Journal of Machine Learning Research 22 (2021) 1-41 Submitted 10/20; Revised 8/21; Published 9/21

Learning partial correlation graphs and graphical models by covariance queries

Gábor Lugosi GABOR.LUGOSI@UPF.EDU
ICREA, Pg. Lluís Companys 23,
08010 Barcelona, Spain;
Department of Economics and Business
Pompeu Fabra University; and
Barcelona School of Economics

Jakub Truszkowski JAKUB.TRUSZKOWSKI@BIOENV.GU.SE
LIRMM, CNRS
Université de Montpellier
Montpellier, France, and
Department of Biological and Environmental Sciences and
Gothenburg Global Biodiversity Centre
University of Gothenburg
Gothenburg, Sweden

Vasiliki Velona VASILIKI.VELONA@UPF.EDU
Department of Economics and Business
Pompeu Fabra University; and
Einstein Institute of Mathematics
Hebrew University of Jerusalem

Piotr Zwiernik PIOTR.ZWIERNIK@UPF.EDU
Department of Economics and Business
Pompeu Fabra University; and
Barcelona School of Economics
Barcelona, Spain

Editor: David Sontag

Abstract

We study the problem of recovering the structure underlying large Gaussian graphical models or, more generally, partial correlation graphs. In high-dimensional problems it is often too costly to store the entire sample covariance matrix. We propose a new input model in which one can query single entries of the covariance matrix. We prove that it is possible to recover the support of the inverse covariance matrix with low query and computational complexity. Our algorithms work in a regime when this support is represented by tree-like graphs and, more generally, for graphs of small treewidth. Our results demonstrate that for large classes of graphs, the structure of the corresponding partial correlation graphs can be determined much faster than even computing the empirical covariance matrix.

Keywords: Gaussian graphical models, partial correlation graphs, structure learning, high-dimensional structures

1. Introduction

Learning the graph structure underlying probabilistic graphical models is a problem with a long history; see Drton and Maathuis (2017) for a recent exposition. In the classical setting, when the number n of variables is reasonably small, this can be done by using stepwise selection procedures based on information criteria like BIC, AIC, or using the likelihood function; see (Højsgaard et al., 2012, Section 4.4) for a discussion.

In high-dimensional scenarios the methods proposed for Gaussian graphical models have become particularly successful. Here the graphs are encoded by zeros in the inverse covariance matrix (or *precision matrix*) K. Specifically, an edge is present in the graph if and only if the corresponding element of K is not zero and so LASSO-type learning procedures can be applied Banerjee et al. (2008); Yuan and Lin (2007). The link between the entries of K and coefficients obtained by linearly regressing one variable against the rest gave rise to the so-called neighbor selection methods, see Meinshausen and Bühlmann (2010). In all these theoretical developments, the sample complexity required for learning the underlying graph is well understood. On the other hand, in these studies either computational issues played a secondary role or the computational budget was relatively large as all known methods require computing the sample covariance; see, for example, Cai et al. (2016); Dasarathy et al. (2016). Hsieh et al. (2013) perform a careful analysis of the optimization objective used in earlier methods, which leads to a divide-and-conquer algorithm that can be applied to large data sets. More recently, Zhang et al. (2018) devised another scalable procedure based on thresholding the sample covariance matrix followed by a novel optimization step. However, the computational complexity of these approaches is still of at least quadratic order as a function of the number of variables.

In a growing number of applications, the number of variables n is so large that a computational cost of order n^2 becomes prohibitive. This means that even writing down or storing the covariance matrix (or an estimate of it) is not practical, rendering all aforementioned approaches unfeasible. Such examples occur in some applications in biology, such as the problem of reconstructing gene regulatory networks from large scale gene expression data. Hwang et al. (2018) give an extensive discussion of computational challenges of massive amounts of gene expression data and note that issues of computational complexity made researchers rely on pairwise notions of dependence; see, for example, Chan et al. (2016); Zhang et al. (2011). Scalable algorithms are also of interest in phylogenetics, where the problem is to reconstruct the evolutionary relationships between tens to hundreds of thousands of DNA sequences (Price et al. (2010); Brown and Truszkowski (2012, 2011)). Another example leading to large networks is building human brain functional connectivity networks using functional MRI data. In this setting, the data are usually aggregated to obtain a data set with a moderate number of variables that can be processed with current algorithms Huang et al. (2010). More efficient methods to build large networks will allow researchers to study functional MRI data at a much higher resolution.

In this paper we address the problem that quadratic complexity becomes prohibitive in modern large-scale applications. This requires a different approach to structure recovery, which addresses the computational issues much more carefully. This computationally conscious approach has become more popular in the recent years where, in selected scenarios, it was possible to study the trade-off between statistical accuracy and computational

complexity, see Chandrasekaran and Jordan (2013); Rudi et al. (2015). In the main part of the paper we abstract away from statistical considerations and formulate the following mathematical problem. Given a symmetric positive definite matrix Σ the goal is to:

> Learn the support of $K = \Sigma^{-1}$ observing only a small fraction of Σ.

More precisely, we are interested in learning K based on a small number of adaptively selected entries of Σ. As a main motivating example, we may think about Σ as a covariance matrix of some underlying random vector $X = (X_1, \ldots, X_n)$, that is, the entries of Σ are the covariances $\sigma_{ij} = \mathbb{E}[(X_i - \mathbb{E}X_i)(X_j - \mathbb{E}X_j)]$. Although formulated on the population level, our result is a prerequisite for consistently learning the graph from data based on $o(n^2)$ entries of the sample covariance matrix. Our results show that, for data sets where covariances can be estimated with sufficient accuracy, often it is possible to recover the underlying graphical model in much less time than what is required by current approaches.

In order to formalize this approach, we propose the following input model for our analysis. The data can be accessed through queries to a *covariance oracle*. The covariance oracle takes a pair of indices $i, j \in [n]$ as an input and outputs the corresponding entry σ_{ij} of the matrix Σ. This is an idealized scenario that makes the main ideas of this paper more transparent. In practice, of course, these covariances are not exactly available as they are often estimated from data. This setup is meaningful in applications in which one may estimate, relatively easily and accurately, the covariance between any given pair of variables. Importantly, one does not need to estimate the entire covariance matrix. In Section 6 we discuss conditions under which the idealized covariance oracle may be replaced by a noisy version.

The *query complexity* of an algorithm is the number of entries of the covariance matrix Σ queried during the execution of the algorithm. The main findings of the paper show that, in many nontrivial cases, the graph underlying the graphical model of X can be recovered with only $\mathcal{O}(n\,polylog(n))$ queries using randomized algorithms. The computational complexity of the proposed algorithms is also quasi-linear. This is a significant decrease in complexity compared to the quadratic complexity of any recovery algorithm that uses the entire (estimated) covariance matrix as a starting point.

Of course a so stated problem cannot be solved in full generality and the algorithms need to rely on the sparsity of K induced by bounds on related parameters of the underlying graph such as maximum degree and treewidth (to be defined below). We propose randomized procedures that recover the correct graph and have low query and computational complexity with high probability. Our main results are briefly presented in Section 2, after introducing some necessary definitions. The rest of the paper is devoted to a careful analysis of three main cases: trees, tree-like graphs, and graphs with small treewidth. Our main result is an algorithm for each of the three cases, which recovers the correct graph with query and computational complexity $\mathcal{O}(n\,polylog(n))$.

In our analysis we first assume that the true underlying graph is a tree. In this case the Chow-Liu algorithm (Chow and Liu (1968)) is a widely used computationally efficient algorithm to search for the tree that maximizes the likelihood function. The method was originally proposed for categorical variables but it works in a much more general context with the Gaussian likelihood or any other modular criterion such as BIC, AIC as discussed

by Edwards et al. (2010). In our setting the Chow-Liu algorithm is equivalent to computing a maximum-weight spanning tree in the complete graph with edge weights given by the absolute values of the correlations between any two variables. Although the Chow-Liu algorithm is relatively efficient and it has good statistical properties, the computational cost is of order $\Omega(n^2)$, which may be prohibitive in large-scale applications. We introduce a simple randomized algorithm that recovers the tree with computational complexity $\mathcal{O}(dn\log(n))$, where d is the maximum degree of the graph. For a large class of trees, when the maximum degree is small, the algorithm introduced here significantly outperforms the Chow-Liu algorithm. At the same time, the maximum degree of a tree can be as large as $n-1$ (for the so-called star graph). For trees with linear maximum degree, our algorithm does not improve on the Chow-Liu algorithm. However, in view of the lower bound shown in Section 3.4, no algorithm can recover trees of maximum degree d with less than $\Omega(dn)$ covariance queries. In this sense, our algorithm is optimal up to logarithmic factors.

More generally, in Section 3 we study the problem of learning tree-like structures, that is, graphs whose 2-connected components have at most logarithmic size. Such graphs arise in a variety of settings; in particular, several well-studied random graph models often give rise to graphs whose 2-connected components are of logarithmic size, see Panagiotou and Steger (2010). In Sections 4 and 5 we study the much more general family of graphs with bounded treewidth. Bounded treewidth graphs have long been of interest in machine learning due to the low computational cost of inference in such models Chandrasekaran et al. (2012); Karger and Srebro (2001); Kwisthout et al. (2010). Moreover, current heuristics of treewidth estimation in real-world data have indicated small treewidth in various cases of interest Abu-Ata and Dragan (2016); Adcock et al. (2013); Maniu et al. (2019).

