Nestedness and Segmented Nestedness

Heikki Mannila
HIIT
Helsinki University of Technology and
University of Helsinki, Finland
mannila@cs.helsinki.fi

Evimaria Terzi*
IBM Almaden Research Center
San Jose, CA, USA
eterzi@us.ibm.com

ABSTRACT

Consider each row of a 0-1 dataset as the subset of the columns for which the row has an 1. Then a dataset is *nested*, if for all pairs of rows one row is either a superset or subset of the other. The concept of nestedness has its origins in ecology, where approximate versions of it has been used to model the species distribution in different locations. We argue that nestedness and its extensions are interesting properties of datasets, and that they can be applied also to domains other than ecology.

We first define natural measures of nestedness and study their properties. We then define the concept of k-nestedness: a dataset is (almost) k-nested if the set of columns can be partitioned to k parts so that each part is (almost) nested. We consider the algorithmic problems of computing how far a dataset is from being k-nested, and for finding a good partition of the columns into k parts. The algorithms are based on spectral partitioning, and scale to moderately large datasets. We apply the methods to real data from ecology and from other applications, and demonstrate the usefulness of the concept.

Categories and Subject Descriptors: F.2.2 [**ANALYSIS OF ALGORITHMS AND PROBLEM COMPLEXITY**]: Nonnumerical Algorithms and Problems; H.2.8 [**DATABASE MANAGEMENT**]: Database Applications—*Data mining*

General Terms: Algorithms, Experimentation, Theory

Keywords

nestedness, 0-1 matrices, presence/absence data

1. INTRODUCTION

The analysis of 0-1 data is one of the recurring themes in data mining. One of the key issues in the area is to look for interesting concepts that would be useful to compute from such datasets.

In this paper, we consider the concept of *nestedness* of 0-1 matrices. The concept has its origins in ecology; in the study of presence/absence matrices of sites (locations) and species. There, the nestedness hypothesis states that the species found in a site with few species should be a subset of the species found in a site with more species. That is, the sets of species found in the different sets should form a chain of subsets. Since the original publication [19], the nestedness concept has been studied a lot; see, e.g., [4, 11, 14] for some recent work.[1]

In general, consider each row of a 0-1 dataset as the subset of the columns for which the row has an 1. Then a dataset is nested, if for all pairs of rows one row is either a superset or subset of the other. Equivalently, the dataset is nested if the rows and the columns can be reordered so that in the reordered matrix the 1s in each row form a contiguous segment starting from the first column.

Nestedness is of course an extreme state, and there has also been some interesting work on quantifying the degree of nestedness in a dataset [6, 5, 23, 9, 21]. See Figure 1 for examples of datasets that are completely nested, almost nested, and very far from being nested. In the figure the rows and columns have been ordered so that the degree of nestedness is easy to see; in general, it is not easy to determine the degree of nestedness.

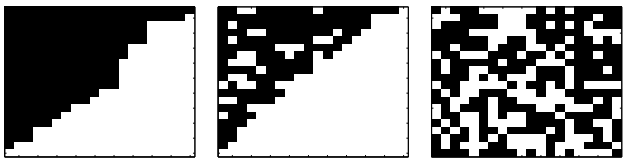


Figure 1: Examples of fully nested, almost nested and non-nested datasets. Black = 1, white = 0.

Nestedness is a concept that makes sense also for other types of datasets than ecological presence/absence data. Consider for example data about students and courses. Then, assuming that students follow a suggested study program with no electives, the set of courses that a student has taken is a sub- or a superset of the set of courses taken by another student. Deviations from this behavior tell us something about the students, the courses, or both.

*Work done while the author was at HIIT, University of Helsinki, Finland.

[1]Note that this concept of nestedness is different from the concept of nestedness of model classes that is sometimes used in model selection.

KDD'07, August 12–15, 2007, San Jose, California, USA.

As another example, consider a set of documents about a single theme, say probability. Most documents contain the basic terms "probability", "random variable", "expectation" etc., while fewer contain terms such as "limit theorems", and still fewer documents contain terms such as "martingales". The set of terms in a document talking about martingales is most likely to be a superset of the set of terms in a document about the basic concepts of probability. In general, if the collection of documents is about a single topic and there are terms of different levels of difficulty or specialty, then the dataset can be expected to be nested.

Most datasets are not nested, but identifying the degree of their nestedness can give useful insight into the processes that produced the datasets.

In some cases the whole dataset might not be nested, but when limited to a subset of the variables, it is. Starting again with an ecological example, consider presence/absence data of species in sites distributed widely over space, say from north to south. Then the dataset as a whole is not nested: the northernmost and southernmost sites might each have many species, but very few species are found in both places. Thus, the subset/superset phenomenon does not occur.

However, the set of species can be partitioned into two sets such that when projected on those sets the data is nested. In this example the sets of species could be the ones that are prevalent in the south and those that are prevalent in the north. Such *segmented nestedness* has not, to our knowledge, been considered in the literature. See Figure 2 for an example.

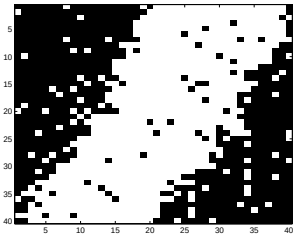


Figure 2: Example of a dataset that has segmented nestedness.

Segmented nestedness can be applied to other domains, as well. For example, in the course data we can find that the courses can be partitioned into two sets so that on those sets the data is nested; this would correspond to two different curricula. In the document example we could find that the terms can be divided into two sets in such a way that the use of the terms from both sets fall into a nested pattern.

In this paper we consider the concepts of nestedness and segmented nestedness, their use in data analysis, and the algorithmic properties of the concepts. We argue that nestedness and its extensions are interesting properties of datasets that can be applied also to domains other than ecology. We study the algorithmic properties of computing how far from being nested a dataset is. We then define the concept of k-nestedness: a dataset is (almost) k-nested if the set of columns can be partitioned to k parts so that each part is (almost) nested. We consider the algorithmic problems of computing how far a dataset is from being k-nested, and for finding a good partition of the columns into k parts. The algorithms are based on spectral partitioning. We apply the methods to real data from ecology and from other applications, and demonstrate the usefulness of the concept.

The rest of this paper is organized as follows. In Section 2 we give the definitions of almost nestedness and segmented nestedness. Section 3 gives the algorithms, and Section 4 the empirical results. Section 5 discusses related work, and Section 6 is a short conclusion.

2. PROBLEM DEFINITION

2.1 Nestedness

Consider an $n \times m$ 0-1 matrix M. In an ecological application, the rows could correspond to sites and columns to species. In a course enrollment data, the rows would correspond to students and the columns to courses. We denote the ith row of M by M_i and the jth column of M by M^j. The rows and the columns of matrix M have also set interpretations. That is, M_i, except for being a 0-1 vector, is also used to denote the subset of the species that appear in the i-th site. We use vector and set interpretation of the rows and columns of M interchangeably.

