

# Segmented nestedness in binary data

Esa Junttila\*

Petteri Kaski†

## Abstract

A binary matrix is *fully nested* if its columns form a *chain* of subsets; that is, any two columns are ordered by the subset relation, where we view each column as a subset of the rows indicated by the 1-entries. A binary matrix is *k-nested* if its columns can be partitioned into  $k$  pairwise disjoint blocks, each of which is fully nested. Such nested patterns are encountered, for example, in presence/absence patterns of species in ecological data.

We study the automated discovery of  $k$ -nestedness on synthetic data and real ecological data. First, we show that  $k$ -nestedness can be efficiently discovered in a noise-free setting using a polynomial-time algorithm. Second, we show that it is NP-hard to find a  $k$ -nested matrix that minimizes the Hamming distance to a given dataset. Thus, it is likely that in the presence of noise no efficient algorithm exists for discovering  $k$ -nestedness in the general case. Third, we develop and evaluate multiple heuristic algorithms for discovering  $k$ -nestedness on noisy synthetic data. The methods based on a combination of singular value decomposition and k-means++ give the best performance in terms of structure discovery and noise tolerance. Fourth, we develop an MDL-based model selection technique for assessing nestedness, and discover  $k$ -nested structure in (a) paleontological data, and (b) geographical occurrence data for mammal species in Europe.

## 1 Introduction

The automated discovery of structure in binary data is one of the basic themes in data mining. A recent instantiation of this theme is the notion of *nestedness* [14], which is useful in contexts where one is expected to witness a progressive development of attributes, such as students advancing through a curriculum from introductory to specialized courses. The concept of nestedness is frequently used, for example, in applications in ecology [18], where species found on barren areas also occur at abundant areas, which demonstrates a hierarchy of the species.

A binary matrix is *fully nested* if its columns form a *chain* of subsets; that is, any two columns are ordered by the subset relation, where we view each column as a subset of the rows indicated by the 1-entries. It follows immediately that every fully nested matrix can be transformed, by permutation of the rows and columns, into *normal form* where (a) the 1s in each column are placed in the topmost rows; and (b) the number of 1s in each column is at most the number of

1s in the column to the left of it (if any). An example is displayed in Fig. 1.

A conceptual extension of nestedness is *segmented nestedness* [14], which assumes that the dataset can be partitioned into  $k = 1, 2, \dots$  parts, each of which is independently nested. Formally, a binary matrix is *fully k-nested* if its columns can be partitioned into  $k$  pairwise disjoint submatrices, called *blocks*, each of which is fully nested. See Fig. 2 for an example.

A fully  $k$ -nested dataset is of course rarely witnessed in practice, for example, due to noise or a more rich structure present in the data. In practice one can hope to discover that the dataset is *almost k-nested*,

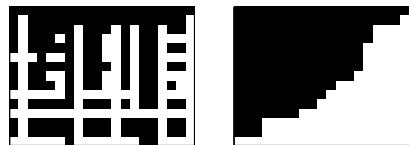


Figure 1: A fully nested matrix in plain form (left) and, after permutation of rows and columns, in normal form (right). We indicate 1s by black and 0s by white pixels.



Figure 2: A fully 3-nested matrix. We display the matrix in plain form (top), with the three fully nested submatrices grouped into consecutive columns (middle), and each of the three fully nested submatrices in normal form (bottom). Observe that the fully nested blocks in general cannot be simultaneously placed in normal form.

\*HIIT, University of Helsinki

†HIIT, Aalto University

that is to say, close to a fully  $k$ -nested matrix with respect to some metric such as the Hamming distance. Figure 3 illustrates a paleontological dataset partitioned into three almost nested blocks.

**1.1 Contributions.** In this paper we develop exact and heuristic algorithms for studying segmented nestedness in binary data. Our main findings are as follows.

First, it has been open problem whether one can find efficiently the minimum  $k$  for which a given dataset is  $k$ -nested. We answer this question in the affirmative and present an efficient algorithm by reduction to the MINIMUM PATH COVER problem in transitive directed acyclic graphs. It should be observed, however, that this algorithm is prone to overfitting in the case of noisy or more richly structured data, and hence is not particularly useful in practice.

Second, in the presence of noise we show that for a given  $k$  finding the minimum Hamming distance of a given dataset to a fully  $k$ -nested binary matrix is NP-hard. It is thus unlikely that there would exist an efficient general-purpose algorithm for discovering almost  $k$ -nestedness.

Third, we investigate the performance of a num-

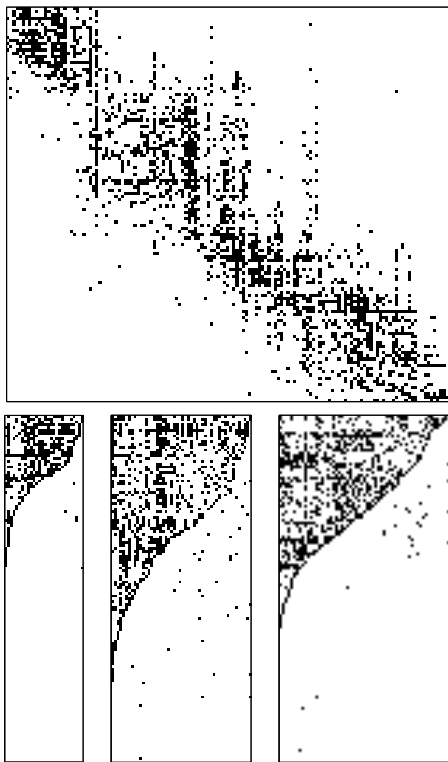


Figure 3: A paleontological dataset in its original form (top) and partitioned into three almost nested blocks that are individually sorted (bottom).

ber of novel and existing heuristics for finding almost  $k$ -nested structure on synthetic and real data. In particular, we introduce a heuristic relying on a combination of preprocessing with the singular value decomposition (SVD) followed by  $k$ -means++ clustering that appears to provide the best results in terms of structure discovery and robustness to noise.

Fourth, we study the model selection problem of choosing the number  $k$  of almost nested blocks in the presence of noise. We propose a novel solution based on the minimum description length (MDL) principle and investigate its performance on synthetic and real data.

Fifth, we test the algorithms on both synthetic and real-world data to show that  $k$ -nested patterns can be reliably recognized, and the patterns found in real-world data have meaningful interpretations that reflect the structure of the data.

**1.2 Background and related work.** The concept of nestedness has its roots in ecology [16, 18], but it was not defined formally until Mannila and Terzi [14] introduced the concept to the data mining community. They also presented another concept, segmented nestedness, which is the main focus in this paper.

The problem of finding  $k$  fully or almost nested submatrices is an instance of matrix reordering problems, for which many kind of patterns exist [12]. Examples of finding several reorderable patterns in one dataset include, for example, frequent itemsets [9] and maximally banded patterns [1], none of which requires a partitioning of columns. On the other hand, bucket orders [17] represent the data as ordered partitions. This is a dual problem of  $k$ -nestedness, which focuses on partitions of orders. In general the emphasis on matrix reordering has been on global patterns that fill the whole matrix, but the existence of several independent patterns has received less attention, perhaps partly due to the computational challenges involved in the task. Indeed, already in the case for a single pattern (that is,  $k = 1$ ), finding the minimum number of modifications to reach a target pattern can be computationally challenging.