The main motivation of this work was learning Gaussian graphical models but our results hold in a much broader context. Our goal is to learn zeros in the inverse covariance matrix when covariances may be queried. Hence, this work may be phrased as a contribution to numerical linear algebra. In the statistical setting, learning zeros in K corresponds to learning the partial correlation graph. Vanishing partial correlations correspond to conditional independence (and so graphical models) in the Gaussian case but also in the non-paranormal case of Liu et al. (2009, 2012). We provide more details in Section 2.2. In general, partial correlation graphs inform only about linear dependences but there are still interesting situations when much more is implied by vanishing partial correlations Rossell and Zwiernik (2020).

Our work also provides a new way of performing constraint-based inference in Gaussian graphical models and partial correlation graphs. This approach was pioneered by the TETRAD program Scheines et al. (1998) where vanishing tetrad constraints are used to infer the structure of hidden variable graphical models; see also Spirtes et al. (2000) and Pearl (2009). For Gaussian graphical models, the most complete results listing the underlying model constraints were provided by Sullivant et al. (2010) and their results are used extensively in this article. Our new paradigm of adaptively searching over the set of potential constraints may also be useful for discrete graphical models, where the underlying graphical model can be read off the partial correlation graph of appropriately extended random vector as explained by Loh and Wainwright (2013).

The case when Σ is observed with error leads to additional complications. Solving this problem in full generality is beyond the scope of this paper. In order to present the main

ideas and some bottlenecks, in Section 6 we study the problem of recovering tree models when only a noisy covariance oracle is available.

We finish this introduction by mentioning that, in a different context, similar algorithms to some of those introduced in this paper were proposed and analyzed for bounded-degree trees by Jagadish and Sen (2013) and for Bayesian networks in Bello and Honorio (2018).

2. Preliminaries and overview of the results

In this section we first present some definitions. Then we briefly overview our main results.

2.1 Graph-theoretic definitions

A *graph* $G = (V, E)$ is a pair of finite sets $V = V(G)$ and $E = E(G)$ called *vertices* and *edges*, where E is a set of subsets of V of size two. We typically write uv instead of $\{u, v\}$ to denote an edge and our graphs are simple, that is, $u \neq v$. A *subgraph* of G is a graph $G' = (V', E')$ such that $V' \subseteq V$ and $E' \subseteq E$. For $V' \subseteq V$, denote by $G[V']$ the graph $(V', \{uv \in E | u, v \in V'\})$, called the *induced subgraph* of G on V'. If $S \subset V$ we write $G \setminus S$ to denote $G[V \setminus S]$. A path between u and v is a sequence of edges v_0v_1, v_1v_2,...,$v_{k-1}v_k$ with $v_0 = u$ and $v_k = v$. We allow for *empty paths* that consist of a single vertex. Two vertices $u, v \in V$ are *connected* if there is a path between u and v. For $v \in V$, the set $N(v) = \{u \in V | uv \in E\}$ is the *neighborhood* of v, $\deg(v) := |N(v)|$ is its *degree*, $\Delta(G) := \max_{v \in V} \deg(v)$ denotes the *maximum degree* of G, and $|V|$ is the *size* of G.

A graph on $n \geq 3$ vertices is a *cycle* if there is an ordering of its vertices $v_1, \ldots, v_n$, such that $E = \{v_1v_2, \ldots, v_{n-1}v_n, v_nv_1\}$. A graph is *connected* if all $u, v \in V$ are *connected*. A *tree* is a connected graph with no cycles.

A *connected component* of G is a maximal, with respect to inclusion, connected subgraph of G. A set $S \subseteq V$ *separates* $A, B \subseteq V$ *in* G if any path from A to B contains a vertex in S. Then S is called a *separator of* A *and* B in G. When S is of minimum size, it will be called a *minimal separator*. Note that we allow A and B to intersect, in which case $A \cap B$ needs to be contained in every separator of A and B. Denote by $\mathcal{C}^S$ the set of connected components of the graph $G \setminus S$. If S separates two disjoint sets A and B, then for every $u \in A \setminus S$ and $v \in B \setminus S$, u, v lie in two different connected components of $G \setminus S$.

2.2 Partial correlation graphs and graphical models

Let $\Sigma = [\sigma_{ij}]$ be an $n \times n$ symmetric positive definite matrix and denote $K = \Sigma^{-1}$. For a given graph G over vertex set $[n] = \{1, \ldots, n\}$, denote by $\mathcal{M}(G)$ the set of covariance matrices Σ satisfying $K_{ij} = 0$ for all $ij \notin E(G)$. If Σ is a covariance matrix of a Gaussian random vector X then the condition $\Sigma \in \mathcal{M}(G)$ can be equivalently formulated through a set of conditional independence statements because of the equivalence (see Lauritzen (1996))

$$K_{ij} = 0 \quad \iff \quad X_i \perp\!\!\!\perp X_j \mid X_{[n]\setminus\{i,j\}}. \tag{1}$$

For a given Σ, the *concentration graph* $\mathcal{G}(\Sigma) = ([n], E)$ is the graph with $E = \{ij | K_{ij} \neq 0\}$.

Given a vector $x \in \mathbb{R}^n$ and a subset $A \subset [n]$ denote by x_A the subvector of x with entries x_i for $i \in A$. Similarly, for sets $A, B \subseteq [n]$ and a matrix $M \in \mathbb{R}^{n \times n}$, let $M_{A,B}$ denote the restriction of M to rows in A and columns in B. Write M_A for $M_{A,A}$. If Σ is the

covariance of X then $\Sigma_{A,B} = \mathrm{cov}(X_A, X_B)$. In this article we extensively use the following result of Seth Sullivant, Kelli Talaska and Jan Draisma, which translates zero restrictions on a positive definite matrix $K = \Sigma^{-1}$ in terms of minors of Σ.

Theorem 1 (Sullivant et al., 2010, Theorem 2.15) *Let G be a connected graph with vertex set $[n]$. We have $\mathrm{rank}(\Sigma_{A,B}) \leq r$ for all $\Sigma \in \mathcal{M}(G)$ if and only if there is a set $S \subseteq [n]$ with $|S| \leq r$ such that S separates A and B in G. Consequently, $\mathrm{rank}(\Sigma_{A,B}) \leq \min\{|S| :$ S separates A and $B\}$. Moreover, there exists a dense open subset Γ of $\mathcal{M}(G)$ such that equality holds for all matrices in Γ.*

We call Γ a *generic* set. By a slight abuse of terminology, we call covariance matrices Σ in Γ, as well as the corresponding random vectors X, generic.

In this paper we assume that $\mathcal{G}(\Sigma)$ is connected or, equivalently by Theorem 1, that Σ has no zero entries. Without this assumption the problem quickly becomes impossible to solve. For example, whether $\mathcal{G}(\Sigma)$ has zero or one edge can only be decided after seeing the entire covariance matrix.

Assumption 1 *The graph $\mathcal{G}(\Sigma)$ is connected.*

The following well-known characterisation of graphical models over trees will be useful. This result is well known for Gaussian tree models (see, for example, Zwiernik (2019)) but here we prove it formally to emphasize that this is a purely algebraic result.

Lemma 2 *If $\Sigma \in \mathcal{M}(T)$ for a tree T, then for every $i, j \in V$, the normalized entries $\rho_{ij} = \sigma_{ij}/\sqrt{\sigma_{ii}\sigma_{jj}}$ for $i, j \in V$ satisfy the product formula*

$$\rho_{ij} \;=\; \prod_{uv \in \overline{ij}} \rho_{uv}, \tag{2}$$

where $\overline{ij}$ denotes the unique *path between i and j in T. Also, if the normalized entries ρ_{ij} in Σ satisfy (2) for some tree T then $\Sigma \in \mathcal{M}(T)$.*

Proof The first implication can be easily proved by induction on the number of edges in the path $\overline{ij}$ in T. If $\overline{ij}$ contains a single edge then ρ_{ij} trivially satisfies (2). Suppose $\overline{ij} = i - i_1 - \cdots - i_k - j$. Since $\Sigma \in \mathcal{M}(T)$ and i_k separates $\{i, i_k\}$ and $\{i_k, j\}$, Theorem 1 assures that $\mathrm{rank}(\Sigma_{ii_k, i_kj}) \leq 1$, or in other words, $\sigma_{ij}\sigma_{i_ki_k} = \sigma_{ii_k}\sigma_{i_kj}$. Equivalently, $\rho_{ij} = \rho_{ii_k}\rho_{i_kj}$. By the induction hypothesis, ρ_{ii_k} satisfies the path-product formula over the path $\overline{ii_k}$ and so

$$\rho_{ij} \;=\; \rho_{ii_k}\rho_{i_kj} \;=\; \left(\prod_{uv \in \overline{ii_k}} \rho_{uv}\right) \rho_{i_kj} \;=\; \prod_{uv \in \overline{ij}} \rho_{uv}$$

proving that (2) holds.