DEFINITION 1. *An $n \times m$ 0-1 matrix M is* fully nested *if for any two rows i and j we have $M_i \cap M_j \in \{M_i, M_j\}$.*

That is, any two rows of a fully nested matrix have a subset-superset relationship. Alternatively, a matrix M is fully nested if there is a permutation of the rows of M such that in the permuted matrix M^π, $M^\pi{}_i \supseteq M^\pi{}_j$ for every $i < j$. From the definition above, it is obvious that checking if a matrix is fully nested can be done in polynomial time.

In practice, the input matrices are not expected to be fully nested. Therefore, we need to define a measure of how far a 0-1 matrix is from being fully nested. We do so by looking at the minimum number of 0s that need to be transformed into 1s so that the matrix M becomes nested and we denote this by $\mathcal{N}(M)$. Therefore, for a fully nested matrix $\mathcal{N}(M) = 0$. We loosely call matrices with non-zero, but small values of $\mathcal{N}$, *almost nested*. Next, we formally define the corresponding optimization problem that we call the MINIMUM NESTEDNESS AUGMENTATION (or MNA).

PROBLEM 1 (MINIMUM NESTEDNESS AUGMENTATION). *Given a 0-1 matrix M, find the minimum number of 0s that have to be transformed into 1s so that M becomes fully nested.*

We have the following.

PROPOSITION 1. *The* MINIMUM NESTEDNESS AUGMENTATION *problem is NP-complete.*

The hardness proof of the problem is easy if we consider the graph-theoretic interpretation of the MNA problem. The input matrix M defines a bipartite graph $G(M) = (B, S, E)$. Every node $b \in B$ corresponds to a row in M and every node $s \in S$ corresponds to a column in M. There exists an undirected edge between b and s if and only if $M(b, s) = 1$. Therefore, $|B| = n$, $|S| = m$ and $|E|$ is the number of 1s in M.

A bipartite graph (B, S, E) is a *chain graph* [26] if there is a bijection $\pi : \{1, \ldots, |B|\} \to B$ (an *ordering* of B) such that $\Gamma(\pi(1)) \supseteq \Gamma(\pi(2)) \supseteq \ldots \supseteq \Gamma(\pi(|B|))$, where Γ is a function that maps a node to its neighbors.

The Minimum Chain Completion problem is as follows. Given a bipartite graph $G(M)$, find the minimum set of edges F that need to be added in $G(M)$ such that the bipartite graph $(B, S, E \cup F)$ is a chain graph. This problem is NP-hard [26]. It is easy to see that the Minimum Chain Completion problem is equivalent the Minimum Nestedness Augmentation problem, and hence Mna is also NP-hard. It is trivially in NP.

The equivalence of the matrix and graph formulations also implies the following result.

Proposition 2. *For any matrix M we have $\mathcal{N}(M) = \mathcal{N}(M^T)$, where M^T denotes the transpose of M.*

Lemma 1 in [26] states that a bipartite graph is a chain graph if and only if it does not contain a pair of edges that do not share any endpoints (independent edges). We can restate this lemma in terms of the 0-1 matrix M as follows.

Lemma 1. *A 0-1 matrix M is fully nested if and only if it does not contain any submatrix of the form*

$$\begin{pmatrix} & \vdots & & \vdots & \\ \dots & 0 & \dots & 1 & \dots \\ & \vdots & & \vdots & \\ \dots & 1 & \dots & 0 & \dots \\ & \vdots & & \vdots & \end{pmatrix}.$$

We call such submatrices *switch boxes*. We show in Section 3 how we use Lemma 1 to design a greedy algorithm for Mna.

Note that Mna considers only transformations of 0s into 1s, and charges for each such transformation. The reason for this is that in the paleontological/ecological datasets there is high certainty associated with 1s and low certainty associated with 0s. In terms of problem definitions, we can generalize Mna so that we allow changes of 1s into 0s as well. We call this generalized version of Mna, Bidirectional Mna problem (or BMna), and we formally define it as follows.

Problem 2 (BMna). *Given a 0-1 matrix M, find the minimum number of 0s or 1s that have to be transformed into 1s or 0s, respectively, so that M becomes fully nested.*

We denote by $\mathcal{B}(M)$ the cost of the optimal solution of the BMna problem for input matrix M. The complexity of the BMna problem is unknown (see [17]). Although the focus of the paper is on the Mna problem, we also show how algorithms for Mna can be adopted to solve the BMna problem as well. It is rather easy to see the following straightforward relationship between the optimal solutions to the Mna and BMna problems.

Proposition 3. *For a 0-1 matrix M we have*

$$\mathcal{N}(M) \geq \mathcal{B}(M).$$

2.2 Segmented nestedness

In this subsection we consider the problem of finding a good partition of the columns of a matrix so that the projections of the dataset to the parts are (almost) nested.

Consider the 0-1 matrix shown in Figure 2. Clearly, this matrix is not almost nested. However, there are evidently two almost nested submatrices that are induced by the first and the last 20 columns of the input matrix.

As mentioned in the introduction, such matrices motivate the definition of the *segmented* version of the Mna problem.[2] In the segmented version of the problem we wish to partition columns of the matrix into k parts so that the submatrices induced by each part are fully or almost nested. If $\{P_1, \dots, P_k\}$ is a partition of the columns of M into k parts, we denote by $M[P_i]$ the matrix of size $n \times |P_i|$ obtained from M by just considering the columns in P_i. We can extend the definitions of the fully and almost nested matrices for segmented nestedness.

Definition 2. *An $n \times m$ 0-1 matrix M is* fully k-nested *if there is a partition of its columns into k parts $\{P_1, \dots, P_k\}$ such that each $M[P_i]$ is fully nested.*

An immediate consequence of the above definition is the following.

Proposition 4. *For integers k_2, k_1 with $k_2 > k_1$, if a 0-1 matrix M is fully k_1-nested, then it is also fully k_2-nested.*

The matrices that appear in practice are not expected to be fully k-nested, but rather *almost k-nested*. That is, the submatrices $M[P_i]$ are almost nested.

For input matrix M, the segmented version of the Mna problem asks for a partition $\{P_1, \dots, P_k\}$ of the columns of M such that the total number of conversions of 0s to 1s in the $M[P_i]$'s is minimized. We call this problem the k-Mna problem and we formally define it as follows.

Problem 3 (k-Mna). *Given a 0-1 matrix M and an integer k, find a partition of the columns of M into k parts $\{P_1, \dots, P_k\}$, such that*

$$\mathcal{N}_k(M) = \sum_{i=1}^{k} \mathcal{N}(M[P_i])$$

is minimized.

Therefore, fully k-nested datasets have $\mathcal{N}_k(M) = 0$, while almost k-nested datasets have small values of $\mathcal{N}_k(M)$. Figure 2 is an example of a dataset M that has high value of $\mathcal{N}(M)$ but a small value of $\mathcal{N}_2(M)$.

Proposition 5. *For a 0-1 matrix M and integers k_1, k_2 such that $k_2 > k_1$ we have $\mathcal{N}_{k_1}(M) \geq \mathcal{N}_{k_2}(M)$.*

Since the k-Mna is a generalization of the Mna problem, k-Mna is also NP-hard.