For example, it is an NP-hard problem to find the minimum number of 0-to-1 flips needed to obtain a fully nested matrix from a given binary matrix [19, 14]; a recent study gives approximation algorithms for this problem [7]. It remains an open problem whether this task is NP-hard if both 0-to-1 and 1-to-0 flips are allowed; that is, if the task is to determine the minimum Hamming distance of a given input to a fully nested matrix. Our hardness proof for  $k > 1$  provides some evidence that this is the case, and is to our knowledge the first result that addresses Hamming distance in such a setting.

Another related problem (in the noise-free case) is the  $k$ -chain subgraph cover problem [20], in which the task is to determine if the edge set of a given bipartite graph is the union of the edges in  $k$  chain graphs. The difference to  $k$ -nestedness is that the partition is on edges, and several paths may be used to cover all 1s on a column.

**1.3 Organization.** The rest of this paper is organized as follows. Theoretical background for  $k$ -nestedness is built in Section 2. Heuristic algorithms for finding  $k$ -nested structure are introduced in Section 3 and an MDL-based approach for choosing a good  $k$  in Section 4. The performance of the algorithms is tested on synthetic data in Section 5 and on real-world data in Section 6.

## 2 Theoretical development

**2.1 Problem definitions.** A *binary matrix* is a matrix with entries in  $\{0, 1\}$ . Let  $A$  be an  $m \times n$  binary matrix with the entry at row  $i = 1, 2, \dots, m$  and column  $j = 1, 2, \dots, n$  denoted by  $a_{ij}$ . Associate with each column  $j$  of  $A$  the set  $C_j = \{i \in \{1, 2, \dots, m\} : a_{ij} = 1\}$ .

**DEFINITION 2.1.** *The matrix  $A$  is fully nested if there exists a permutation  $\pi : \{1, 2, \dots, n\} \rightarrow \{1, 2, \dots, n\}$  such that  $C_{\pi(1)} \supseteq C_{\pi(2)} \supseteq \dots \supseteq C_{\pi(n)}$ .*

We briefly recall the following facts about nestedness. First, a matrix  $A$  is fully nested if and only if its transpose  $A^T$  is fully nested. Second, a matrix  $A$  is fully nested if and only if the matrix obtained by exchanging the roles of the 0s and 1s in  $A$  is fully nested. Third, a matrix is fully nested if and only if it does not contain, up to permutation of rows and columns, a  $2 \times 2$  submatrix of the form

$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}.$$

We call such submatrices *switch boxes*.

**DEFINITION 2.2.** *A fully nested matrix is in normal form if there exist integers  $l_1 \geq l_2 \geq \dots \geq l_n$  such that  $C_j = \{1, 2, \dots, l_j\}$  for each  $j = 1, 2, \dots, n$ .*

Each fully nested matrix can be transformed into a unique fully nested matrix in normal form by permuting the rows and columns. Note that while the normal form is unique, the permutations that take the matrix into normal form need not be unique.

**DEFINITION 2.3.** *For  $k = 1, 2, \dots, n$ , the matrix  $A$  is fully  $k$ -nested if there exists a partition  $\{S_1, S_2, \dots, S_k\}$  of the columns of  $A$  into pairwise disjoint subsets such*

*that the submatrix  $A[S_j]$  obtained by restricting  $A$  to the columns  $S_j$  is fully nested for each  $j = 1, 2, \dots, k$ . We say that the partition  $\{S_1, S_2, \dots, S_k\}$  splits  $A$  into the fully nested submatrices  $A[S_j]$ .*

It follows that a matrix is fully nested if and only if it is fully  $k$ -nested with  $k = 1$ . Furthermore, if a matrix is fully  $k$ -nested, then it is  $k'$ -nested for each  $k' \geq k$ . Also, every  $m \times n$  matrix is fully  $k$ -nested with  $k = n$ , since a submatrix with only one column cannot contain a switch box.

**PROBLEM 2.1. ( $k$ -NESTEDNESS)** *Given a binary matrix  $A$  and a positive integer  $k$  as input, either (a) find a partition  $\{S_1, S_2, \dots, S_k\}$  that splits  $A$  into fully nested parts, or (b) assert that no such partition exists.*

The *Hamming distance*  $d(A, B)$  between two  $m \times n$  binary matrices,  $A$  and  $B$ , is the number of entries in which the matrices differ. Put otherwise,  $d(A, B) = |\{(i, j) : a_{ij} \neq b_{ij}\}|$ .

**PROBLEM 2.2. (ALMOST  $k$ -NESTEDNESS)** *Given a binary matrix  $A$  and a positive integer  $k$  as input, find a fully  $k$ -nested matrix  $B$  together with a partition  $\{S_1, S_2, \dots, S_k\}$  that splits  $B$  such that  $d(A, B)$  is minimized.*

Observe that such a fully  $k$ -nested matrix  $B$  need not be unique, and if  $A$  is fully  $k$ -nested, we have  $B = A$  and  $d(A, B) = 0$ .

**2.2 A polynomial-time algorithm for the  $k$ -Nestedness problem.** This section presents a reduction from  $k$ -NESTEDNESS to the MINIMUM PATH COVER problem in transitive directed acyclic graphs. For basic graph-theoretic terminology we refer to [4].

Let  $k = 1, 2, \dots, n$  and let  $A$  be an  $m \times n$  binary matrix given as input. For the remainder of this section we assume, without loss of generality, that  $A$  has distinct columns. Indeed, we can collapse each class of repeated columns into one representative column, solve for  $k$ -nestedness, and expand each representative column in the solution.

Let us construct a directed graph  $D$  with the columns of  $A$  as vertices, that is,  $V(D) = \{1, 2, \dots, n\}$ . Now, for each pair of distinct vertices  $s, t \in V(D)$  introduce a directed edge from  $s$  to  $t$  if and only if  $C_s \supseteq C_t$ . Put otherwise,  $E(D) = \{(s, t) : s \neq t \text{ and } C_s \supseteq C_t\}$ .

Because  $A$  has no repeated columns,  $D$  is acyclic. Furthermore, we observe that  $D$  is *transitive*, that is, for all  $a, b, c \in V(D)$  it holds that if  $(a, b) \in E(D)$  and  $(b, c) \in E(D)$ , then  $(a, c) \in E(D)$ . Figure 4 shows an example matrix and its directed graph.

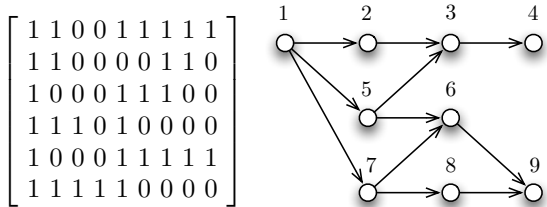


Figure 4: A fully 3-nested matrix with nine columns and its acyclic directed graph (transitive edges omitted). Three paths are needed to cover all the vertices.