For the other implication, suppose that the normalized entries of Σ satisfy (2) for some tree T. We want to show that $\Sigma \in \mathcal{M}(T)$, equivalently, $K_{ij} = 0$ for all $ij \notin T$. Without loss of generality, suppose Σ is already standardized so that $\sigma_{ii} = 1$ for $i \in V$. ($\mathcal{M}(T)$ is invariant under $\Sigma \mapsto D\Sigma D$ with D diagonal). Let k be any vertex that separates i and j in T. Let now A/B be any partition of $V \setminus \{k\}$ into two subsets such that $i \in A$, $j \in B$,

and k separates A and B. By (2), $\Sigma_{A,B} = \Sigma_{A,k}\Sigma_{k,B}$. We will show that $K_{A,B} = 0$ and so in particular $K_{ij} = 0$. Direct computations show that the matrix equation

$$\begin{bmatrix} \Sigma_{A,A} & \Sigma_{A,k} & \Sigma_{A,k}\Sigma_{k,B} \\ \Sigma_{k,A} & 1 & \Sigma_{k,B} \\ \Sigma_{B,k}\Sigma_{k,A} & \Sigma_{B,k} & \Sigma_{B,B} \end{bmatrix} \cdot \begin{bmatrix} \mathbf{A} & \mathbf{b} & \mathbf{0} \\ \mathbf{b}^T & c & \mathbf{d}^T \\ \mathbf{0} & \mathbf{d} & \mathbf{E} \end{bmatrix} = I_n \tag{3}$$

has a solution with

$$\mathbf{A} = (\Sigma_{A,A} - \Sigma_{A,k}\Sigma_{k,A})^{-1}, \qquad \mathbf{b} = -(\Sigma_{A,A} - \Sigma_{A,k}\Sigma_{k,A})^{-1}\Sigma_{A,k},$$

$$\mathbf{E} = (\Sigma_{B,B} - \Sigma_{B,k}\Sigma_{k,B})^{-1}, \qquad \mathbf{d} = -(\Sigma_{B,B} - \Sigma_{B,k}\Sigma_{k,B})^{-1}\Sigma_{B,k},$$

$$c = 1 + \Sigma_{k,A}(\Sigma_{A,A} - \Sigma_{A,k}\Sigma_{k,A})^{-1}\Sigma_{A,k} + \Sigma_{k,B}(\Sigma_{B,B} - \Sigma_{B,k}\Sigma_{k,B})^{-1}\Sigma_{B,k}.$$

Since the first term in the product in (3) is the matrix Σ, the second matrix must be K. We conclude that $K_{A,B} = 0$. ■

In this article we assume that the genericity condition of Theorem 1 holds.

Assumption 2 *The matrix $\Sigma \in \mathcal{M}(G)$ is always to be generic, or equivalently, for every $A, B \subseteq V$, $\operatorname{rank}(\Sigma_{A,B}) = \min\{|S| : S \text{ separates } A \text{ and } B\}$.*

This assumption gives us the following important result that translates small sets of covariance queries into information about the underlying concentration graph $\mathcal{G}(\Sigma)$.

Lemma 3 *Under Assumption 2, $\operatorname{rank}(\Sigma_{AC,BC}) = \operatorname{rank}(\Sigma_{A,B})$ if and only if C is a subset of a minimal separator of A and B in $\mathcal{G}(\Sigma)$.*

Here and throughout we use the convention of writing $A \cup B$ as AB in subindices.

Proof By Assumption 2, $\operatorname{rank}(\Sigma_{AC,BC})$ is the size of a minimal separator of $A \cup C$ and $B \cup C$, and $\operatorname{rank}(\Sigma_{A,B})$ is the size of a minimal separator of A and B. Since $\operatorname{rank}(\Sigma_{AC,BC}) = \operatorname{rank}(\Sigma_{A,B})$, there is a minimal separator of $A \cup C$ and $B \cup C$ that is also a minimal separator for A and B. By construction, this separator contains C. ■

Remark 4 *As mentioned in the introduction, vanishing partial correlations do not necessarily translate to conditional independence statements. In the tree case, conditional independence is implied not only for Gaussian and non-paranormal data but also for binary variables, or more generally, in situations where the dependence of adjacent variables in the tree is linear; see Zwiernik (2019) for more details.*

2.3 Overview of the main results

Formulating simplified versions of our main results, we use the notation $\mathcal{O}_\alpha$ to denote that the complexity order contains a factor depending on parameters α. Our first result studies computationally efficient ways to learn a tree.

Theorem 12 (Simplified version) *Suppose $\mathcal{G}(\Sigma) = T = ([n], E)$ is a tree with n vertices and maximum degree $\Delta(T) \leq d$. Then there is an algorithm that outputs the correct tree and, with probability at least $1 - \epsilon$, works in time and query complexity $\mathcal{O}_{\epsilon,d}(n \log^2 n)$.*

In Theorems 15 and 16 we show that these bounds are essentially optimal and the dependence on the maximum degree is essential.

Our second result is for graphs with small 2-connected components and small degree of the block-cut tree; see Section 3 for formal definitions.

Theorem 14 (Simplified version) *Let $G = ([n], E)$ be a graph whose largest 2-connected component has size at most b and whose maximum degree of the block-cut tree is at most d. If $\Sigma \in \mathcal{M}(G)$ is generic, then there is an algorithm that outputs the correct graph and, with probability at least $1 - \epsilon$, works in time and query complexity $\mathcal{O}_{\epsilon,d,b}(n \log^2 n)$.*

Our main result is presented in Sections 4 and 5. We propose a randomized algorithm that is able to recover efficiently the concentration graph $\mathcal{G}(\Sigma)$ as long as this graph has bounded *treewidth* and maximum degree. (In fact, the algorithm remains efficient when both parameters grow slowly with n.) Graphs with bounded treewidth form an important class of sparse graphs that have played a central role in graph algorithms. The class of graphs with small treewidth includes series-parallel graphs, outerplanar graphs, Halin graphs, Apollonian networks, and many others, see Bodlaender (1998) for a general reference. Treewidth has also been known to be an essential parameter in inference and structure recovery for graphical models Chandrasekaran et al. (2012); Kwisthout et al. (2010); Wainwright et al. (2008).

Theorem 32 (Simplified version) *Let $G = ([n], E)$ be a graph with treewidth at most k and maximum degree at most d. If $\Sigma \in \mathcal{M}(G)$ is generic, then there is an algorithm that outputs the correct concentration graph and, with probability at least $1 - \epsilon$, works in time and query complexity $\mathcal{O}_{\epsilon,k,d}(n \log^5 n)$.*

Actually, the algorithm we propose not only reconstructs the concentration graph $\mathcal{G}(\Sigma)$ but also it computes the precision matrix K. Since there are at most kn edges in a graph with treewidth k, there is no contradiction with the stated computational complexity.

3. Recovery of tree-like structures

In this section we discuss in detail procedures for learning trees and graphs with small 2-connected components. A graph is *2-connected* if for any vertex v, $G \setminus v$ is connected. If $V' \subseteq V$ is maximal, with respect to inclusion, such that $G[V']$ is 2-connected, then $G[V']$ is a *2-connected component* or *block* of G.

Theorem 14 below clarifies what we mean by "small 2-connected components." In particular, if the size b of the largest 2-connected component is at most of the order $\log^{1/3} n$, then both the query and computational complexity of the proposed recovery algorithm are of the order $n \log^2 n$.

For a given graph G, let $\mathcal{B}$ be the set of 2-connected components of G and let A be the set of *cut-vertices*, that is, vertices that belong to more than one 2-connected components. The *block-cut tree* bc(G) of G is a bipartite graph on $A \cup \mathcal{B}$ where an edge between $a \in A$

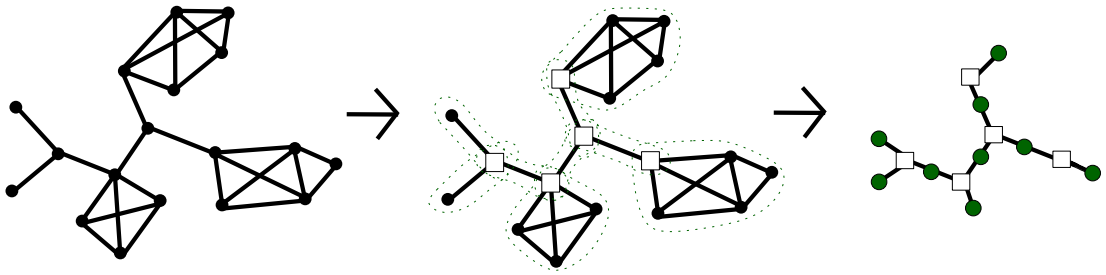


Figure 1: A graph, its cut-vertices (in squares) and its 2-connected components (circled by dotted curves), and the corresponding block-cut tree.

and $B \in \mathcal{B}$ exists if $a \in B$. A block-cut tree is a tree by (Harary, 1969, Theorem 4.4). See Figure 1 for an example of a graph and its block-cut tree.