Instead of looking for a partition of the columns that defines nested submatrices one can alternatively define the problem of removing the minimum number of rows and columns from the input matrix, so that the remaining (maximal) submatrix is almost nested. By using the results of [25] one can show that this variation of the segmented nestedness problem is also NP-hard. In the case where we are restricted in removing only rows or only columns of the input matrix, the problem becomes solvable in polynomial time. The relationship between these alternatives and the k-Mna problem is the following: while k-Mna looks for groups of columns (or rows) so that there is a nesting structure within each group,

[2]The term segmented comes from the definition of "Segmentation Problems" given in [15].

the aforementioned variants look for removal of columns and rows so that a nesting structure appears. Therefore, if one thinks of the k-MNA problem as an analogue for clustering, then the above alternatives correspond to problems like outlier detection or finding a single cluster.

We will also refer to the segmented version of Problem 2 as the k-BMNA problem. The problem definition and the corresponding properties are in accordance to those of Problem 3 and thus omitted. For input matrix M, we use $\mathcal{B}_k(M)$ to denote the optimal solution of the k-BMNA problem for M.

3. ALGORITHMS

3.1 Algorithms for estimating nestedness

In this section we consider algorithms for estimating $\mathcal{N}(M)$ for a matrix M. As the task is NP-hard, we cannot hope for an exact solution; rather, we will compute scores $\widehat{\mathcal{N}}(M)$ that are upper bounds for $\mathcal{N}(M)$.

Two simple heuristics have been used in the ecology literature for the nestedness and related problems [6, 5]. The first algorithm is `RowSum`. For input matrix M, the algorithm counts the number of 1s in each row and reorders the rows of M in decreasing order of their row sums. The algorithm then converts the necessary 0s to 1s so that the final matrix becomes fully nested. The complexity of the algorithm is $O(mn + n \log n)$. The following example shows the cost of the solution output by the `RowSum` algorithm can be arbitrarily bad compared to the cost of the optimal solution.

EXAMPLE 1. *Consider the $(p+2) \times (3m)$ matrix M that has the following structure*

$$M = \begin{pmatrix} 0 & \dots & 0 & 1 & \dots & 1 & 1 & \dots & 1 \\ 1 & \dots & 1 & 1 & \dots & 1 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & \dots & 1 & 1 & \dots & 1 & 0 & \dots & 0 \\ 1 & \dots & 1 & 0 & \dots & 0 & 1 & \dots & 1 \end{pmatrix}$$

The `RowSum` *algorithm can keep the order of the rows of M unchanged, causing a cost of $2m + mp$. The cost of the optimal solution would be just $2m$, achieved by moving the last row of M on top. Therefore, the performance ration of* `RowSum` *is $\frac{2m}{2m+mp}$, which goes to 0 as p grows.*

Some straightforward alternatives to `RowSum` can be the `ColSum` and the `BestSum` algorithms. The `ColSum` is exactly the same as the `RowSum` but it operates on the columns of M. That is, the columns of M are rearranged in decreasing order of their column sums and the necessary 0s are converted into 1s so that the final matrix is fully nested. For given input matrix M, the `BestSum` algorithm is the best of `RowSum` and `ColSum` for the given input. Both these algorithms can give solutions with cost arbitrarily bad with respect to the optimal with input the matrix given in Example 1.

Some other algorithms for quantifying nestedness have also been considered in the ecological literature (see, e.g., [23, 9, 21]), but they are fairly heuristic and difficult to analyze.

In the rest of the section we give a simple greedy algorithm for the MNA problem. We also show how we can adopt this algorithm to solve the BMNA problem as well.

Consider an input matrix M of size $n \times m$. By Lemma 1, M is fully nested if it does not contain any switch boxes. Given this, we can design a greedy algorithm that switches 0s into 1s one at a time. At every step, the algorithm converts the 0 that participates in the largest number of distinct switch boxes. Consider entry (i, j) of the matrix. The *coverage* $C(i, j)$ of entry (i, j) is the number of switch boxes this entry participates in. By converting the value in (i, j) from 0 to 1, all the $C(i, j)$ switch boxes are resolved.

After each conversion, the algorithm recalculates the coverages of the remaining 0s in the matrix and proceeds in the next greedy step. Algorithm 1 gives an outline of the `Greedy` algorithm.

Algorithm 1 The `Greedy` algorithm for computing $\widehat{\mathcal{N}}(M)$.

```
1: Input: An n × m 0-1 matrix M.
2: Output: N̂(M), an approximation of N(M).
3: N̂(M) = 0;
4: for i = 1 to m do
5:    for j = 1 to n do
6:       C(i, j) = coverage of entry (i, j) with M(i, j) = 0;
7:    end for
8: end for
9: while M is not fully nested do
10:   (i, j) = the entry with the largest coverage;
11:   M(i, j) = 1
12:   N̂(M)++
13:   update the coverages;
14: end while
15: return N̂(M)
```

The running time of the `Greedy` algorithm is $O(mn + IT_1T_2)$, where I is the number of iterations of the `while` loop, T_1 the time required for finding the maximum coverage, and T_2 the time required for updating the coverages.

By using a heap, T_1 is $O(1)$ and T_2 is $O(mn \log(mn))$. When entry (i, j) is converted from 0 to 1, we only have to make the following updates: (i) decrease by 1 the coverages of entries (v, u) with $M_{vu} = 0$, $M_{iu} = 1$, and $M_{vj} = 1$; (ii) increase by 1 the coverages of entries (i, u) and (v, j) with $M_{vu} = 1$, $M_{iu} = 0$, and $M_{vj} = 0$. These are due to (i) switch boxes that (i, j) participated in and that are removed, and (ii) switch boxes that are created by the conversion.

As the maximum number of iterations is $O(mn)$, the total time is $O((mn)^2 \log(mn))$. In practice the number of iterations is much smaller than $O(mn)$, indicating that the method scales reasonably well.

The same greedy principle can be used for solving the BMNA problem. The corresponding `Bi-Greedy` algorithm is essentially the same as the `Greedy` algorithm above, with the only difference that since both 0s are transformed into 1s and 1s into 0s, the coverages of both 0 and 1 entries are computed in the beginning and updated in every iteration. Whenever entry (i, j) is converted from 0 to 1 the following updates need to be done by the `Bi-Greedy` algorithm: (i) decrease by 1 the coverages of entries (v, u) with $M_{vu} = 0$, $M_{iu} = 1$, and $M_{vj} = 1$; (ii) increase by 1 the coverages of entries (v, u) with $M_{vu} = 1$, $M_{iu} = 0$, and $M_{vj} = 0$; (iii) increase by 1 the coverages of entries (i, u) and (v, j) with $M_{vu} = 1$, $M_{iu} = 0$, and $M_{vj} = 0$; (iv) decrease by 1 the coverages of entries (i, u) and (v, j) with $M_{vu} = 0$, $M_{iu} = 1$, and $M_{vj} = 1$. The complementary updates are made when

entry (i, j) is converted from 1 to 0. The running time of the `Bi-Greedy` algorithm is of the same order as the running time of the `Greedy` algorithm discussed above. Note that the algorithm is designed so that at most one conversion is allowed per entry.