LEMMA 2.1. *A directed path  $P$  in  $D$  defines a fully nested submatrix  $A[V(P)]$ . Conversely, if  $A[S]$  is a fully nested submatrix obtained by restricting  $A$  to columns  $S \subseteq \{1, 2, \dots, n\}$ , then there exists a directed path  $P$  in  $D$  with  $V(P) = S$ .*

*Proof.* Consider a directed path  $P$  in  $D$  with vertices  $V(P) = \{j_1, j_2, \dots, j_p\}$  and edges  $E(P) = \{(j_t, j_{t+1}) : t = 1, 2, \dots, p-1\}$ . We observe that  $C_{j_1} \supseteq C_{j_2} \supseteq \dots \supseteq C_{j_p}$  and  $A[V(P)]$  is fully nested. Conversely, let  $A[S]$  be a fully nested submatrix of  $A$  with  $|S| = p$ . Because  $A[S]$  is fully nested, there exists a bijection  $\pi : \{1, 2, \dots, p\} \rightarrow S$  such that  $C_{\pi(1)} \supseteq C_{\pi(2)} \supseteq \dots \supseteq C_{\pi(p)}$ . It follows that  $(\pi(t), \pi(t+1)) \in E(D)$  for each  $t = 1, 2, \dots, p-1$ . In particular, these  $p$  edges form the edge set of a directed path  $P$  with vertex set  $S$  in  $D$ .  $\square$

LEMMA 2.2. *Matrix  $A$  is fully  $k$ -nested if and only if there exist  $k$  directed paths  $P_1, P_2, \dots, P_k$  in  $D$  such that  $V(D) = \cup_{i=1}^k V(P_i)$ .*

*Proof.* It follows immediately from Definition 2.3 and the previous lemma that  $A$  is fully  $k$ -nested if and only if there exist  $k$  directed paths  $P_1, P_2, \dots, P_k$  in  $D$  such that (a)  $V(D) = \cup_{i=1}^k V(P_i)$  and (b) for all  $1 \leq i < j \leq k$  it holds that  $V(P_i) \cap V(P_j) = \emptyset$ . In particular, it suffices to show that the requirement (b) is redundant. Put otherwise, we must show that if there exists a collection of paths in  $D$  that meets (a), then there exists a collection of paths in  $D$  that meets both (a) and (b). It turns out that this follows from the transitivity of  $D$ .

Let  $P_1, P_2, \dots, P_k$  be a collection of paths that meets (a). For technical convenience, let us allow empty paths; that is,  $V(P_i) = \emptyset$  may hold. Now suppose that  $V(P_i) \cap V(P_j) \neq \emptyset$  holds for some  $1 \leq i \neq j \leq k$ . That is, there exists a  $v \in V(P_i) \cap V(P_j)$ . Let us delete  $v$  from  $P_i$  as follows. If  $v$  either starts or ends the path, we delete  $v$  and the only (if any) incident edge with  $v$  from  $P_i$ . Otherwise  $v$  is an internal vertex, with incident edges  $(u, v)$  and  $(v, w)$  in  $P_i$  with some  $u, w \in V(D)$ . By

transitivity of  $D$ , also edge  $(u, w)$  exists in  $D$ . We delete  $v$  and its edges from  $P_i$ , and add edge  $(u, w)$  to the path. After  $v$  has been removed,  $P_i$  is still a path, but it may be empty. By deleting vertices in this way from the paths, until no such deletions can be made, we obtain a collection of paths  $P_1, P_2, \dots, P_k$  (some of which may be empty) that meet both (a) and (b).

Suppose there are  $t$  empty paths. The  $(k-t)$  non-empty paths still form a path cover, which implies that  $A$  is  $(k-t)$ -nested, and also  $k$ -nested.  $\square$

PROBLEM 2.3. (MINIMUM PATH COVER) *Given a directed graph  $D$  as input, find directed paths  $P_1, P_2, \dots, P_k$  in  $D$  such that  $V(D) = \cup_{i=1}^k V(P_i)$  and  $k$  is minimized.*

MINIMUM PATH COVER in transitive acyclic directed graphs can be transformed [15] into a MINIMUM FLOW or a MAXIMUM MATCHING problem, for which efficient polynomial-time algorithms are known. The idea is to find the maximum independent set of vertices (that is, a maximum *antichain* in the set of columns partially ordered by the subset relation), which, by Dilworth's Theorem [5], has size equal to that of a minimum path cover. In the context of nestedness, this is a maximum set of columns such that every pair of columns from the set forms a switch box. Since constructing the directed graph  $D$  takes  $O(mn^2)$  time, which includes subset checks between all column-pairs, we have a polynomial-time algorithm for checking whether a binary matrix is fully  $k$ -nested.

**2.3 NP-hardness of the Almost  $k$ -Nestedness problem.** We will first develop a family of binary matrices where we have control over the distance to a fully nested matrix, after which NP-hardness of ALMOST  $k$ -NESTEDNESS will be established by reduction from a relaxation of the vertex cover problem.

Let us start by defining two families of undirected graphs, the *stars* and *daisies* as in Fig. 5. Throughout this section we assume that all graphs are undirected, loopless (that is, no edge joins a vertex to itself) and simple (that is, no two edges have the same endvertices).

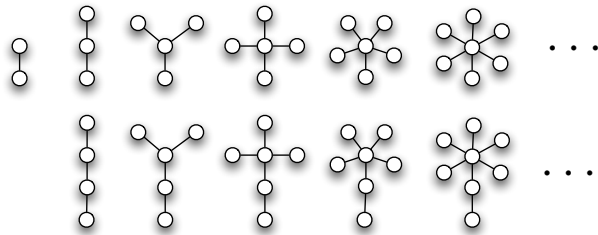


Figure 5: Stars (top) and daisies (bottom).

A *center* of a star or a daisy is a vertex of the maximum degree. Note that a center need not be unique if the number of edges is small (at most 1 for stars and at most 3 for daisies).

For an undirected graph  $G$  with  $n$  vertices and  $s$  edges, denote by  $N(G)$  the *incidence matrix* of  $G$ . That is,  $N(G)$  is the  $n \times s$  binary matrix with rows indexed by the vertices and columns indexed by the edges such that the entry at row  $i$ , column  $j$  is 1 if and only if the vertex  $i$  and edge  $j$  are incident in  $G$ . Because of our assumptions (all graphs are simple and loopless), every column of  $N(G)$  has exactly two 1s and there are no repeated columns.

Let us display the incidence matrix of the 3-edge star and the 4-edge daisy as an example:

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad \begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

The following lemmas analyze the (Hamming) distance of certain incidence matrices to a fully nested matrix. A *flip* refers to a change of one entry in a matrix (either 1-to-0 or 0-to-1); that is, the distance between two matrices is the minimum number of flips required to transform one matrix into the other.

LEMMA 2.3. *The incidence matrix of a  $s$ -edge star has distance  $s - 1$  to a fully nested matrix.*

*Proof.* (sketch) The idea is to flip  $s - 1$  entries (1s to 0s) in the identity part of the matrix. Details omitted.  $\square$

LEMMA 2.4. *The incidence matrix of a  $s$ -edge daisy has distance  $s - 1$  to a fully nested matrix.*

*Proof.* (sketch) The idea is to flip  $s - 2$  entries (1s) in the identity part of the matrix so that one specific 1 is left. Only one more flip is needed on the row that corresponds to the center vertex. Details omitted.  $\square$

LEMMA 2.5. *Let  $G$  be a triangle-free loopless simple graph with  $s \geq 1$  edges. Then, the incidence matrix of  $G$  has distance at least  $s - 1$  to a fully nested matrix, with equality if and only if  $G$  is a daisy or a star.*

*Proof.* (sketch) The idea is to use induction on  $s$  together with the observation that a triangle-free graph either is a star or has two disjoint edges. Exactly two flips are required to remove the switch boxes associated with the two disjoint edges, and a careful consideration of the remaining switch boxes establishes the claim. Details omitted.  $\square$

PROBLEM 2.4. (MINIMUM DAISY/STAR COVERING)

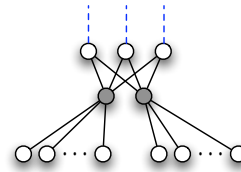
*Given an undirected graph  $G$  and a nonnegative integer  $k$  as input, decide whether the edges of  $G$  can be covered with at most  $k$  subgraphs of  $G$ , each of which is a daisy or a star.*

We observe that this problem is a relaxation of the classical minimum vertex cover problem, which insists on a covering with stars only.