Both proposed learning procedures—one for recovering trees, another, more general one, for graphs with small 2-connected components—are divide-and-conquer type algorithms. In both methods we first determine a cut vertex that splits the graph into relatively small pieces, identify the pieces, and proceed recursively. Hence, the starting point of our analysis is to identify, at each step of the algorithm, a cut vertex (i.e., a separator of size one) that is balanced.

3.1 Centrality and balanced separators

Let $\mathcal{C}^v$ be the set of connected components of $G \setminus v$ and define

$$c(v) \;=\; \frac{1}{|V|-1} \max_{C \in \mathcal{C}^v} |C|. \tag{4}$$

Denote by v^* a vertex that attains the minimum such value, that is, $v^* = \operatorname{argmin}_{v \in V} c(v)$. If G is a tree, v^* is called a *centroid*. It is a well-known fact (see (Harary, 1969, Theorem 4.3), for instance) that a tree can have at most two centroids and $c(v^*) \leq \frac{1}{2}\frac{|V|}{|V|-1}$.

In the first phase we efficiently find vertices with $c(v) \leq \alpha$ for a fixed $\alpha < 1$. To that end, we introduce a measure of vertex centrality, called *s-centrality* $s(v)$, that can be used as a surrogate for $c(v)$ and whose minimizer can be approximated efficiently. For each $v \in V$, s-centrality is defined as

$$s(v) \;=\; \frac{1}{(|V|-1)^2} \sum_{C \in \mathcal{C}^v} |C|^2. \tag{5}$$

We denote $v^\circ = \operatorname{argmin}_{v \in V} s(v)$.

Lemma 5 *For every vertex $s(v) \leq c(v) \leq \sqrt{s(v)}$. Moreover, $s(v^\circ) \leq c(v^*)$.*

Proof Let $v \in V$ and $\mathcal{C}^v = \{C_1, \ldots, C_m\}$. The first inequality follows from

$$s(v) \;\leq\; \frac{1}{(|V|-1)^2} \sum_{j=1}^{m} |C_j| \max_i |C_i| \;=\; \frac{|V|-1}{(|V|-1)^2} \max_{i \in [m]} |C_i| \;=\; c(v).$$

To show $c(v) \leq \sqrt{s(v)}$, consider the vector $p = (p_1, \ldots, p_m)$ with $p_i = \frac{|C_i|}{|V|-1}$. Then $c(v) = \|p\|_\infty$ and $s(v) = \|p\|_2^2$. The second inequality simply follows from the fact that $\|p\|_\infty \leq \|p\|_2$ for every $p \in \mathbb{R}^m$. To show the last inequality note that $s(v^\circ) \leq s(v^*)$ by the optimality of v° and $s(v^*) \leq c(v^*)$ by the first proved inequality. ■

The procedure `sCentral` outlined in Algorithm 1 finds, with high probability, a vertex $\hat{v}$ with $s(\hat{v})$ close to $s(v^\circ)$. For each vertex $v \in V$ the algorithm approximates $s(v)$ by randomly sampling a few pairs u, w of vertices in $V \setminus \{v\}$ and checking if v separates u and w. By Lemma 2, this can be accomplished by checking if $\Sigma_{uv}\Sigma_{vw} = \Sigma_{uw}\Sigma_{vv}$, or equivalently, if $\det(\Sigma_{uv,vw}) = 0$.

The algorithm outputs a vertex with smallest approximate value of $s(v)$.

Algorithm 1: `sCentral`(V)

```
Parameter: κ;
ŝ(v) := 0 for all v ∈ V;
for all v ∈ V do
    for i = 1 to κ do
        Pick u, w uniformly at random in V \ {v} ;
        if det(Σ_{uv,vw}) ≠ 0 then
            ŝ(v) := ŝ(v) + 1/κ;
Return arg min_v ŝ(v);
```

Proposition 6 *Let $G = (V, E)$ be a graph. The time and query complexity of computing $\hat{v} =$ `sCentral`(V) are $\mathcal{O}(|V|\kappa)$. Moreover, for any $\delta > 0$*

$$\mathbb{P}\left(s(\hat{v}) \geq s(v^\circ) + 2\delta\right) \;\leq\; 2|V| \exp\left(-2\delta^2\kappa\right).$$

Proof The time and query complexity are obtained in a straightforward way. For the second statement note that, for every $v \in V$, $\kappa\hat{s}(v)$ is a binomial random variable with mean $\kappa s(v)$. Hence, by Hoeffding's inequality and the union bound, we obtain

$$\mathbb{P}[\max_v |\hat{s}(v) - s(v)| \geq \delta] \;\leq\; 2|V| \exp\left(-2\delta^2\kappa\right).$$

For $\hat{v} = \arg\min_v \hat{s}(v)$ after running Algorithm 1

$$\begin{aligned}\mathbb{P}[s(\hat{v}) \geq s(v^\circ) + 2\delta] \;&\leq\; \mathbb{P}[s(\hat{v}) - \hat{s}(\hat{v}) + \hat{s}(v^\circ) - s(v^\circ) \geq 2\delta] \\ &\leq\; \mathbb{P}[\max_v\{|\hat{s}(v) - s(v)|\} \geq \delta] \leq 2|V| \exp\left(-2\delta^2\kappa\right),\end{aligned}$$

where the second inequality follows from the fact that $(\hat{s}(v^\circ) - s(v^\circ)) - (\hat{s}(\hat{v}) - s(\hat{v})) \leq 2\max_v |\hat{s}(v) - s(v)|$. ■

We now show that the above procedure finds a good splitting vertex with high probability.

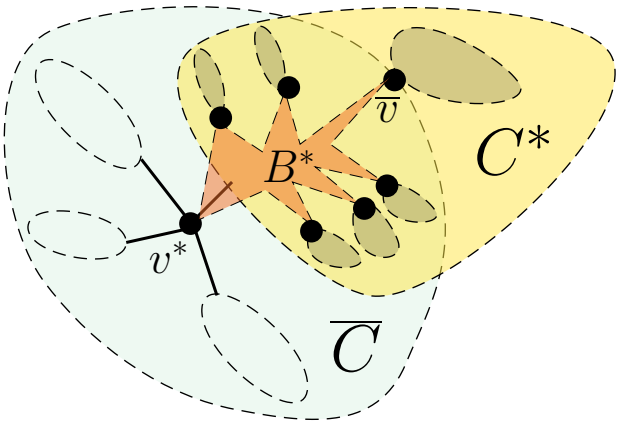


Figure 2: Illustration of the proof of Lemma 9

Proposition 7 *For any graph $G = (V, E)$, if $s(v) < s(v^\circ) + 2\delta$ then*

$$c(v) \;<\; \sqrt{s(v^\circ) + 2\delta} \;\leq\; \sqrt{c(v^*) + 2\delta}.$$

In particular, if $G = (V, E)$ is a tree with $|V| \geq 4$, and $\delta < \frac{1}{6}$, then $c(v) < 1$.

Proof Follows from Lemma 5 and the fact that for trees $c(v^*) \leq \frac{1}{2}\frac{|V|}{|V|-1} \leq \frac{2}{3}$. ■

Proposition 6 and Proposition 7 imply the following result, by fixing $\delta = 1/8$.

Corollary 8 *If $G = (V, E)$ is a tree, $|V| \geq 4$, and $\hat{v} = \texttt{sCentral}(V)$ then*

$$\mathbb{P}\left(c(\hat{v}) > \sqrt{\frac{11}{12}}\right) \;\leq\; 2|V|\exp(-\kappa/32).$$

In Proposition 7 we used the fact that for trees $c(v^*) \leq \frac{2}{3}$ if $|V| \geq 4$. In general, for graphs with small 2-connected components, we rely on the following result.

Lemma 9 *Suppose $G = (V, E)$ is a connected graph. Let d be the maximum degree of the block-cut tree, and let b be the size of the largest 2-connected component of G. If $|V| > db$ then*

$$c(v^*) \;\leq\; 1 - \frac{1}{2d}.$$

Proof If $|V| > db$ then $b < |V|$ and therefore G has a cut vertex. Let C^* be the largest component in $\mathcal{C}^{v^*}$ and let B^* be the 2-connected component of G such that $v^* \in B^*$ and $B^* \setminus \{v^*\} \subseteq C^*$; c.f. Figure 2. We can assume that there exists a cut vertex in $B^* \setminus \{v^*\}$, because otherwise $B^* = C^*$ and clearly $|V| \leq db$. For each cut vertex $v \neq v^*$ in B^*, let k_v be the number of all vertices in the union of all the connected components of $\mathcal{C}^v$ excluding the component containing B^* (grey blobs in Figure 2). By construction, $\sum_{v \neq v^*} k_v = |C^*| - |B^*|$ and there are at most $d-1$ such vertices. If $\overline{v} = \arg\max_{v \neq v^*} k_v$, then $k_{\overline{v}} \geq \frac{1}{d-1}(|C^*| - |B^*|)$.