3.2 Algorithms for estimating segmented nestedness

In this section we give algorithms for the k-MNA problem. The algorithms for the k-BMNA problem are easily developed using the same principles, and thus their discussion is omitted. Remember that the task is to find a partition $\{P_1, \ldots, P_k\}$ of the columns of the matrix M so that the projections $M[P_i]$ of M to these columns have a small value of $\mathcal{N}(M)$. Again, as the problem is NP-hard, we have to be content with producing upper bounds $\widehat{\mathcal{N}}_k(M)$ for $\mathcal{N}_k(M)$.

We approach the problem as a clustering problem for the columns of the input matrix M. When should two columns be placed to the same group in the partition? Obviously, if the columns a and b are similar, i.e., they tend to have 1s in the same rows, then it makes sense to have them in the same part P_i. On the other hand, if a and b are independent or negatively correlated, then they should be in different parts of the partition.

This simple observation leads to an algorithm for computing an approximation to $\mathcal{N}_2(M)$: define a similarity notion between columns, use a spectral bisection method [22] to obtain the two parts of the partition, and then compute the score for both parts using the `Greedy` algorithm. By recursive applications of the algorithm we obtain a way of approximating $\mathcal{N}_k(M)$ as well.

We first define two simple and intuitive similarity functions between the columns of the input matrix M. Then, we combine these functions with a simple spectral bisection algorithm to solve the 2-MNA problem.

One straightforward measure of similarity that serves our purposes is the *correlation similarity*. We define the correlation similarity between two columns M^a and M^b as

$$\mathrm{CorrS}(M^a, M^b) \quad = \quad 1 + \rho_{ab}, \tag{1}$$

where ρ_{ab} is the Pearson correlation between columns a and b. As ρ_{ab} takes values in the range $[-1, 1]$, CorrS takes values in $[0, 2]$. Value 2 is obtained if the columns are identical, and value 0 when they are perfectly anticorrelated.

Alternatively, we use the following *inclusion similarity* between columns M^a and M^b:

$$\mathrm{InclS}(M^a, M^b) \quad = \quad \frac{M^a \cap M^b}{\min\{|M^a|, |M^b|\}}. \tag{2}$$

The inclusion similarity captures our intuition that two columns need to be in the same cluster if the one includes the other. For example, when column a is included in column b (or vice versa), then $\mathrm{InclS}(M^a, M^b) = 1$, and thus columns a and b are considered very similar. Similarly, the inclusion similarity of two non-intersecting columns is zero.

Spectral algorithms are important tools for a wide range of problems such as, solving linear systems [20], ordering problems [1, 16], data clustering [18, 10] and many more. Before describing the spectral bisection algorithm we use, we give some background for the spectral method in general.

Consider a weighted undirected graph $G = (V, E)$ where each $(i, j) \in E$ has a weight w_{ij}. Let A be the matrix with $A(i, j) = w_{ij}$ if $(i, j) \in E$ and $A(i, j) = 0$ otherwise. The *Laplacian* of A is defined to be the symmetric and zero-sum matrix $\mathcal{L} = D - A$, where D is a diagonal matrix whose (i, i)-th entry is $d_i = \sum_{(i,j)\in E} w_{ij}$. The eigenvalues of $\mathcal{L}$ are real and nonnegative, and the smallest eigenvalue is 0 (corresponding to the eigenvector of all 1s). The eigenvector v that corresponds to the second smallest eigenvalue of $\mathcal{L}$ is also known as the *Fiedler vector*.

We consider the graph A, whose nodes are the columns of the input matrix M. There is an edge between all pairs of nodes, and the weight of the edge (a, b) is $\mathrm{CorrS}(M^a, M^b)$ (Equation (1)) or $\mathrm{InclS}(M^a, M^b)$ (Equation (2)). We form the Laplacian matrix $\mathcal{L}_A$ of the graph A and compute the Fiedler vector v. Then v is a vector with m components.

Each element v_a of v defines a partition by $P_1 = \{b | v_b \leq v_a\}$ and $P_2 = \{b | v_b > v_a\}$. For each v_a, we evaluate $\widehat{\mathcal{N}}(M[P_1]) + \widehat{\mathcal{N}}(M[P_2])$ and return the partition with the smallest value of the sum.

The algorithm can be augmented by local moves. That is, we start from the initial partition given by the spectral bisection method, and then repeatedly search for columns that can be moved from one part to the other so that the score improves.

If $k > 2$, then we recursively call the algorithm on each of the parts to check which one should be divided further. The method, called `Partition`, is described also in Algorithm 2.

Algorithm 2 The `Partition` algorithm for computing $\widehat{\mathcal{N}}_k(M)$.

1: **Input:** An $n \times m$ 0-1 matrix M, and an integer $k > 1$.
2: **Output:** A partition of the columns of M to k parts and an upper bound $\widehat{\mathcal{N}}_k(M)$ for $\mathcal{N}_k(M)$.
3: Compute the similarity graph A, with entries $A(a, b) = \mathrm{CorrS}(a, b)$.
4: Form the Laplacian $\mathcal{L}_A$ of A, and compute the Fiedler vector v.
5: Find the column a such that the bisection $P_1 = \{b | v_b \leq v_a\}$ and $P_2 = \{b | v_b > v_a\}$ has the best score of $\widehat{\mathcal{N}}(M[P_1]) + \widehat{\mathcal{N}}(M[P_2])$;
6: **if** $k > 2$ **then**
7: Recursively call `Partition` on the restriction of M to P_1 and to P_2 to check which gives the larger decrease in the $\widehat{\mathcal{N}}_2(M)$ score
8: **end if**
9: Do local moves: for each column a, test whether it can be moved to another part of the partition so that the score improves

Every call of the `Partition` algorithm requires time $O(T_\lambda + mT_p)$, where T_λ is the time needed for the eigenvalue computation and T_p is the time required for the approximation of $\mathcal{N}$ at step 5 of the algorithm. The dominating factor is the computation of $\widehat{\mathcal{N}}$ for $k = 1$ for both sets of m different partitions, resulting a total time complexity of $O(m^3 n^2 \log(mn))$. In practice the method scales to datasets of moderate size, as the above bound is a worst-case one.

We will compare the performance of the `Partition` algorithm with the CorrS and InclS similarity functions with two straightforward heuristics, the `kmeans` and the `InclS-kmeans`. The `kmeans` algorithm performs a clustering of the columns considering each column as an observation consisting of n attributes (where n is the number of rows of the input matrix). We use the Hamming distance func-

tion as the optimization criterion for the `kmeans` algorithm. The `InclS-kmeans` again clusters the columns of the input matrix, this time using as input the similarity matrix induced by the InclS function. In this case, the centers of the k-means procedure are bound to be from the input points, since only the similarity matrix is given as input.

3.3 Selecting the value of k

The concept of k-MNA has the parameter k that can vary. For any dataset M, we have $\mathcal{N}_k(M) \geq \mathcal{N}_{k+1}(M)$, as the ability to divide the columns into $k+1$ sets instead of k sets will make it easier to have small number of nestedness violations. At the limit, when $k = m$, the number of columns in M, we have $\mathcal{N}_m(M) = 0$. So how should a good value of k be chosen?