THEOREM 2.1. MINIMUM DAISY/STAR COVERING is NP-complete when restricted to triangle-free graphs.

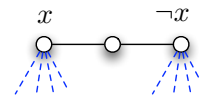
*Proof.* The problem is in NP because a given covering with daisies/stars can be checked in polynomial time. We reduce from 3SAT to establish completeness. Suppose we are given as input an instance of 3SAT with  $v$  variables and  $c$  clauses of length 3, such that each variable occurs in at least one clause. We construct a triangle-free graph with  $3v + c(5 + 2(2(v + 2c) + 1))$  vertices and  $2v + c(9 + 2(2(v + 2c) + 1))$  edges that has a covering with  $v + 2c$  daisies/stars if and only if the 3SAT instance is satisfiable.

For each clause in the 3SAT instance, introduce the following gadget that consists of  $5 + 2(2(v + 2c) + 1)$  vertices and  $9 + 2(2(v + 2c) + 1)$  edges; in the bottom row there are two groups of  $2(v + 2c) + 1$  vertices each.



Observe that all the black solid edges (and at most two of the dashed blue edges) in this gadget can be covered with two disjoint daisies/stars. In particular, those two must have their centers at the gray vertices. Indeed, if we do not use two daisies/stars centered at the gray vertices, we require at least  $v + 2c + 1$  daisies/stars to cover the black edges.

For each variable  $x$  in the 3SAT instance, introduce the following variable gadget that consists of 3 vertices and 2 edges (dashed blue edges not included).



Join each clause gadget via the dashed blue edges to the 3 vertices in the variable gadgets that correspond to the 3 literals in the clause.

Observe that if we use  $2c$  daisies/stars to cover the clause gadgets (which we must do if we want to use at

most  $v + 2c$  daisies/stars), then at least  $v$  daisies/stars are required to cover the variable gadgets, one for each gadget. Without loss of generality we may assume that such a daisy/star is a daisy and is centered at one of the literal vertices ( $x$  or  $\neg x$ ) and covers all the dashed blue edges incident to the vertex. Indeed, such a daisy covers at least the edges covered by any other daisy/star that covers both black edges in a variable gadget. Here we apply our assumption that every variable occurs in at least one clause; that is, each variable gadget is incident to at least one dashed blue edge.

We now claim that a daisy cover of size  $v + 2c$  exists if and only if the 3SAT instance is satisfiable. In the “if” direction, use a satisfying assignment to cover the variable gadgets with daisies so that each satisfied literal is the center of a daisy. Since each clause contains at least one satisfied literal, the corresponding dashed blue edge is covered by the daisy centered at the variable gadget, and the remaining at most two dashed blue edges can be covered with the daisies/stars in the clause gadget. Moreover, the daisies/stars in the cover can be taken to be pairwise disjoint. In the “only if” direction, use the centers of the daisies covering the variable gadgets to reconstruct a satisfying truth assignment.  $\square$

From the proof of the previous theorem we observe that the problem is NP-complete if instead of a cover we ask for a partition of the edges into daisies/stars.

**THEOREM 2.2.** *Let  $G$  be a simple loopless triangle-free graph with  $n$  vertices and  $s$  edges. Then,  $G$  has a partition into  $k$  daisies/stars if and only if  $N(G)$  has distance at most  $s - k$  to a fully  $k$ -nested binary matrix.*

*Proof.* Let us abbreviate  $N = N(G)$ . We start with the “only if” direction. Consider a daisy/star partition  $S_1, S_2, \dots, S_k$  of  $G$ . Denote by  $s_i$  the number of edges in  $S_i$ ,  $i = 1, 2, \dots, k$ . Consider the  $n \times s_i$  submatrix  $N[S_i]$  obtained by restricting  $N$  to columns  $S_i$ . From Lemma 2.5 we have that  $N[S_i]$  has distance exactly  $s_i - 1$  to a fully nested matrix. It follows that  $N$  has distance at most  $\sum_{i=1}^k (s_i - 1) = s - k$  to a fully  $k$ -nested matrix.

We continue with the “if” direction. Let  $B$  be a fully  $k$ -nested binary matrix at distance at most  $s - k$  from  $N$ , and let  $S_1, S_2, \dots, S_k$  be a partition of the columns that splits  $B$  into fully nested submatrices. Let  $s_i = |S_i|$  for  $i = 1, 2, \dots, k$  and observe that  $\sum_{i=1}^k s_i = s$ . Let  $d_i$  be the number of changes required to turn  $N[S_i]$  to  $B[S_i]$ . By assumption we have  $\sum_{i=1}^k d_i \leq s - k$ . For each  $i = 1, 2, \dots, k$ , the submatrix  $N[S_i]$  meets the conditions of Lemma 2.5, and hence  $d_i \geq s_i - 1$ , with equality if and only if  $S_i$  is either a daisy or a star. It follows that  $d_i = s_i - 1$  and that  $S_i$  is either a daisy or a star.  $\square$

It follows from Theorem 2.1 and Theorem 2.2 that, given positive integers  $k, d$  and a binary matrix  $A$  as input, it is an NP-complete problem to decide whether  $A$  is at distance at most  $d$  from a fully  $k$ -nested matrix. Consequently, ALMOST  $k$ -NESTEDNESS is NP-hard.

### 3 Algorithms

Given that ALMOST  $k$ -NESTEDNESS is NP-hard, it is unlikely that an exact polynomial-time algorithm exists for the problem. This section investigates heuristic algorithms that, given a binary matrix  $A$  and an integer  $k$  as input, find a fully  $k$ -nested matrix  $B$  such that the distance  $d(A, B)$  is as small as possible.

Our focus is on algorithms that attempt to find a good partition for the columns of  $A$  into  $k$  sets such that each induced submatrix  $A_1, A_2, \dots, A_k$  is close to a fully nested matrix. We compute a fully nested  $B_i$  for each  $A_i$  individually, and use  $\sum_{i=1}^k d(A_i, B_i)$  as an upper bound for the minimum distance of  $A$  to a fully  $k$ -nested matrix. Note that no polynomial-time algorithm is known for computing the minimum distance of a given  $A_i$  to a fully nested matrix. To approximate the distance, we use the method **Greedy** [14]. In its basic form, **Greedy** flips the entry that participates in the most switch boxes, and continues until no more switch boxes exist. Thus we arrive from  $A_i$  to  $B_i$ , and obtain an upper bound to  $d(A_i, B_i)$  by keeping track of the number of flips. We extend this method into **GreedyWeights** by associating positive weights with the entries of  $A_i$ . In the weighted setting, the distance between two matrices is the sum of the weights of the entries that differ, and the next entry to be flipped is an entry with the largest switch-boxes/cost efficiency.

We study one existing algorithm and two types of novel algorithms. First, the **Mannila-Terzi** algorithm [14] relies on hierarchical refinement of the partition of columns based on spectral ordering. Second, we introduce two novel techniques, **SVD-k-Baseline** and **SVD-k-Sim**, which are based on computing a singular value decomposition (SVD) of the input data. Third, we consider **k-Cut** and **AgglomerativeClustering** on the columns in the input, where pairs of columns are weighted according to the number of conflicts (switch boxes) that result if the two columns are placed in the same set of the partition.

**3.1 Mannila-Terzi.** The first algorithm for ALMOST  $k$ -NESTEDNESS was proposed by Mannila and Terzi [14], in which spectral ordering [13] is used on a graph that represents the similarities between columns. The method is hierarchical: in each step it evaluates each set of the current partition and then splits one set into two, and continues until the partition has  $k$  sets.

We use the variant that employs the inclusion similarity measure.