Denote by $\overline{C}$ the largest connected component of $G \setminus \{\overline{v}\}$. By optimality of v^*, $\overline{C}$ must be equal to the connected component containing the complement of C^*. Thus we have

$$\begin{aligned} |C^*| \;\leq\; |\overline{C}| \;&\leq\; (|V| - 1 - |C^*|) + |B^*| + (|C^*| - |B^*| - k_{\overline{v}}) \leq \\ &\leq\; (|V| - 1 - |C^*|) + |B^*| + \frac{d-2}{d-1}(|C^*| - |B^*|) \end{aligned}$$

Now simple algebra and the fact that $|B^*| \leq b$ allow us to rewrite this inequality as

$$c(v^*) \;\leq\; 1 - \frac{1}{d}\left(1 - \frac{b}{|V| - 1}\right).$$

To prove our inequality, it is enough to show that $\frac{b}{|V|-1} \leq \frac{1}{2}$, which is obvious if $|V| \geq db{+}1$.■

Proposition 7 and Lemma 9 imply that whenever $s(v) < s(v^\circ) + 2\delta$ and $\delta < \frac{1}{4d}$ then $c(v) < 1$. By choosing $\delta = \frac{1}{8d}$, the following corollary follows from Proposition 6.

Corollary 10 *Let $G = (V, E)$ be a graph with $|V| > db$ and let $\hat{v} =$ `sCentral`(V). Then*

$$\mathbb{P}\left(c(\hat{v}) > \frac{\sqrt{4d-1}}{\sqrt{4d}}\right) \;\leq\; 2|V| \exp\left(-\frac{\kappa}{32d^2}\right).$$

3.2 Recovering a tree

In this section we present procedure `ReconstructTree` (Algorithm 2), which efficiently recovers the structure of the tree $T = \mathcal{G}(\Sigma)$. We start the procedure by running `ReconstructTree`$([n])$. The algorithm updates an edge set $\hat{E}$ that is initiated as $\hat{E} = \emptyset$. At each call, if $|V| > 1$, V gets partitioned into sets $V_1, \ldots, V_m$ by procedure `ComponentsTree` (Algorithm 3) and the edge set $\hat{E}$ gets updated. Then, `ReconstructTree` recurses into all the generated sets.

`ComponentsTree`(V) first picks a central vertex $w =$ `sCentral`(V). It then sorts, in descending order, the absolute values of the pairwise correlations $\rho_{uw} = \frac{\Sigma_{uw}}{\sqrt{\Sigma_{uu}\Sigma_{ww}}}$, where $u \in V \setminus \{w\}$, and places them in an ordered list B. For every vertex u in the sorted list, the algorithm checks whether u is separated by w from any of the known neighbours v of w, or equivalently, if $\det(\Sigma_{uw,vw}) = 0$; c.f. Section 3.1. If this happens, then it adds u to the connected component where v belongs after removing w. If this does not happen, then a new connected component is registered corresponding to the vertex u and the edge uv is added in $\hat{E}$. In the end, `ComponentsTree` returns the vertex sets of all such connected components $V_1, \ldots, V_m$. The edges between w and each of the m neighbours in the m connected components are added to $\hat{E}$.

Algorithm 2: `ReconstructTree`(V)

```
if |V| > 1 then
    V_1, ..., V_m ← ComponentsTree(V);
    for i from 1 to m do
        ReconstructTree(V_i);
```

Algorithm 3: `ComponentsTree`(V)

$w \leftarrow$ `sCentral`(V);
$N \leftarrow \emptyset$;
Sort $|\rho_{uw}|$ for $u \in V \setminus \{w\}$ in decreasing order and put them in list B;
for every u in the order of B **do**
  $t := true$;
  for *all* $v \in N$ **do**
    if $\det(\Sigma_{uw,vw}) \neq 0$ **then**
      $V_v \leftarrow V_v \cup \{u\}$; $t := false$;
  if t=*true* **then**
    $\hat{E} \leftarrow \hat{E} \cup \{uw\}$;
    $N \leftarrow N \cup \{u\}$, $V_u \leftarrow \{u\}$;
Return all V_u for $u \in N$;

Proposition 11 *Algorithm 2 is correct, that is, if $\mathcal{G}(\Sigma)$ is a tree T then* `ReconstructTree`$([n])$ *gives $\hat{E} = E(T)$.*

Proof After every call of `ComponentsTree`(V) it holds that $\bigcup_{i=1}^{m} V_i = V \setminus \{w\}$, hence `ReconstructTree`$([n])$ always terminates.

For every $w \in [n]$, either $w =$ `sCentral`(V) in one of the calls of `ComponentsTree` (call such a vertex central), or $\{w\}$ is one of the components $V_1, \ldots, V_m$ that are returned by `ComponentsTree`(V) (call such a vertex terminal).

Initially the algorithm picks a vertex $w =$ `sCentral`$([n])$, which induces the partition of $[n] \setminus \{w\}$, $\mathcal{C}^w = \{V_1, \ldots, V_m\}$. The vertices $u \in [n] \setminus \{w\}$ are examined in descending order with respect to $|\rho_{uw}|$. Let $v \in N(w)$ be adjacent to w and let u be any other other vertex in the same connected component $C \in \mathcal{C}^w$ as v. Then v separates u and w and, in particular, by Lemma 2, $|\rho_{vw}| > |\rho_{uw}|$. This shows that, for any $C \in \mathcal{C}^w$, the vertex v in C which is a neighbour of w comes earlier in the order specified in the algorithm than any other vertex in C. Hence, $v \in N$ (c.f. Algorithm 3) and for all other $u \in C$ it holds that $\det(\Sigma_{uw,vw}) \neq 0$. This shows that in the first call of `ComponentsTree` the algorithm:

(i) adds to $\hat{E}$ the edges between the central vertex w and its neighbours in T,

(ii) assigns all vertices to their connected components in $\mathcal{C}^w$.

Since each $G[V_i]$ is a tree, the same argument can be applied to subsequent calls of `ReconstructTree`. Hence, by induction, these two properties hold at any call of the algorithm `ComponentsTree`. In particular, $\hat{E} \subseteq E(T)$. To show the opposite inclusion first note that if $uv \in E(T)$ and u or v is central, then $uv \in \hat{E}$ by (i). Moreover, if u, v are both terminal, then there is some call of `ComponentsTree` that places them in different sets V_i. Then by (ii), there is no edge uv in E. ■

Subroutine `sCentral` is a probabilistic component of the algorithm that is essential to obtain good complexity bounds.

Theorem 12 *Suppose $\mathcal{G}(\Sigma)$ is a tree $T = ([n], E)$ with maximum degree $\Delta(T) \le d$. Fix $\epsilon < 1$ and define $\kappa = \lceil 32 \log\left(\frac{2n^2}{\epsilon}\right) \rceil$ to be the parameter of Algorithm 1. Then, with probability at least $1-\epsilon$, Algorithm 2 requires time and queries of order*

$$\mathcal{O}\left(n \log(n) \max\left\{\log\left(\frac{n}{\epsilon}\right), d\right\}\right).$$

Proof First we analyze the complexity of one call of $\texttt{ComponentsTree}(V)$. By Proposition 6, the call of $\texttt{sCentral}(V)$ takes time and queries of the order $\mathcal{O}\big(|V|\kappa\big)$. We then query $|V|$ pairwise correlations and sort them, which takes time $\mathcal{O}(|V| \log |V|)$. Partitioning V into sets V_i takes time and queries both of order $\mathcal{O}(d|V|)$ since $|N| \le d$. For all calls of $\texttt{ReconstructTree}(V_i)$ in a recursion level (i.e., distance from the first $\texttt{ReconstructTree}$ call in the recursion tree), it holds that $V_i \cap V_j = \emptyset$. Hence, in each recursion level the time complexity is of order $\mathcal{O}(n \log n + n\kappa + nd)$ and the query complexity is of order $\mathcal{O}(n\kappa + nd)$.

Assume first that $\hat{v} = \texttt{sCentral}(V)$ satisfies $c(\hat{v}) \le \alpha := \sqrt{11/12}$ in each call with $|V| \ge 4$. In this case the recursion depth is at most $\log_{1/\alpha}(n) + 4$ and, overall, the algorithm has time complexity $\mathcal{O}\big(n \log(n)(\log(n) + \kappa + d)\big)$ and query complexity $\mathcal{O}\big(n \log(n)(\kappa + d)\big)$. Since $\kappa = \mathcal{O}(\log(n/\epsilon) + \log d)$, the announced bounds follow.