This is, of course, a classical example of the model selection problem. Different solutions abound: one can use MDL, BIC, AIC, cross-validation, etc. (see, e.g., [12] for a description of some of the methods).

Here we use a fairly simple alternative: we monitor the change of the score $\widehat{\mathcal{N}}_k(M)$ as a function of k; when the score flattens, the correct value of k is reached. This, of course, is a heuristic approach, but seems to be sufficient. We also compare the change in the score with the change in the score $\widehat{\mathcal{N}}_k(M^\pi)$, where M^π is a dataset obtained from M by permuting each column of M independently at random.[3] That is, we test whether the score on real data drops clearly faster than for random data with the same density. Other methods of selecting k are left for further study.

4. EXPERIMENTS

In this section, we give experimental evidence of the utility of the nestedness concept. For this, we use both synthetic and real datasets; the latter come from a variety of application domains like ecology, paleontology and students' course enrollment data.

4.1 Synthetic datasets

To test the behavior of the algorithms we generated synthetic data as follows. For given n and m we first generate a fully nested 0-1 matrix by first sampling row counts r_i uniformly between 0 and m, and sorting them so that $r_i \geq r_j$ for every $i < j$. Then we generate rows so that row i has 1s in its first r_i columns and 0s in the other columns. Let such a matrix be H, and let H^1 be all the cells in H that have value 1 and H^0 all the cells that have value 0. Given H we generate our synthetic matrices by altering entries of H^1 from 1 to 0, and entries of H^0 from 0 to 1. An entry in H^1 keeps its original value with probability $1-p$, and it is converted to 0 with probability p. Similarly, the entries in H^0 are switched to 1 with probability q and they maintain their original value with probability $1-q$. Thus, p is the probability of an 0-entry in H^1 and q the probability of an 1-entry in H^0. The interesting parameter values are ones where $p > q$ and q is fairly close to 0; this is due to the asymmetry of 0s and 1s mentioned earlier. Figure 3 shows the relative performance of the three algorithms `Greedy`, `RowSum` and `ColSum` algorithms for synthetic datasets generated as above. Note that when $p = 0$ and $q = 0$, the generated matrices are fully nested and all algorithms find the optimal solution that has cost 0. In Figure 3 we compare the number of flips done by the different algorithms for datasets generated with $q = 0.05$ and $p \in [0.1, 0.95]$. Among these three algorithms we can clearly see that the `Greedy` algorithm consistently outperforms the other two, and the difference is fairly high especially for smaller values of p. In the same figure we additionally report the cost of the `Bi-Greedy` algorithm using cost function $\mathcal{B}$ instead of $\mathcal{N}$. Note that although the results of `Bi-Greedy` are not comparable with the results of the other algorithms, they are given here for completeness.

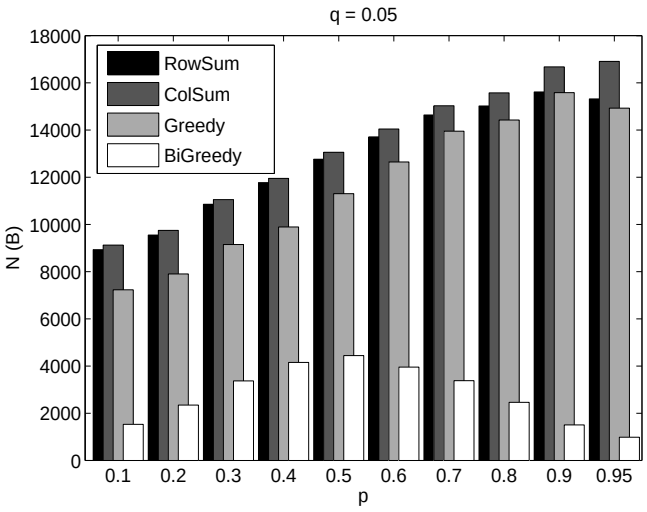


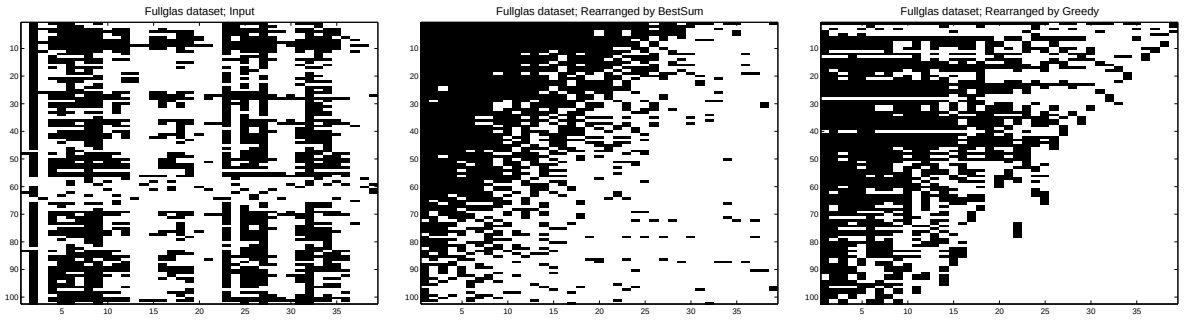
Figure 3: Comparative performance of algorithms for MNA problem on synthetic datasets for $q = 0.05$. x-axis: parameter p; y-axis: the quantity $\widehat{\mathcal{N}}(M)$ of the input M for `RowSum`, `ColSum` and `Greedy` algorithm and the quantity $\widehat{\mathcal{B}}(M)$ for `Bi-Greedy` algorithm.

4.2 Habitat datasets

We have also tested the relative performance of the different algorithms for MNA by using a set of real datasets available by AICS Research Inc, University Park, New Mexico, and The Field Museum, Chicago. The datasets are available online[4] and they have been used for a wide range of ecological studies [3, 9, 7, 2, 23]. The collection contains 280 datasets in the form of presence/absence matrices, that cover a wide variety of taxa (mammals, bats, land and freshwater birds, reptiles and amphibians, fish, terrestrial arthropods, terrestrial mollusks, plants and other miscellaneous species). Figure 4 serves as a summary of comparisons between the `Greedy` algorithm and the best of the `RowSum` and `ColSum` algorithms (`BestSum`). For the purposes of the experiment, we group the datasets with respect to their densities (number of 1's in the matrix divided by the size of the matrix). For each density range, we count the number of datasets in the range for which `Greedy` performs better, equal or worse than `BestSum` algorithm. For most datasets with small density the `Greedy` algorithm performs considerably better than `BestSum`, and throughout the datasets `Greedy` is only seldom worse than `BestSum`. Figure 5 shows

[3] Note that in a fully nested dataset there are no switch boxes; thus we cannot use the swap randomization method [13].