In its original form, the **Mannila-Terzi** algorithm invokes a greedy method **LocalMoves** afterwards to fine-tune the partition. We did not include this time-consuming method in the experiments, as it provides only minor improvements to the best algorithms.

**3.2 SVD- $k$ -Baseline.** We recall that the *singular value decomposition* (SVD) decomposes a real-valued matrix  $M$  into a product of three real matrices  $M = U\Sigma V^T$  where  $\Sigma$  is a diagonal matrix and  $U$  and  $V$  are unitary matrices; that is,  $UU^T$  and  $VV^T$  are identity matrices.

In the new **SVD- $k$ -Baseline** algorithm, given a binary matrix  $A$  and the number of nested blocks  $k$  as input, we run SVD on  $A$  and choose from  $V$  the first  $k$  columns that correspond with  $k$  largest singular values in  $\Sigma$ . These vectors form the basis of a  $k$ -dimensional Euclidean space. We project the columns of  $A$  into this space and run  **$k$ -means++** clustering algorithm [2], which gives us a partition of the columns.

Figure 6 displays the columns of a fully 3-nested matrix in the Euclidean space. We observe that the SVD apparently separates the nested blocks from each other, which enables one to find a good partition.

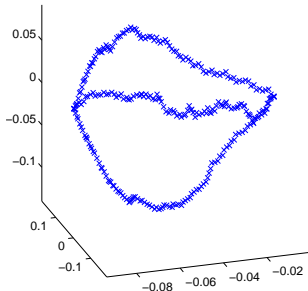


Figure 6: The columns of a fully 3-nested matrix  $A$  projected into a 3-dimensional Euclidean space. The basis vectors are the first three columns in the  $V$  matrix from the SVD of matrix  $A$ . We see three almost disjoint chains, each of which corresponds to a nested block in  $A$ . The chains intersect roughly at two points, which correspond to columns that are full of 1s and full of 0s.

**3.3 SVD- $k$ -Sim.** The  **$k$ -means++** clustering assumes that the data points approximately follow a mixture of Gaussian distributions, but the chains seen in Fig. 6 do not have this property. To improve on the results of **SVD- $k$ -Baseline**, we run SVD on a derived subset-similarity matrix instead of  $A$  directly, which appears to produce more Gaussian-like concentrations of points.

We construct an  $n \times n$  *subset-similarity matrix* as follows. Two columns  $c_1$  and  $c_2$ , and their corresponding sets,  $C_1$  and  $C_2$ , in an  $m \times n$  matrix are compared to see how likely they belong to same nested structure. Suppose the 1s in the columns were independent and uniformly distributed. Then the number of common 1s between  $c_1$  and  $c_2$ , denoted by  $X$ , would be distributed as  $X \sim \text{Hypergeom}(m, |C_1|, |C_2|)$ , for which expectation  $E(X)$  and variance  $\text{Var}(X)$  are known. Denote by  $|C_1 \cap C_2|$  the actual number of common 1s between  $c_1$  and  $c_2$ . The symmetric *subset-similarity* measure is defined as

$$\text{sim}(c_1, c_2) = \frac{1}{1 + \exp\left(-\frac{|C_1 \cap C_2| - E(X)}{\text{Var}(X)}\right)}.$$

Values in  $[0, 0.5)$  indicate that similarity is less than randomly expected, whereas values in  $(0.5, 1]$  indicate greater similarity. A logistic function was chosen as the basis of the measure because (a) the range of values is  $[0, 1]$ ; and (b) the curve is smooth.

**3.4  $k$ -Cut.** We recall that a  $k$ -cut in an undirected graph  $G$  is a set of edges whose removal partitions  $G$  into  $k$  connected components. In the **MAXIMUM  $k$ -CUT** problem, we are given (a) an undirected graph  $G$ , (b) a nonnegative integer weight for each edge of  $G$ , and (c) a nonnegative integer  $k$ ; the task is to find a  $k$ -cut with the maximum total weight.

We employ the following greedy algorithm for **MAXIMUM  $k$ -CUT**. Start with  $k$  empty bins, and insert the vertices of  $G$  one by one into the bins so that each vertex  $v$  is placed into the bin that has the smallest cost, where the cost is the sum of conflict-weights between  $v$  and the vertices already in the bin.

To obtain a heuristic solution to **ALMOST  $k$ -NESTEDNESS**, we view the columns of the input matrix  $A$  as the vertices of  $G$ , with the weights defined as follows. For two columns,  $c_1$  and  $c_2$ , denote by  $\text{conf}(c_1, c_2)$  the *conflict-weight* of the columns; that is, the number of switch boxes in the two columns. We obtain a partition of the columns of  $A$  into  $k$  sets from the connected components of the  $k$ -cut produced by the algorithm.

**3.5 Agglomerative clustering.** Agglomerative clustering [3] starts by viewing each column as its own cluster, and then proceeds to merge a pair of clusters with the smallest distance, continuing until  $k$  clusters with the smallest distance, the maximum, or the average of all pairwise conflict-weights between the columns of two clusters. We refer to these methods as **Agglo-Min**, **Agglo-Max**, and **Agglo-Ave**.

**3.6 Benchmark methods.** The following two benchmarks are used to assess the performance of the column-partitioning algorithms on synthetic data where an underlying ground truth is available.

Given the sizes of the underlying nested blocks in synthetic data, method **Random** returns a partition of the columns with these sizes, selected uniformly at random.

The generative process for the synthetic data includes a partition of the columns, which is used as an underlying ground truth before adding noise. Method **Original** returns this partition.

#### 4 Choosing $k$ with MDL

This section studies the problem of choosing the number  $k$  of almost nested blocks when the data is noisy and the correct number  $k$  is unknown. This problem is analogous to the problem of choosing the number  $k$  of clusters in the context of clustering, where techniques such as BIC, MML and MDL are employed to determine a good value of  $k$  [3, 10]. Here we develop an approach based on MDL for choosing a good number of nested blocks.

The *minimum description length* (MDL) principle states that the best model for the data uses as few bits as possible to represent the data. In other words, MDL chooses the model that best compresses the data by exploiting the regularities in the data.

We introduce a model and an associated encoding scheme for binary data that succinctly encodes a  $k$ -nested structure if such structure is present in the data. Put otherwise, our encoding scheme for almost  $k$ -nested matrices first computes a partition that identifies the almost nested blocks, then constructs fully nested matrices that are close to these blocks, and finally encodes the fully  $k$ -nested structure together with the differences between the fully nested matrices and the data. The best value of  $k$  is the one that gives the minimum encoded length for the data.

Next we describe the *nestedness scheme* in more detail. To encode an  $m \times n$  binary data matrix  $A$  using  $k = 1, 2, \dots, n$  nested blocks, we proceed as follows. First, we encode the dimensions  $m$  and  $n$  of the matrix with Elias  $\delta$ -coding [6], which is a universal integer coding scheme. Then, we encode the number of nested blocks  $k$  using  $\log_2 n$  bits. (We adopt the convention of not rounding up the codelengths to integers.) Next we use **SVD- $k$ -Sim** to find a partition of the columns of  $A$  into  $k$  almost nested blocks, and encode the partition with  $n \log_2 k$  bits.

We then repeat the following operations for each almost nested block  $A_i$  of  $A$  with  $i = 1, 2, \dots, k$ . Let  $n_i$  be the number of columns in  $A_i$ ; the value  $n_i$  can be recovered from the encoded partition. We run

**GreedyWeights** on  $A_i$  and obtain (a) a matrix  $Q_i$  that is normal form; and (b) row and column permutations for  $Q_i$  such that the permuted matrix is close to  $A_i$ . We encode the permutations using  $\log_2(m!) + \log_2(n_i!)$  bits.