It remains to show that, with the given choice of κ, with probability at least $1 - \epsilon$ we get that $c(\hat{v}) \le \alpha$ in each call with $|V| \ge 4$. By Corollary 8, in a single call the probability that $c(\hat{v}) > \alpha$ is at most $2|V| \exp(-\kappa/32)$, which is further bounded by $2n \exp(-\kappa/32)$. As $\texttt{ReconstructTree}([n])$ runs, the procedure $\texttt{sCentral}$ is called at most n times, which is the total number of available vertices. From the union bound, the probability that in at least one call $\hat{v} = \texttt{sCentral}(V)$ satisfies $c(\hat{v}) > \alpha$ is at most $2n^2 \exp(-\kappa/32)$. Demanding the latter to be at most ϵ, we obtain the indicated value for κ and the desired result. ■

3.3 Graphs with small blocks

We now present an algorithm that recovers concentration graphs with small 2-connected components and small maximum degree of the block-cut tree. The procedure $\texttt{ReconstructSB}$ (Algorithm 4) takes as input a vertex set V and, like in the tree case, updates the global variable $\hat{E}$, which is initially set as $\hat{E} := \emptyset$. If V is small enough, that is, $|V| \le db$, then the algorithm reconstructs the induced graph over V by directly inverting the matrix Σ_V. Otherwise it calls $\texttt{ComponentsSB}$, which first finds a vertex $w = \texttt{sCentral}(V)$ and returns sets $C \cup \{w\}$ for all $C \in \mathcal{C}^w$. This part of the algorithm is similar to $\texttt{ComponentsTree}(V)$, but the edges incident with w are not recovered at this stage.

Proposition 13 *Algorithm 4 is correct, that is, if $\Sigma \in \mathcal{M}(G)$ and Σ is generic then $\texttt{ReconstructSB}([n])$ gives $\hat{E} = E(G)$.*

Proof Assume we are on the first call of $\texttt{ReconstructSB}$. If $n \le db$ then the algorithm outputs $\mathcal{G}(\Sigma)$, which, by definition, is the correct graph. If $n > db$, then G contains a cut vertex, so with probability 1 a cut vertex w will be found by $\texttt{sCentral}([n])$. Let $C_1, \ldots, C_m$ be the connected components of $G \setminus \{w\}$. The sets V_i produced by this call of $\texttt{sCentral}([n])$ correspond to the sets $C_1 \cup \{w\}, \ldots, C_m \cup \{w\}$. This is clear by Lemma 3: a vertex u belongs

Algorithm 4: `ReconstructSB`(V)

```
if |V| > db then
    V_1, ..., V_m ← ComponentsSB(V);
    for i from 1 to m do
        ReconstructSB(V_i);
else
    Ê ← Ê ∪ E(G(Σ_{V,V}));
```

Algorithm 5: `ComponentsSB`(V)

```
w ← sCentral(V);
N := ∅;                                  // contains one vertex from each C ∈ C^w
for all u ∈ V \ {w} do
    if there exists v ∈ N, det(Σ_{uw,vw}) ≠ 0 then
        V_v ← V_v ∪ {u};
    else
        V_u ← {u};
        N ← N ∪ {u};
return V_u ∪ {w} for all u ∈ N;
```

to the same connected component as v in $G \setminus \{w\}$ if and only if w does not separate u and v in G. Note also that for any $A, B \subset V_i$ any minimal separator of A and B is contained in V_i. In particular, by Theorem 1,

(i) For every V_i, the edge-set of $G[V_i]$ is the same as the edge-set of the graph of the marginal distribution, $\mathcal{G}(\Sigma_{V_i})$.

By induction, statement (i) holds for every call of `ReconstructSB`. ■

Theorem 14 *Let $G = ([n], E)$ be a graph with maximum degree of the block-cut tree bounded by d, and let b be the size of the largest 2-connected component. Fix $\epsilon < 1$ and define $\kappa = \lceil 32d^2 \log(\frac{2dn}{\epsilon}) \rceil$ to be the parameter of Algorithm 1. If $\Sigma \in \mathcal{M}(G)$ and Σ is generic, then, with probability at least $1 - \epsilon$, Algorithm 4 runs with query and time complexity of order*

$$\mathcal{O}\left(d^3 n \log(n)\left(\log\left(\frac{n}{\epsilon}\right) + b^2\right)\right) \quad \text{and} \quad \mathcal{O}\left(d^3 n \log(n)\left(\log\left(\frac{n}{\epsilon}\right) + db^3\right)\right)$$

Proof First we analyze the complexity of one call of `ComponentsSB`(V). If $|V| \leq db$ then $\mathcal{G}(\Sigma_{V,V})$ is obtained. Since matrix inversion takes at most cubic time, the time and queries required are $\mathcal{O}(d^3b^3)$ and $\mathcal{O}(d^2b^2)$ respectively. If $|V| > db$ in `ReconstructSB`(V), then `ComponentsSB`(V) is called, which calls `sCentral`(V). By Proposition 6, the call of `sCentral`(V) takes time and queries of the order $\mathcal{O}\big(|V|\kappa\big)$. The latter provides the splitting

vertex w and then V is split into at most d sets $U_i = V_i \cup \{w\}$. This last step takes $\mathcal{O}(|V|d)$ queries and time. Hence, each call $\texttt{ReconstructSB}(V)$ requires

$$\mathcal{O}\left(|V|\kappa + |V|d + d^2b^2\right) \quad \text{and} \quad \mathcal{O}\left(|V|\kappa + |V|d + d^3b^3\right)$$

queries and time, respectively.

Let $U_1, \ldots, U_r$ be the sets on which the algorithm recurses on the i-th level of the recursion tree. By construction, these sets are not disjoint and, for each $v \in V$, at most d copies of it are created during the algorithm. Hence, in each recursion level there are at most nd vertices, implying that $\sum_{i=1}^{r} |U_i| \leq nd$ and that $r \leq nd$. Using the complexity bounds for a single call of $\texttt{ReconstructSB}$, we get that any recursion level in the recursion tree requires

$$\mathcal{O}\left(nd\kappa + nd^3b^2\right) \quad \text{and} \quad \mathcal{O}\left(nd\kappa + nd^4b^3\right)$$

queries and time, respectively.

Assume first that $\hat{v} = \texttt{sCentral}(V)$ satisfies $c(\hat{v}) \leq \alpha := \sqrt{4d-1}/\sqrt{4d}$ in each call. In this case the recursion depth is at most $\log_{1/\alpha}(n)$ and overall the algorithm requires

$$\mathcal{O}\left(\frac{dn\log(n)}{\log(1/\alpha)}\left(\kappa + d^2b^2\right)\right) \quad \text{and} \quad \mathcal{O}\left(\frac{dn\log(n)}{\log(1/\alpha)}\left(\kappa + d^3b^3\right)\right)$$

queries and time, respectively. Since $\kappa = \mathcal{O}(d^2\log(n/\epsilon))$ and $1/\log(1/\alpha) \leq 20d$, we obtain the expressions in the statement of the theorem.

It remains to show that with the given choice of κ, with probability at least $1-\epsilon$, we get that $c(\hat{v}) \leq \alpha$ in each call. By Corollary 10, in a single call the probability that $c(\hat{v}) > \alpha$ is at most $2|V|\exp\left(-\frac{\kappa}{32d^2}\right)$. As $\texttt{ReconstructSB}([n])$ runs, the procedure $\texttt{sCentral}$ is called at most dn times. From the union bound, the probability that in at least one call $\hat{v} = \texttt{sCentral}(V)$ satisfies $c(\hat{v}) > \alpha$ is at most $2dn\exp(-\kappa/(32d^2))$. This is at most ϵ for the indicated value of κ. ■

3.4 Lower bounds

In this section we show that the result of Theorem 12 is optimal up to logarithmic factors, in the sense that one cannot reconstruct trees with maximum degree d with less than $\Omega(dn)$ covariance queries.

Let $\mathcal{X}$ be the class of $n \times n$ covariance matrices whose concentration graph is a tree. We write $T(\Sigma)$ for the tree induced by $\Sigma \in \mathcal{X}$. We also denote by $\mathcal{X}_d$ the class of covariance matrices whose concentration graph is a tree of maximum degree bounded by d. In our construction we use the characterization of the class $\mathcal{X}$ given in Lemma 2.

We first prove that any algorithm that recovers the correct tree (without any restriction on the maximum degree) needs to access the covariance oracle $\Omega(n^2)$ times.

In order to formalize such a statement, let $\mathcal{A}_k$ be the class of all randomized adaptive algorithms that query the covariance oracle at most k times (*adaptive* means that the

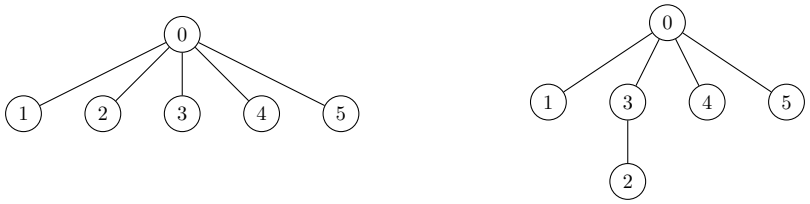


Figure 3: Illustration of the construction in the proof of Theorem 15 with $n = 6$, $\{I, J\} = \{2, 3\}$, and $B = 0$ (left), $B = 1$ (right).

algorithms are allowed to dynamically choose their queries). An algorithm $A \in \mathcal{A}_k$ outputs the tree $\mathcal{T}(A)$. The probability of error of algorithm A for $\Sigma \in \mathcal{X}$ is denoted by

$$P(A, \Sigma) = \mathbb{P}\left\{\mathcal{T}(A) \neq T(\Sigma)\right\} \ ,$$

where the probability is with respect to the randomization of the algorithm A. The quantity of interest is the *minimax risk*

$$R(\mathcal{A}_k, \mathcal{X}) = \inf_{A \in \mathcal{A}_k} \sup_{\Sigma \in \mathcal{X}} P(A, \Sigma) \ .$$

$R(\mathcal{A}_k, \mathcal{X})$ expresses the worst-case probability of error of the best algorithm that takes at most k covariance queries.