[4] `http://www.aics-research.com/nestedness/`



(a) Fulglas dataset; Input matrix (b) Fulglas dataset; Rearrangement by `BestSum` (c) Fulglas dataset; Rearrangement by `Greedy`

Figure 5: Rearrangements of the Fulglas dataset matrix

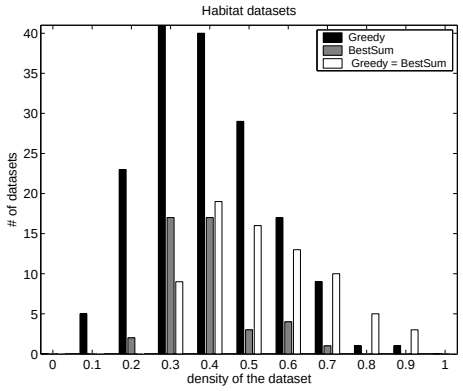


Figure 4: The performance of the algorithms on the habitats datasets. The columns show the number of datasets for which the `Greedy` algorithm or the `BestSum` algorithm (the better of `RowSum` and `ColSum`) gives better results or whenever they are equal.

an indicative example of this performance for the `Fullglas` dataset. The dataset has 102 rows, 39 columns and 1503 non-zero entries. Figure 5(a) renders the input matrix. The matrix corresponds to presence/absence information of prairie plants (goldenrods, milkweeds, and legumes) in 102 prairie fragments in Iowa and Minnesota. The `BestSum` algorithm makes 2026 conversions, while the `Greedy` only needs 959. The rearranged matrices output by the two algorithms are shown in Figures 5(b) and 5(c). The example above illustrates the philosophy of the `Greedy` algorithm. Instead of a rearrangement that brings rows with larger row sum before rows with smaller row sum, it rather prefers bringing some sparse rows on the top for the sake of creating a matrix with a clear triangular form. Notice that the rearranged matrices output by the `Greedy` algorithm have no 1s in their right lower part.

4.3 Segmented nestedness

In this section we test the algorithms for computing $\widehat{\mathcal{N}}_k(M)$. There are two problems to consider: whether the algorithm is able to recognize the cases where the dataset has a clear k-nested structure, and whether there are clear cases of k-nested structure in real data. We evaluate the algorithms for approximating $\mathcal{N}_k(M)$ via experiments on both synthetic and real datasets. We generate the synthetic datasets using the same method described in Section 4.1. For given m and n, we use the method described in Section 4.1 to generate two matrices M_1 and M_2 of sizes $n \times m_1$ and $n \times m_2$, such that $m_1 + m_2 = m$. By random permutations of the rows and the columns of M_1 and M_2, we obtain the permuted matrices M_1' and M_2'. The final matrix M is obtained by concatenating the i-th row of M_1' with the i-th row of M_2'. The matrices generated in such a way are almost 2-nested, i.e., they have a small value of $\mathcal{N}_2(M)$. We use such matrices as input to the bisection algorithms we described in Section 3.2. The same technique can be used to generate almost k-nested datasets for any k. Figure 6 shows the performance of the `Partition` algorithm using the CorrS and InclS similarity functions and the effect of the local moves. We compare the performance of these algorithms with the performance of two straightforward heuristics, the `kmeans` and the `InclS-kmeans` (see Section 3.2). As a baseline, we also compare the quantity $\widehat{\mathcal{N}}_2$ returned by the algorithms to the ground truth, i.e., the value determined by the generation process. As shown in Figure 6 the `Partition` algorithm for both the similarity measures defined in Section 3.2 outperforms the other heuristics. This performance is independent of how balanced the sizes of the planted nests are (Figure 6(a)) or the noise level of the dataset (Figure 6(b)). The cost achieved by the `Partition` algorithm is quite close to the ground truth. The effect of the local moves step becomes more obvious for datasets generated to have components of unequal sizes. In many cases, the $\widehat{\mathcal{N}}_k$ score of the `Partition` algorithm with local moves and the `InclS` similarity function is better than the score obtained by the ground truth partition.

The next experiment investigates the selection of the correct value of k. We generated datasets that have 1, 2, or 3-nested structure, and tested the values of $\widehat{\mathcal{N}}_k$ and $\widehat{\mathcal{B}}_k$ for $k = 1, 2, 3, 4, 5$. Table 1 shows the results, together with the results for the permuted version of the datasets. We observe that for 1-nested data the score $\widehat{\mathcal{N}}_k$ stays about the same for all values of k, while for 2-nested data the score $\widehat{\mathcal{N}}_k(M)$ drops a lot when moving from $k = 1$ to $k = 2$, and for 3-nested data the score $\widehat{\mathcal{N}}_k(M)$ drops a lot when moving from $k = 1$ to $k = 2$ and $k = 3$. I.e., the drop rate can be used as an indication of the correct value of k. The same

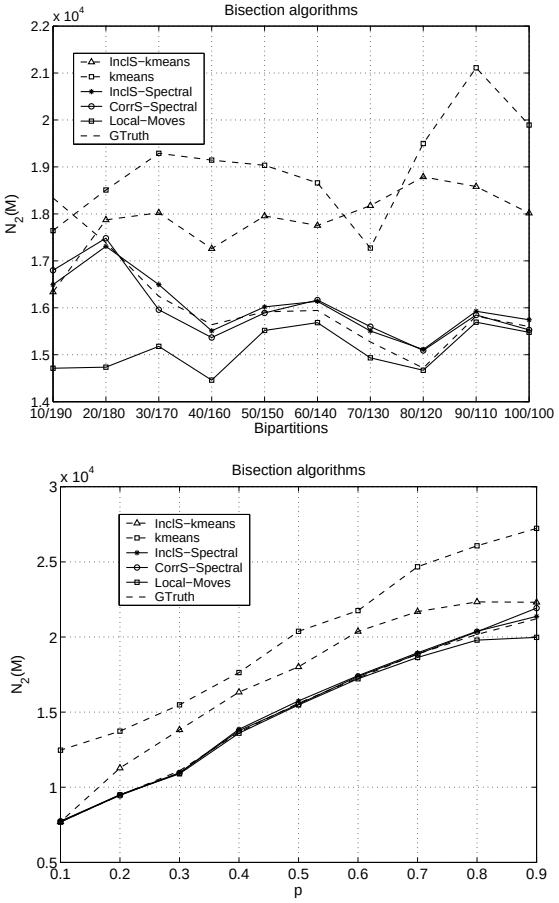


Figure 6: Comparative performance of bisection algorithms for approximating $\mathcal{N}_2$. Upper panel: Data set generated with $p = 0.5$ and $q = 0.01$ (see text), with two matrices of 200 rows and varying number columns in two nested parts. x-axis: the sizes of the nested parts; y-axis: $\widehat{\mathcal{N}}_2$. Lower panel: Data set generated with $q = 0.01$ and varying p; 200 rows, and two nested parts of 100 columns. x-axis: the value of p; y-axis: $\widehat{\mathcal{N}}_2$. Algorithms: `kmeans` and the `InclS-kmeans`: see Section 3.2; `InclS-Spectral` and `CorrS-Spectral`: the `Partition` algorithm with the `InclS` and `CorrS` similarity metrics, but without local moves; `Local-Moves`: the `Partition` algorithm with the `InclS` similarity metric and with local moves; `GTruth`: the ground truth from the data generation process.