We observe that because  $Q_i$  is in normal form, its structure can be described compactly by using a staircase path that identifies the boundary between the 1s and 0s (see Fig. 1). Put otherwise, we start at the bottom-left corner of  $Q_i$ , and move in between the entries of  $Q_i$  either right or up until we arrive at the top-right corner. In particular, such a staircase path consists of  $m + n_i$  moves, exactly  $m$  of which are up-moves. It thus takes  $\log_2 \binom{m+n_i}{m}$  bits to encode  $Q_i$ .

To recover  $A_i$  from the encoded  $Q_i$  and the row and column permutations, we still must encode the differences between  $A_i$  and the permuted  $Q_i$ . Observe that we can recover the number of 1s and 0s in  $Q_i$ , denoted by  $p_1$  and  $p_0$ , respectively, from the existing encoding. We encode the number of differences, that is, the number of 1s in  $A_i$  where the permuted  $Q_i$  has 0s, denoted by  $e_1$ , as well as the number of 0s that occur in  $A_i$  but not in the permuted  $Q_i$ , denoted by  $e_0$ . Finally, the positions of all the differences are encoded using  $\log_2 \binom{p_0}{e_1} + \log_2 \binom{p_1}{e_0}$  bits.

The total codelength is thus

$$L_{\text{nest}} = \delta(m) + \delta(n) + \log_2 n + n \log_2 k + \sum_{i=1}^k \left( \log_2(m!) + \log_2(n_i!) + \log_2 \binom{m+n_i}{m} + \log_2(p_0+1) + \log_2(p_1+1) + \log_2 \binom{p_0}{e_1} + \log_2 \binom{p_1}{e_0} \right).$$

As a baseline encoding scheme—that is, without assuming a nested structure—we employ the following *uniform scheme*, which encodes the binary matrix as is. First, encode the dimensions  $m$  and  $n$ , again with Elias  $\delta$ -coding. Then, encode the number of 1s in the data,  $p = 0, 1, \dots, mn$ , together with the  $p$  entries in the matrix out of all  $\binom{mn}{p}$  possibilities. The total codelength is

$$L_{\text{unif}} = \delta(m) + \delta(n) + \log_2(mn+1) + \log_2 \binom{nm}{p}.$$

#### 5 Experiments on synthetic data

In the following sections we show how to generate almost  $k$ -nested synthetic binary data, and how the generative process gives a ground truth, against which ALMOST  $k$ -NESTEDNESS algorithms can be compared. We then proceed to three tests, in which the algorithms need to recognize existing  $k$ -nested structure, reconstruct

an underlying  $k$ -nested structure, and find reliably the correct number of nested blocks  $k$  for a data matrix.

**5.1 Data generation.** We generate almost  $k$ -nested binary data  $A$  synthetically by first generating a fully  $k$ -nested matrix and then adding errors to it.

The parameters include the number of rows  $m$ , columns  $n$ , nested blocks  $k$ , and the size for each block  $(n_1, n_2, \dots, n_k)$ . Each block  $i = 1, 2, \dots, k$  represents an  $m \times n_i$  matrix  $A_i$ , where row  $r$  has its first  $\ell_{i,r}$  entries as 1s and  $\ell_{i,r}$  is a random number from  $\{0, 1, \dots, n_i\}$ .

We then construct a fully  $k$ -nested  $m \times n$  matrix  $B$  by merging all the blocks together.

Real-world datasets often have two types of errors: missing data and misclassifications. In the former some true 1s appear in data as 0s, and vice versa for the latter. We simulate these errors in synthetic data by noise probability parameters  $p(1\text{-to-}0)$  and  $p(0\text{-to-}1)$ . Dataset  $A$  is generated by independently flipping the entries in  $B$  according to these probabilities.

It is common that ecological presence/absence datasets have lots of missing data, yet misclassifications are rare. To simulate this, we use *asymmetric* noise, in which  $p(0\text{-to-}1)$  and  $p(1\text{-to-}0)$  are unequal. An alternative is to use *symmetric* noise, in which the noise parameters are equal, that is  $p(0\text{-to-}1) = p(1\text{-to-}0)$ .

When  $k$ -nested data has high levels of noise, the data may deviate from the underlying structure. Furthermore, the partition under which data is generated does not minimize Hamming distance anymore, and other algorithms can do better than **Original**.

**5.2 Distance test.** In this test we measure the Hamming-distances the algorithms produce in ALMOST  $k$ -NESTEDNESS problem, when the correct  $k$  is given but the number of columns in each block is unknown. A low distance indicates that the underlying  $k$ -nested structure is recognized.

For each level of symmetric and asymmetric noise, we generate 30 samples of  $150 \times 150$  matrices that are 3-nested with block sizes 25, 50, and 75. We use each algorithm to produce a partition for each dataset, and we approximate the Hamming-distance to the closest fully 3-nested matrix by using **Greedy** algorithm on each submatrix induced by the partition.

The results for asymmetric noise are shown in Fig. 7, where each data point is an average of 30 samples. We see that **SVD- $k$ -Sim** has the lowest distances. Furthermore, the distances are practically the same that the benchmark method **Original** produces, which suggests **SVD- $k$ -Sim** is near-optimal at least with low noise levels. At higher noise levels the partition from **SVD- $k$ -Sim** describes the data even better than

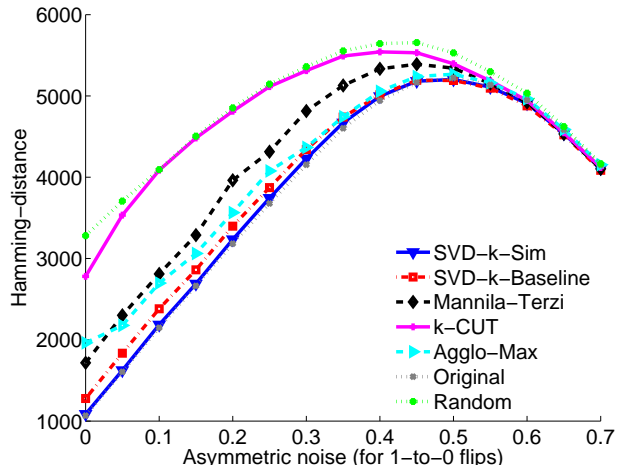


Figure 7: Results of the distance test with asymmetric noise; the effect of missing data on the distance. On the  $x$ -axis the noise levels are shown for asymmetric missing data  $p(1\text{-to-}0)$ . The misclassification parameter is fixed to  $p(0\text{-to-}1) = 0.10$ . On the  $y$ -axis are the approximated Hamming distances to closest fully 3-nested matrices; the distances are averages over 30 samples.

**Original.** Of the other methods, **SVD- $k$ -Baseline** is close to **SVD- $k$ -Sim**, and **Agglo-Max** is a bit behind. These three algorithms perform better than **Mannila-Terzi**, in which a bad early choice of splitting sets in partition might lead to poor results. The **k-Cut** algorithm has the worst performance, which is roughly the same as what a random partition with correct block sizes produces. Of the three **Agglo** algorithms presented, we only show the results for **Agglo-Max**, which is the best of the three. The results for symmetric noise (not shown) are analogous.

From now on, we include only **SVD- $k$ -Sim** in further experiments, as it produces the best distances and is reasonably fast in practice.

**5.3 Classification test.** In this test we evaluate how well the partition produced by **SVD- $k$ -Sim** matches with the underlying  $k$ -nested structure. To be more precise, we measure classification *accuracy*, which is the proportion of columns that a certain partition assigns to same sets as **Original**. We assume the correct  $k$  is given, but the number of columns in each block is unknown.