Theorem 12 implies that there exists a constant $c > 0$ such that, for every $\epsilon > 0$, we have $R(\mathcal{A}_k, \mathcal{X}_d) \leq \epsilon$ whenever $k > cn \log(n)\,(d + \log(n/\epsilon))$. In this section we prove that this upper bound is tight up to logarithmic factors.

We start with the case $d = n - 1$ (i.e., no restriction on the maximum degree) since this simpler case already contains the main ideas. The lower bound for $R(\mathcal{A}_k, \mathcal{X}_d)$ follows by a simple adjustment.

Theorem 15 *For all $k \leq \binom{n}{2}$,*

$$R(\mathcal{A}_k, \mathcal{X}) \geq \frac{1}{2} - \frac{k}{(n-1)^2} \ .$$

In particular, $R(\mathcal{A}_k, \mathcal{X}) \geq 1/2 - o(1)$ whenever $k = o(n^2)$.

Proof In order to prove the lower bound, we define a probability distribution $\mathcal{D}$ on the set $\mathcal{X}$ and write

$$R(\mathcal{A}_k, \mathcal{X}) \geq \inf_{A \in \mathcal{A}_k} \mathbb{E}_{\Sigma \sim \mathcal{D}} P(A, \Sigma) \ .$$

Next we specify how a random symmetric matrix Σ, distributed according to $\mathcal{D}$, is generated. Σ is defined by a collection of independent random variables: let B be a Bernoulli random variable with parameter $1/2$, let $U_1, \ldots, U_{n-1}$ be independent random variables, uniformly distributed on $[0, 1]$, and let I, J be different indices in $[n-1]$, uniformly distributed over all $(n-1)(n-2)$ such pairs. Then $\Sigma = \Sigma(B, U_1, \ldots, U_{n-1}, I, J)$ is defined as follows; c.f. Figure 3. (We index the n columns and rows from 0 to $n-1$.)

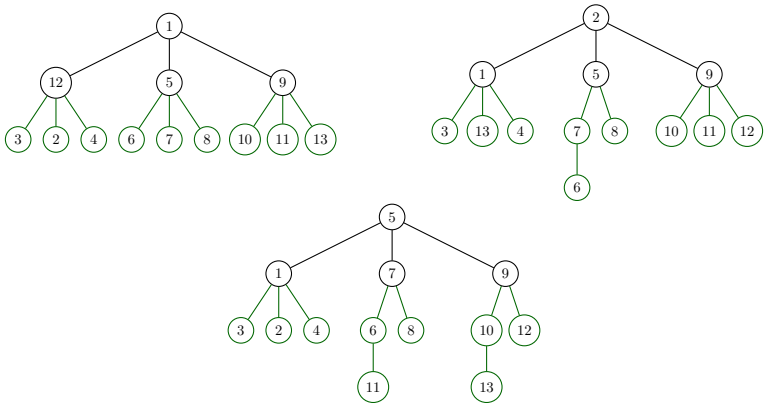


Figure 4: Three trees made from a ternary tree, by moving at most one lower level leaf from its parent to one of its siblings.

• $\Sigma_{i,i} = 1$ for all $i = 0, \ldots, n-1$.
• Regardless of B, I, J, we have $\Sigma_{0,i} = U_i$ for all $i = 1, \ldots, n-1$.
• If $B = 0$, then $\Sigma_{i,j} = U_i U_j$ for all $i, j \in \{1, \ldots, n-1\}$, $i \neq j$. Note that in this case, by Lemma 2, the concentration graph is a star with vertex 0 as a center (and therefore indeed $\Sigma \in \mathcal{X}$).
• If $B = 1$, then $\Sigma_{i,j} = U_i U_j$ for all $i, j \in \{1, \ldots, n-1\}$ such that $i \neq j$ and $\{i, j\} \neq \{I, J\}$. Moreover, $\Sigma_{I,J} = \min(U_I, U_J)/\max(U_I, U_J)$. In this case, again by Lemma 2, the concentration graph is a tree in which vertex 0 has degree $n-2$, every vertex $i \notin \{I, J\}$ has degree 1 and is attached to vertex 0, and vertices $0, I, J$ form a path such that, if $U_I < U_J$, then J is the middle vertex and if $U_I > U_J$, then I is the middle vertex.

Clearly, regardless of what the algorithm A is, it is unable to distinguish between $\Sigma(0, U_1, \ldots, U_{n-1}, I, J)$ and $\Sigma(1, U_1, \ldots, U_{n-1}, I, J)$ before the entry $\Sigma_{I,J}$ is queried. (No other entry of Σ provides any information about $\Sigma_{I,J}$.) In other words, if B, $U_1, \ldots, U_{n-1}$ and I, J are fixed and $\Sigma = \Sigma(B, U_1, \ldots, U_{n-1}, I, J)$ then

$$P(A, \Sigma) \;\geq\; \frac{1}{2}\,\mathbb{E}\left(\mathbb{1}_{(I,J) \text{ is not queried}} | (I, J)\right).$$

Thus, for any algorithm A,

$$\mathbb{E}_{\Sigma\sim\mathcal{D}} P(A, \Sigma) \;\geq\; \frac{1}{2\binom{n-1}{2}} \sum_{\{i,j\}\subset[n-1]:i\neq j} \mathbb{E}\left(\mathbb{1}_{(i,j) \text{ is not queried}} | (I, J) = (i, j)\right) \;\geq\; \frac{\binom{n-1}{2} - k}{2\binom{n-1}{2}}$$

by symmetry, proving the theorem. ∎

For the class of covariance matrices $\mathcal{X}_d$ whose concentration graph is a tree with maximum degree bounded by d, we have the following bound. Its proof is similar to that of Theorem 15. To avoid repetitions, we do not detail the proof. The only difference is that the class of trees that support the distribution $\mathcal{D}$ now includes the complete d-ary tree of

height h of $n = (d^{h+1} - 1)/(d - 1)$ vertices and its modifications such that, in each d-ary branch at the leaf level, one can remove a leaf and attach it to another one of the same branch (see Figure 4 for such instances, made from a ternary tree).

Theorem 16

$$R(\mathcal{A}_k, \mathcal{X}_d) \geq \frac{1}{2}\left(1 - o(1)\right)$$

whenever $k = o(nd)$.

4. Separators in bounded treewidth graphs

In the next two sections we deal with the main results of the paper. We show that a large and important class of sparse concentration graphs can be reconstructed efficiently with $O(n\,polylog\,n)$ correlation queries. The class includes all graphs with bounded treewidth and bounded maximum degree.

The algorithm we propose is a divide-and-conquer algorithm. The main idea is that, once one finds a small set of vertices (a separator) whose removal decomposes the graph into small connected components, and these components are identified, one may recurse in these components. The nontrivial task is to find such separators efficiently.

Our algorithm starts by taking a random sample W of the vertices, of size proportional to the treewidth of G. Then we find a separator of W of size at most $k + 1$ that splits the vertices of W into two sets of comparable size. We argue that, with high probability, such a separator exists. We also prove that this separator is a balanced separator of the entire vertex set. Removal of this separator decomposes the graph into connected components of significantly reduced size. We identify these components using a linear number of queries. Then the algorithm recurses into each of the components. In this section we discuss the first splitting of the graph. How to subsequently recurse into the smaller subsets is described in detail in Section 5.

A *tree decomposition* of a graph $G = (V, E)$ is a tree T with vertices $B_1, \ldots, B_m$ where $B_i \subseteq V$ satisfy

1. The union of all sets B_i equals V.

2. If B_i and B_j both contain v, then all vertices B_k of T in the unique path between B_i and B_j contain v as well.

3. For every edge uv in G, there is B_i that contains both u and v.

The *width* of a tree decomposition is the size of its largest set B_i minus one. The *treewidth* of a graph G, denoted $\mathrm{tw}(G)$, is the minimum width among all possible tree decompositions of G.

The key property of bounded-treewidth graphs is that they have small "balanced" separators. To define a balanced separator, we generalize the notion of centrality (4) for any set $S \subset V$ by writing

$$c(S) \;=\; \frac{1}{|V \setminus S|} \max_{C \in \mathcal{C}^S} |C| \;, \tag{6}$$

where recall that $\mathcal{C}^S$ is the set of connected components of the graph induced by $V \setminus S$. We say that a separator S is *balanced* when $c(S) \leq 0.93$. We start by noting that every graph with bounded treewidth has a small balanced separator, see, for example, (Bodlaender, 1998, Theorem 19).