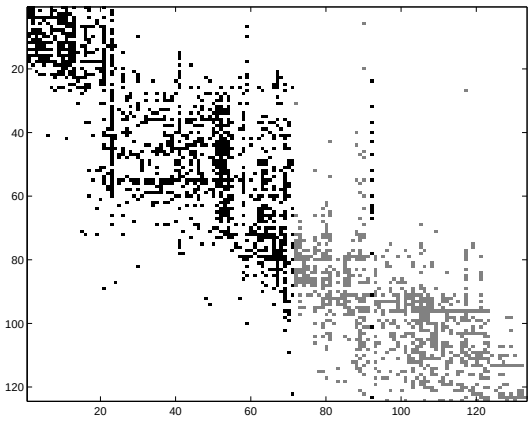
patterns are observed even more clearly for the values of $\widehat{\mathcal{B}}_k$. (The drop in the score of the permuted datasets tells us something about how large changes should be expected on the average.) Table 2 shows the same type of results

true k=1						
k	$\widehat{\mathcal{N}}_k(M)$	$\widehat{\mathcal{N}}_k(M^\pi)$	std	$\widehat{\mathcal{B}}_k(M)$	$\widehat{\mathcal{B}}_k(M^\pi)$	std
1	6643	18717	67	2194	9861	25
2	5263	18523	45	2170	9810	54
3	4202	17383	330	2156	9741	40
4	3944	16731	587	2139	9661	53
5	3789	15954	354	2121	9561	59
true k=2						
k	$\widehat{\mathcal{N}}_k(M)$	$\widehat{\mathcal{N}}_k(M^\pi)$	std	$\widehat{\mathcal{B}}_k(M)$	$\widehat{\mathcal{B}}_k(M^\pi)$	std
1	12790	18107	51	6021	10162	55
2	7163	17814	68	2212	10134	46
3	6270	17395	246	2200	10079	12
4	5744	16037	92	2186	10023	59
5	5481	15353	225	2166	9906	51
true k=3						
k	$\widehat{\mathcal{N}}_k(M)$	$\widehat{\mathcal{N}}_k(M^\pi)$	std	$\widehat{\mathcal{B}}_k(M)$	$\widehat{\mathcal{B}}_k(M^\pi)$	std
1	14622	18768	73	7405	10280	49
2	10442	18532	56	4525	10256	32
3	6579	17508	194	2195	10216	40
4	6248	16816	524	2183	10077	10
5	5842	15824	44	2155	10088	37

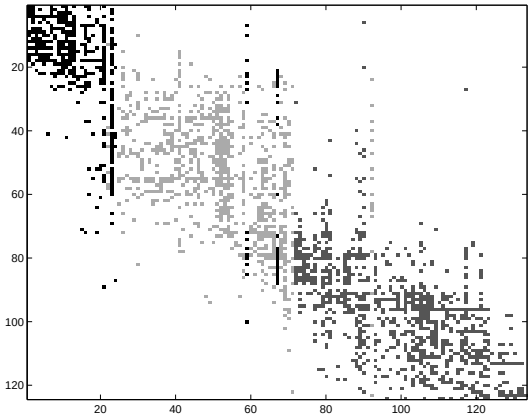
Table 1: The k-nestedness count on generated datasets with 1, 2, or 3 components. True k: the number of components in the data; k: the input parameter given to the `Partition` algorithm; $\widehat{\mathcal{N}}_k(M)$ ($\widehat{\mathcal{B}}_k(M)$): the MNA (BMNA) score of the partition; $\widehat{\mathcal{N}}_k(M^\pi)$ ($\widehat{\mathcal{B}}_k(M^\pi)$): the MNA (BMNA) score for the dataset where each column has been permuted independently; std: standard deviation of $\widehat{\mathcal{N}}_k(M^\pi)$ and $\widehat{\mathcal{B}}_k(M^\pi)$. The size of the generated data is 200 rows and 200 columns. The parameters are $p = 0.1$ and $q = 0.01$ (see text).

for the paleontological and course datasets. We observe that the drop continues until $k = 3$ for the paleontological data and until $k = 2$ for the course data, but levels off after that. In all cases the drop is larger for the real dataset than for the permuted one. The interpretation of the results is fairly clear for both real datasets. The paleontological dataset has a 3-nested structure, and the same is true also for the course dataset. In the course dataset there are actually students from two different curricula: the names of the courses have changed, and some students have moved from one system to another. This explains why there are clearly more than one components in the data.

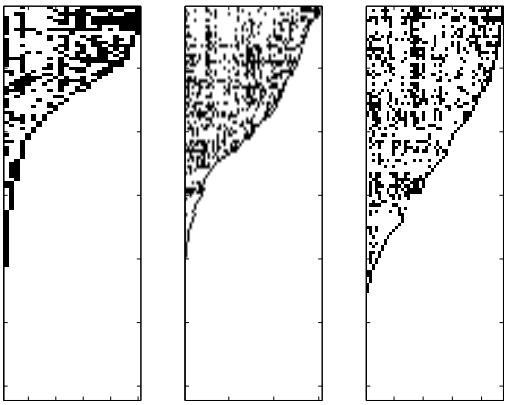
Figure 7 shows the results on the paleontological dataset. Panels (a) and (b) show the results of partitioning the columns into two or three parts so as to maximize nestedness. The dataset is drawn so that the order of the rows corresponds to the temporal order of the sites, and the order of the columns corresponds to the order of the occurrence times of the species. The sets of columns are nicely consecutive in the figure, indicating the expected result that the nested subsets of columns are temporally connected. Panel (c) shows how the subsets of the data projected to 3 parts look like, with the rows reordered so as to show maximum nestedness. We observe a quite strong nested structure in each plot.



(a) Paleo dataset; 2-partitioning of the columns, with 64 and 75 columns.



(b) Paleo dataset; 3-partitioning of the columns, with 25, 47, and 61 columns



(c) The three subsets of the columns of the paleontological dataset, with rows and columns reordered by the `Greedy` algorithm.

Figure 7: Segmented nestedness in the paleontological dataset.

Paleo data						
k	$\widehat{\mathcal{N}}_k(M)$	$\widehat{\mathcal{N}}_k(M^\pi)$	std	$\widehat{\mathcal{B}}_k(M)$	$\widehat{\mathcal{B}}_k(M^\pi)$	std
1	12156	12534	92	1749	1912	13
2	6375	10662	66	1566	1829	21
3	4674	9058	163	1397	1773	26
4	3933	7971	73	1277	1720	15
5	3505	7360	295	1181	1673	14
Course data						
k	$\widehat{\mathcal{N}}_k(M)$	$\widehat{\mathcal{N}}_k(M^\pi)$	std	$\widehat{\mathcal{B}}_k(M)$	$\widehat{\mathcal{B}}_k(M^\pi)$	std
1	105814	140310	892	20013	27676	65
2	66196	126990	345	16546	27069	85
3	60713	109660	331	14124	26831	35
4	49420	93397	1140	13842	26555	92
5	47485	86102	1201	13503	26407	109

Table 2: The k-nestedness count on real datasets. k: the input parameter given to the `Partition` algorithm; $\widehat{\mathcal{N}}_k(M)$ ($\widehat{\mathcal{B}}_k(M)$): the MNA (BMNA) score of the partition; $\widehat{\mathcal{N}}_k(M^\pi)$ ($\widehat{\mathcal{B}}_k(M^\pi)$): the MNA (BMNA) score for the dataset where each column has been permuted independently; std: standard deviation of $\widehat{\mathcal{N}}_k(M^\pi)$ and $\widehat{\mathcal{B}}_k(M^\pi)$. The data sizes are paleontological: 124×139 and course: 2401×106.