For each noise level, we generate 50 samples of  $r \times 150$  matrices, where the number of rows is  $r = 6, 25, 100, 400, 1600$ , and matrices are 3-nested with block sizes 25, 50, and 75. For each dataset we run the **SVD- $k$ -Sim** algorithm and compute accuracy for

its partition. To compare the sets in two partitions correctly, we try all bijections between sets and choose the one that results in maximum accuracy.

The results for symmetric noise are displayed in Fig. 8. The more rows the matrix has, the better accuracy `SVD- $k$ -Sim` achieves. The columns that are full of either 1s or 0s can belong to any nested block. Despite of this, `SVD- $k$ -Sim` has high accuracy, and when the number of rows is low, we see satisfactory results. At high noise levels the underlying fully  $k$ -nested structure has vanished, and `SVD- $k$ -Sim` produces partitions that fit better with the noisy data. The results for asymmetric noise (not shown) are analogous.

**5.4 MDL test.** In this test we assess the reliability of MDL model selection, that is, test whether MDL is able to retrieve the correct  $k$  from synthetic data when we use `SVD- $k$ -Sim` to find a partition and `GreedyWeights` to find fully nested blocks.

We generate 30 samples of  $150 \times 150$  matrices that are  $k$ -nested. We repeat the generation process for each true value  $k = 0, 1, 2, \dots, 10$ , and the sizes of  $k$  nested blocks are (almost) equal. With parameter  $k = 0$ , the data has random uniform 50% fill of 1s. Symmetric noise is added to each dataset.

We consider both nestedness scheme and uniform scheme, and use `GreedyWeights` with equal weights

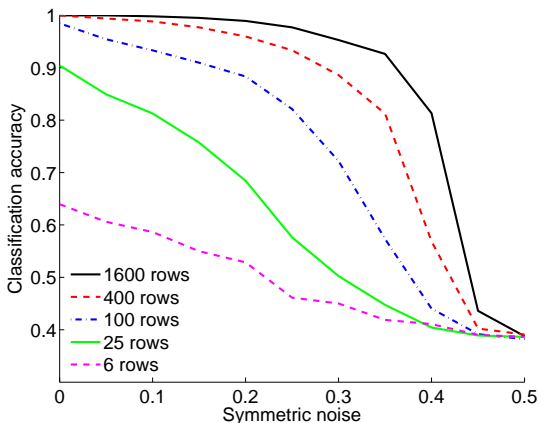


Figure 8: Results of the classification accuracy test with symmetric noise for `SVD- $k$ -Sim`. On the  $x$ -axis are the symmetric noise levels  $p(1\text{-to-}0) = p(0\text{-to-}1)$ . On the  $y$ -axis classification accuracies are shown. Each line corresponds to a number of rows  $r$ , and the average accuracy from 50 samples of  $r \times 150$  data matrices are shown for that line. The more rows in the data, the better accuracy. The expected accuracy of a uniform random partition with correct block sizes is 0.39.

for 0-to-1 and 1-to-0 flips. From numbers of blocks  $1, \dots, 20$ , we choose the one that produces the shortest codelength in nestedness scheme, or zero in case the uniform scheme has the shortest codelength.

The results of MDL model selection with symmetric noise are shown in Fig. 9. The general observation is that the method reliably finds the true  $k$ . If each nested block consists of only a few columns, or if noise levels are high, finding the correct  $k$  is hard. In these situations the MDL method usually favors uniform scheme  $k = 0$  instead of choosing an incorrect positive value for  $k$ .

## 6 Experiments on real-world data

We next describe two real-world datasets and show that they have a  $k$ -nested structure, as found by the `SVD- $k$ -Sim` method and MDL.

**6.1 Mammals data.** The Mammals dataset<sup>1</sup> is a  $124 \times 2179$  binary matrix, in which rows represent mammal species and columns are  $40 \times 40$  km<sup>2</sup> locations in Europe. Value 1 indicates that a mammal occurs in a location.

We use `SVD- $k$ -Sim` and MDL to analyze whether the

<sup>1</sup>The dataset is available by request from the Societas Europaea Mammalogica. <http://www.european-mammals.org/>

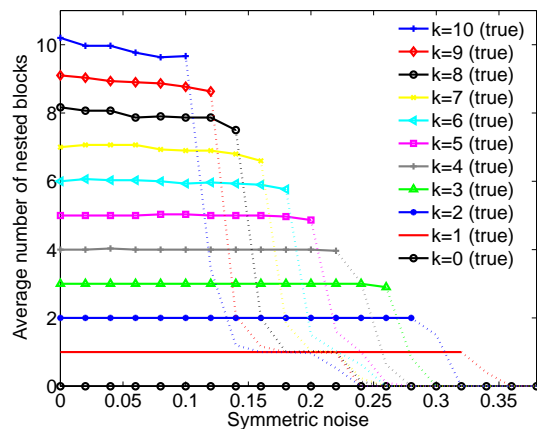


Figure 9: Results of the MDL test with symmetric noise. Displayed on the  $x$ -axis are the levels of symmetric noise; on the  $y$ -axis are the estimated numbers of nested blocks. The legend gives the correct value of  $k$  for each line, and plotted values are estimates by MDL. Each value is an average from 30 samples. If the uniform encoding scheme is preferred, the number of blocks is zero. Each line is divided into two parts: the solid part is for noise levels where correct  $k$  is reliably recovered, and the dotted part displays the divergent part.

locations have a  $k$ -nested structure. If the uniform random noise model produces the shortest codelength, the data is considered non-nested with  $k = 0$ . We selected equal weights for flipping 0s and 1s in `GreedyWeights`.

Figure 10 shows a sample partition of locations when MDL selects  $k = 16$ . Geographically coherent areas are revealed without using spatial information, which asks for detailed ecological investigation. For example, Fig. 11 shows the submatrices that correspond to nested blocks in Western France, Bulgaria and Northern Finland. To our knowledge, the observed nested areas agree with common knowledge about the distribution of mammals, for example with cluster analysis [11].

Because of randomized nature of  $k$ -means++ algorithm, the value of  $k$  selected by MDL is not always the same. From the 50 samples of running `SVD-k-Sim` and MDL on the locations, we obtain median 16 and standard deviation 4.7 for the number of blocks. The differences in MDL codelength are typically small with real-world data, and this is the case with values around 16. We conclude that the locations in Mammals data are indeed  $k$ -nested, but  $k$  is not a single number but a range of numbers around 16.

On the other hand, the same methods suggest that the species in Mammals data are 1-nested. The data is displayed in Fig. 12, with rows and columns permuted to reveal the nested structure.

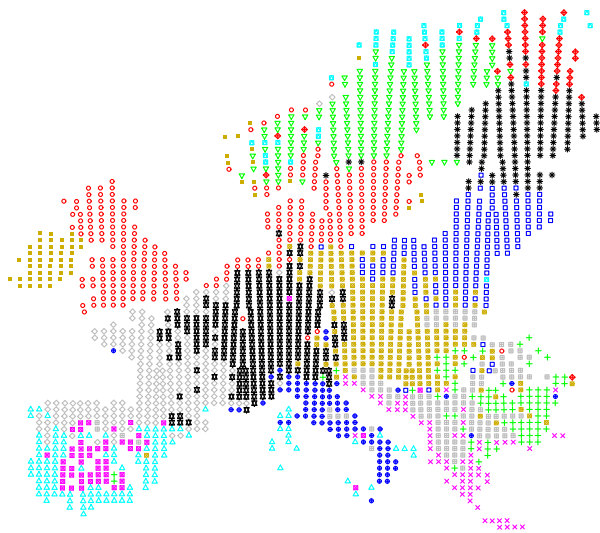


Figure 10: Analyzing nestedness in the Mammals data. We display European locations in their 16-nested form, as selected by MDL. All points that share a color and symbol belong to the same nested block. We observe geographically coherent blocks, without any use of spatial information in the algorithms.

