207, 1994.

[4] J. Alroy. Diachrony of mammalian appearance events: Implications for biochronology. *Geology*, 26:23–26, 1998.

[5] J. Alroy. New methods for quantifying macroevolutionary patterns and processes. *Paleobiology*, 26:707–733, 2000.

[6] 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.

[7] R. Bernor, V. Fahlbusch, P. Andrews, H. d. Bruijn, M. Fortelius, F. Rool, F. F. Steininger, and L. Werdelin. The evolution of Western Eurasian Neogene mammal faunas: A chronologic, systematic, biogeographic and paleoenvironmental synthesis. In *The Evolution of Western Eurasian neogene Mammal Faunas*, pages 449–471. Columbia University Press, 1996.

[8] R. Bernor, L. Kordos, L. Rook, P. A. J. Agusti, M. Armour-Chelu, D. Begun, D. Cameron, G. Daxner-Hoeck, L. d. Bonis, J. Damuth, O. Fejfar, N. Fessaha, M. Fortelius, J. Franzen, M. Gasparik, A. Gentry, K. Heissig, G. Hernyak, T. Kaiser, G. Koufos, E. Krolopp, D. Janossy, M. Llenas, L. Meszaros, P. Mueller, P. Renne, Z. Rocek, S. Sen, R. Scott, Z. Syndlar, G. Tupal, J. v. Dam, L. Werdelin, P. S. Ungar, and R. Ziegler. Recent advances on multidisciplinary research at Rudabanya, late Miocene (MN9), hungary: A compendium. *Palaeontographia Italica*, 89:3–36, 2003.

[9] 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.

[10] G. Brightwell and P. Winkler. Counting linear extensions is #p-complete. In *Proceedings of the twenty-third annual ACM symposium on Theory of computing*, pages 175–181. ACM Press, 1991.

[11] G. Casas-Garrica. Summarizing sequential data with closed partial orders. In *Proceedings of the 2005 SIAM International Conference on Data Mining*, 2005.

[12] W. E. Donath and A. J. Hoffman. Lower bounds for the partitioning of graphs. *IBM J. Res. Develop.*, 17:422–425, 1973.

[13] M. Fortelius, A. Gionis, J. Jernvall, and H. Mannila. Spectral ordering and the biochronology of mammals. To appear, 2005.

[14] E. R. Gansner and S. C. North. An open graph visualization system and its applications to software engineering. *Software: Practice and Experience*, 30(11):1203–1233, 2000.

[15] A. Gionis, T. Kujala, and H. Mannila. Fragments of order. In *KDD 2003*, 2003.

[16] A. Gionis, H. Mannila, and A. Ukkonen. Probabilistic models for finding partial orders. In preparation, 2005.

[17] U. Halekoh and W. Vach. A bayesian approach to seriation problems in archaelogy. *Computational Statistics and Data Analysis*, 45:651–673, 2004.

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

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

[20] 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.

[21] G. Kumfert and A. Pothen. Two improved algorithms for envelope and wavefront reduction. *BIT*, 37(3):559–590, 1997.

[22] 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.

[23] D. Spielman and S. Teng. Spectral partitioning works: Planar graphs and finite element meshes. In *IEEE Symposium on Foundations of Computer Science*, pages 96–105, 1996.

[24] F. Steininger, J. Kovar-Eder, and M. Fortelius. *The middle Miocene environments and ecosystem dynamics of the Eurasian Neogene (EEDEN)*, volume 249. Courier Forschungsinstitut Senckenberg, 2004.