5. RELATED WORK

The concept of nestedness has its roots in ecological studies where nested subsets have been observed in presence/absence matrices of species in biotas. The main focus of the ecological studies has been applying the concept to the study of species distribution. In more computationally oriented work, the main effort has been towards defining measures of nestedness and devising randomization tests for checking whether the nestedness score of a given matrix is significant or whether it can appear by chance.

Several indices of nestedness have been proposed like for example the temperature of a matrix [3], the N_0 index [19, 24] the N_1 index [8, 24], the N_C index [24], the U index [8] and lately the discrepancy [5]. Many of these indices are rather informally defined.

Randomization tests have been developed for comparing the nestedness index of a given matrix with that of matrices generated using an underlying generative model. The models used and tested in the ecological literature include those that produce matrices with the same density, or the same row or (or and) column margins [5, 19, 24]. The randomization tests indicate whether the nestedness evaluated using one of the above indices are significant or they could have occurred by chance. The nestedness indices and the randomization methods have been subject to extensive comparisons and have been applied to a wealth of ecological datasets; see [23] for a review.

The study of the computational problems related to the computation of the nestedness indices has been limited. For example, [6] studies the properties of discrepancy for a given family of 0-1 matrices that have decreasing row and column sums. However, the computational problem of finding the right rearrangement of rows and columns so that the rearranged matrix has the minimum discrepancy has not, to our knowledge, been addressed before. Also, the concept of segmented nestedness and its computational analysis is new.

6. CONCLUSIONS

We have studied the concepts of nestedness and segmented nestedness, arguing that they are useful also outside the field of ecology. We defined the measures that quantify how far a dataset is from being k-nested and studied the properties of the corresponding optimization problem. For the case $k = 1$, we proposed a simple greedy algorithm that clearly outperforms existing algorithms on generated and real data. For the case $k > 1$, we showed how spectral methods can be used to obtain partitions that give good results. We evaluated the methods both on synthetic and real data. The algorithms scale to moderately sized datasets.

There are obviously many open questions. One of the most interesting ones is the selection of the correct value of k; here we were content to use a simple heuristic, but it is interesting to study the behavior of more disciplined methods. The usefulness of the concept of segmented nestedness on real data seems clear, and additional experiments on, e.g., ecological and document data are of interest.

7. REFERENCES

[1] Atkins, J. E., Boman, E. G., and Hendrickson, B. A spectral algorithm for seriation and the consecutive ones problem. *SIAM J. Comput. 28*, 1 (1998), 297–310.

[2] Atmar, W., and Patterson, B. Nestedness temperature calculator.

[3] Atmar, W., and Patterson, B. On the measure of order and disorder in ecological communities on archipelagos. *Oecologia 96* (1993), 539–547.

[4] Bascompte, J., Jordano, P., Melian, C., and Olesen, J. The nested assembly of plant-animal mutualistic networks. *Proceedings of the National Academy of Sciences 100* (2003), 9383–9387.

[5] Brualdi, R., and Sanderson, J. Nested species subsets, gaps, and discrepancy. *Oecologia 119*, 2 (1999), 256–264.

[6] Brualdi, R. A., and Shen, J. Discrepancy of Matrices of Zeros and Ones. *Electr. J. Comb. 6* (1999).

[7] Cutler, A. The dynamics of nested patterns of species distribution. *Biodiversity dynamics*.

[8] Cutler, A. Nested biotas and biological conservation: metrics, mechanisms, and meaning of nestedness. *Conserv. Biol. 5* (1991), 496–505.

[9] Cutler, A. Nested biotas and biological conservation: metrics, mechanisms, and meaning of nestedness. *Landscape and Urban Planning 28* (1994), 73–82.

[10] Ding, C. H. Q., and He, X. Linearized cluster assignment via spectral ordering. In *ICML* (2004).

[11] Donlan, C., Knowlton, J., Doak, D., and Biavaschi, N. Nested communities, invasive species and holocene extinctions: evaluating the power of a potential conservation tool. *Oecologia 145*, 3 (2005), 475–485.

[12] Duda, R., Hart, P., and Stork, D. *Pattern Recognition*. Wiley, 2001.

[13] Gionis, A., Mannila, H., Mielikäinen, T., and Tsaparas, P. Assessing data mining results via swap randomization. In *KDD* (2006), pp. 167–176.

[14] Ibanez, J., Caniego, J., and Garcia-Alvarez, A. Nested subset analysis and taxa-range size distributions of pedological assemblages: implications for biodiversity studies. *Ecological Modelling 182*, 3/4 (2005), 239–256.

[15] Kleinberg, J., Papadimitriou, C., and Raghavan, P. Segmentation problems. *Journal of the ACM* (2004), 263–280.

[16] Koren, Y., and Harel, D. Multi-scale algorithm for the linear arrangement problem, 2002.

[17] Natanzon, A., Shamir, R., and Sharan, R. Complexity classification of some edge modification problems. *Discrete Applied Mathematics 113*, 1 (2001), 109–128.

[18] Ng, A. Y., Jordan, M. I., and Weiss, Y. On spectral clustering: Analysis and an algorithm. In *NIPS* (2001), pp. 849–856.

[19] Patterson, B., and Atmar, W. Nested subsets and the structure of insular mammalian faunas and archipelagos. *Biological Journal of the Linnean Society 28*, 1-2 (1986).

[20] Pothen, A., Simon, H., and Wang, L. Spectral nested dissection. Tech. Rep. CS-92-01, 1992.

[21] Rodriguez-Girones, M. A., and Santamaria, L. A new algorithm to calculate the nestedness temperature of presence-absence matrices. *Journal of Biogeography 33*, 5 (2006), 924–935.

[22] Spielman, D. A., and Teng, S.-H. Spectral partitioning works: Planar graphs and finite element meshes. In *FOCS* (1996), pp. 96–105.

[23] Wright, D., Patterson, B. D., Mikkelson, G. M., Cutler, A., and Atmar, W. A comparative analysis of nested subset patterns of species composition. *Oecologia 113* (1998), 1–20.

[24] Wright, D., and Reeves, J. On the meaning and measurement of nestedness in species assemblages. *Oecologia 92* (1992), 257–264.

[25] Yannakakis, M. The node-deletion problem for hereditary properties is NP-complete. *J. Comput. System Sci. 20* (1980), 219–230.

[26] Yannakakis, M. Computing the Minimum Fill-In is NP-Complete. *SIAM Journal on Algebraic and Discrete Methods 2*, 1 (1981), 77–79.