**6.2 Paleontological data.** The Paleontological dataset [8] has binary information on fossils genera in Europe. There are 124 sites (rows) and 139 genera (columns), and value 1 indicates that a fossil of specific genus has been found on a site, whereas 0 means that it has not been found.

We assume that the data has lots of missing 1s, as is typical for this kind of data, and assign a weight parameter  $w = 4$ , meaning that the cost for a 1-to-0 flip is four times that of a 0-to-1 flip. These weights are used in `GreedyWeights` to find fully nested matrices.

We use `SVD-k-Sim` and MDL to evaluate a good choice of  $k$  for genera, and to determine whether an assumption of  $k$ -nestedness should be discarded (uniform scheme,  $k = 0$ ). The result is that MDL always selects  $k = 3$ ; the dataset and its partition into three nested blocks are shown in Fig. 3. This corroborates an earlier claim [14] that genera are 3-nested.

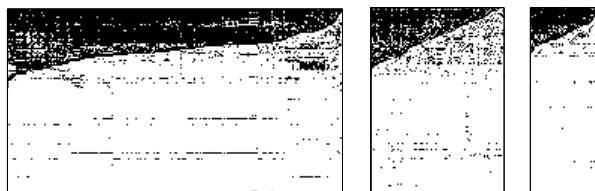


Figure 11: Examples of almost nested blocks in Fig. 10, from left to right: Western France (225 locations), Bulgaria (89), and Northern Finland (47). The rows are mammals and the columns are locations.

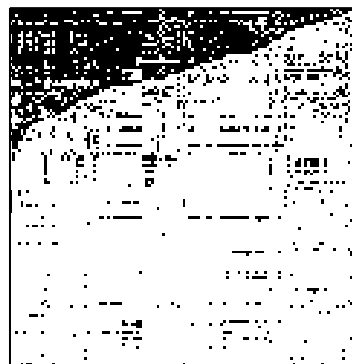


Figure 12: Mammals data with its rows and columns permuted to show its nested structure. Rows are mammal species, columns are European locations, and black dots indicate presence. Only every 18th column is shown.

## 7 Conclusions and further research

While we have shown that it is an NP-complete problem to decide whether a binary matrix  $A$  is  $k$ -nested after at most  $d$  flips for given  $A$ ,  $k$ , and  $d$ , our experiments suggest that  $k$ -nestedness can be studied on real-world data with heuristic tools, such as the SVD-based algorithms. In this connection we would like to highlight two directions for further research.

First, the possibility of rigorous approximation algorithms for  $k$ -nestedness should be investigated, together with the computational complexity of ALMOST  $k$ -NESTEDNESS in the restricted case  $k = 1$ .

Second, we believe that the SVD-based algorithms would benefit from a more detailed analysis. In particular, we would like to highlight Fig. 6, where the chains are easy to distinguish visually—why do such chains emerge, and what tools could be used to automatically detect such chain-like patterns in the presence of noise?

We have proposed an MDL-based model selection approach to discover  $k$ -nestedness in practice. In addition to sound performance on synthetic data, the tests on real-world data show that MDL is able to discover interesting structures, such as geographically coherent European areas based on mammal occurrences, and a 3-nested structure in paleontological data. In this connection we observe that MDL discovers a 1-nested structure for the mammal species and a 16-nested structure for the locations in the Mammals data. While a further analysis of the results from an ecological perspective is clearly warranted, it is also of theoretical interest to understand how the (almost) nestedness properties of a matrix and its transpose may interact.

A conceptual extension of nestedness that remains to be studied is *laminarity*: a collection of sets is a *laminar family* if any two sets in the collection are either disjoint or one set is a subset of the other.

## References

- [1] F. Alqadah, R. Bhatnagar R, and A.G. Jegga, *Mining maximally banded matrices in binary data*, SIAM International Conference on Data Mining (2010), pp. 942–953
- [2] D. Arthur and S. Vassilvitskii, *k-means++: the advantages of careful seeding*, ACM-SIAM symposium on Discrete algorithms (2007), pp. 1027–1035.
- [3] P. Berkhin, *Survey of clustering data mining techniques*, Technical report (2002).
- [4] R. Diestel, *Graph Theory*, Fourth Edition (2010), Springer-Verlag, Heidelberg, Graduate Texts in Mathematics, Vol. 173.
- [5] R. P. Dilworth, *A Decomposition Theorem for Partially Ordered Sets*, The Annals of Mathematics, Second Series, 51 (1950), pp. 161–166.
- [6] P. Elias, *Universal codeword sets and representations of the integers*, IEEE Transactions on Information Theory, Vol. IT-21 (1975), pp. 194–203.
- [7] T. Feder, H. Mannila, and E. Terzi, *Approximating the Minimum Chain Completion problem*, Information Processing Letters, 109 (2009), pp. 980–985.
- [8] M. Fortelius, *Neogene of the old world database of fossil mammals (NOW)*, (2008) <http://www.helsinki.fi/science/now>
- [9] B. Goethals, *Survey on frequent pattern mining*, Technical report, Helsinki Institute for Information Technology HIIT (2003).
- [10] M. H. Hansen and B. Yu, *Model selection and the principle of minimum description length*, Journal of the American Statistical Association, 96 (2001), pp. 746–774.
- [11] H. Heikinheimo, M. Fortelius, J. Eronen, and H. Mannila, *Biogeography of European land mammals shows environmentally distinct and spatially coherent clusters*, Journal of Biogeography, 34 (2007), pp. 1053–1064.
- [12] I. Liiv, *Seriation and matrix reordering methods: An historical overview*, Statistical Analysis and Data Mining, 3 (2010), pp. 70–91.
- [13] U. von Luxburg, *A tutorial on spectral clustering*, Statistics and Computing, 17 (2007), pp. 395–416.
- [14] H. Mannila and E. Terzi, *Nestedness and Segmented Nestedness*, ACM conference on Knowledge Discovery and Data Mining (2007), pp. 480–489.
- [15] S. C. Ntafos and S. L. Hakimi, *On Path Cover Problems in Digraphs and Applications to Program Testing*, IEEE Transactions on Software Engineering, 5 (1979), pp. 520–529 .
- [16] B. Patterson and W. Atmar, *Nested subsets and the structure of insular mammalian faunas and archipelagos*, Biological Journal of the Linnean Society, 28 (1986), pp. 65–82.
- [17] A. Ukkonen, K. Puolamäki, A. Gionis, and H. Mannila, *A randomized approximation algorithm for computing bucket orders*, Information Processing Letters, 109 (2009), pp. 356–359.
- [18] W. Ulrich, M. Almeida-Neto, and N. J. Gotelli, *Consumer’s guide to nestedness analysis*, Oikos, 118 (2009), pp. 3–17.
- [19] M. Yannakakis, *Computing the minimum fill-in is NP-complete*, SIAM Journal on Algebraic and Discrete Methods, 2 (1981), pp. 77–79.
- [20] C.-W. Yu, G.-H. Chen, and T.-H. Ma, *On the complexity of the k-chain subgraph cover problem*, Theoretical Computer Science, 205 (1998), pp. 85–98.