Proposition 17 *If* $\mathrm{tw}(G) \leq k$ *then* G *has a separator* S *such that* $|S| \leq k+1$ *and* $c(S) \leq \frac{1}{2}\frac{|V|-k}{|V|-(k+1)}$.

Remark 18 *By (Bodlaender, 1998, Lemma 11), if* $\mathrm{tw}(G) \leq k$ *then* $\mathrm{tw}(H) \leq k$ *for every subgraph* H *of* G. *In particular, one can recursively split a graph into subgraphs of small treewidth using small balanced separators.*

4.1 Finding a separator of two sets

Let $G = (V, E)$ be a graph and let $\Sigma \in \mathcal{M}(G)$ be generic. We give an algorithm that finds a minimal separator of two subsets $A, B \subset V$. By Assumption 2, the size of such a minimal separator is $r := \mathrm{rank}(\Sigma_{A,B})$. Denote by $\mathcal{S}(A, B)$ the set of all minimal separators of A and B in G. Denote by U the set of all vertices that lie in some minimal separator in $\mathcal{S}(A, B)$.

Lemma 19 *A vertex* $v \in V$ *lies in* U *if and only if* $\mathrm{rank}(\Sigma_{Av,Bv}) = r$.

Proof This follows immediately from Lemma 3. ■

Lemma 19 together with Lemma 3 give a simple and efficient procedure to find an element in $\mathcal{S}(A, B)$, detailed in Algorithm 6.

Algorithm 6: `ABSeparator`(A, B)

```
U ← ∅;
r = rank(Σ_{A,B});
forall v ∈ V do
    if rank(Σ_{Av,Bv}) = r then
        U ← U ∪ {v};
C ← {v_0} for some v_0 ∈ U;
forall u ∈ U \ {v_0} do
    if rank(Σ_{ACu,BCu}) = r then
        C ← C ∪ {u} ;
return C;
```

Proposition 20 *Let* $G = (V, E)$ *be a graph and let* $\Sigma \in \mathcal{M}(G)$ *be generic. For any* $A, B \subset V$ *with* $M = \max\{|A|, |B|\}$, *Algorithm 6 finds a minimal separator of* A *and* B *with query complexity* $\mathcal{O}(|V|M^2)$ *and computational complexity* $\mathcal{O}(|V|M^3)$.

Proof By Lemma 19, the first loop finds the set U of all vertices that lie in a minimal separator of A and B. This loop has $\mathcal{O}(|V|M^2)$ and $\mathcal{O}(|V|M^3)$ query and computational complexity, respectively.

We now take an arbitrary vertex $v_0 \in U$ and show that the second loop of the algorithm finds a minimal separator of A and B that contains v_0. Start with $C = \{v_0\}$ and note that, since $v_0 \in U$, there exists $S \in \mathcal{S}(A, B)$ containing v_0. In each iteration of the second loop we add $u \in U \setminus \{v_0\}$ to C if $\operatorname{rank}(\Sigma_{ACu,BCu}) = r$. Therefore, by Lemma 3, we update $C \leftarrow C \cup \{u\}$ if and only if there exists (not necessarily unique) $S \in \mathcal{S}(A, B)$ such that $C \cup \{u\} \subseteq S$. By Assumption 2, $|S| = r$ and so, if $|C \cup \{u\}| = r$ then $S = C \cup \{u\}$ and the rank condition will *not* be satisfied for the subsequent vertices in the loop (showing correctness of the algorithm). If $|C \cup \{u\}| < r$ then $C \cup \{u\}$ is a strict subset of S and all the remaining vertices in $S \setminus (C \cup \{u\})$ appear in the second loop *after* u. Applying this argument recursively, we conclude correctness of the algorithm.

Since $|U| \leq |V|$ and $r \leq M$, the number of queries and computational complexity of the second loop are $\mathcal{O}(|V|M^2)$ and $\mathcal{O}(|V|M^3)$, respectively, which concludes the proof. ■

4.2 Balanced separators in G

In Section 4.1 we provided an efficient procedure that finds a separator for a given pair of sets $A, B \subset V$. In this section we show how to construct such a pair of small sets so that the obtained separator is, with high probability, a balanced separator for the entire graph G.

Our approach to finding a balanced separator is to base the search on a random subset $W \subset V$ of size m which can be handled within our computational budget. To argue why our randomization works and guide the choice of the parameter m we rely on VC-theory initiated by Vapnik and Chervonenkis (1971). Let $\mathcal{F}_S$ be the set of all connected components in $\mathcal{C}^S$ and their complements in $V \setminus S$. Write

$$\mathcal{F}_k \;:=\; \bigcup_{S:|S|\leq k} \mathcal{F}_S \;, \tag{7}$$

that is, $C \in \mathcal{F}_k$ if it is a connected component of $G \setminus S$ for some S with at most k elements, or C is a union of all but one such components.

Definition 21 *A set $W \subseteq V$ is a* δ-sample *for $\mathcal{F}_k$ if for all sets $C \in \mathcal{F}_k$,*

$$\frac{|C|}{|V|} - \delta \;\leq\; \frac{|W \cap C|}{|W|} \;\leq\; \frac{|C|}{|V|} + \delta. \tag{8}$$

We now present conditions that assure that a uniformly random sample W from the vertex set V is a δ-sample with high probability. A subset $W \subset V$ is *shattered* by $\mathcal{F}_k$ if $W \cap \mathcal{F}_k \;=\; \{W \cap C :\; C \in \mathcal{F}_k\}$ is the set of all subsets of W. Define the *VC-dimension* of $\mathcal{F}_k$, denoted by $\operatorname{VC}(\mathcal{F}_k)$, to be the maximal size of a subset shattered by $\mathcal{F}_k$. The following follows from the classical Vapnik-Chervonenkis inequality (see Devroye and Lugosi (2000) for a version that implies the constants shown here):

Theorem 22 *Suppose that $\operatorname{VC}(\mathcal{F}_k) = r$, $\delta > 0$, and $\tau \leq 1/2$. A set W obtained by sampling m vertices from V uniformly at random, with replacement, is a δ-sample of $\mathcal{F}_k$ with probability at least $1 - \tau$ if*

$$m \;\geq\; \max\left(\frac{10r}{\delta^2}\log\left(\frac{8r}{\delta^2}\right), \frac{2}{\delta^2}\log\left(\frac{2}{\tau}\right)\right) \;. \tag{9}$$

A key property of the set $\mathcal{F}_k$ is that its VC-dimension is bounded by a linear function of the treewidth k.

Lemma 23 (Feige and Mahdian (2006)) *Let $G = (V, E)$ be a graph and let $\mathcal{F}_k$ for $k \geq 1$ be the set defined in (7). Then $\mathrm{VC}(\mathcal{F}_k) \leq 11 \cdot k$.*

Remark 24 *The statement of this lemma in Feige and Mahdian (2006) uses a universal constant. Their proof however allows one to specify this constant to be 11.*

The next result shows that a δ-sample admits a balanced separator.

Proposition 25 *Let $G = (V, E)$ be such that $\mathrm{tw}(G) \leq k$, and let $W \subset V$ be a δ-sample of $\mathcal{F}_{k+1}$ satisfying $|W| \geq 6(k+1)$. If $\delta \leq \frac{1}{24}$ then we can partition W into two sets A, B such that $|A|, |B| \leq \frac{2|W|}{3}$ and a minimal separator S of A and B has at most $k+1$ elements. Moreover, for any such partition, $\max(|A \setminus S|, |B \setminus S|) \leq \frac{4}{5}|W \setminus S|$.*

In the latter proposition we refer to $\mathcal{F}_{k+1}$ and not $\mathcal{F}_k$, in order to be consistent with Proposition 17.
Before we prove this result we formulate two useful lemmas. The first one is merely a simple observation.

Lemma 26 *Let $U = \bigcup_{i=1}^{d} C_i$ be a partition of U into disjoint sets such that $|C_1| \geq \cdots \geq |C_d| \geq 0$. If $|C_1| \leq \alpha|U|$ for $\alpha \geq \frac{2}{3}$ then $(1-\alpha)|U| \leq \sum_{i=1}^{t} |C_i| \leq \alpha|U|$ for some t.*

To state the second lemma, write $\lambda = 1 - \frac{k+1}{|V|}$ and note that $\frac{|V \setminus S|}{|V|} \geq \lambda$ for all S such that $|S| \leq k+1$.

Lemma 27 *Suppose $W \subset V$ is a δ-sample for $\mathcal{F}_{k+1}$ and let $S \subset V$ with $|S| \leq k+1$. If $C \in \mathcal{C}^S$, then*

$$\frac{\lambda c(S) - \delta}{\lambda + 2\delta} \;\leq\; \frac{|W \cap C|}{|W \setminus S|} \;\leq\; \frac{\lambda c(S) + \delta}{\lambda - 2\delta}. \tag{10}$$

Proof Using (8), we get

$$\frac{|W \cap C|}{|W \setminus S|} \;\leq\; \left(\frac{|C|}{|V|} + \delta\right) \frac{|W|}{|W \setminus S|} \;\leq\; \left(c(S)\frac{|V \setminus S|}{|V|} + \delta\right) \frac{|W|}{|W \setminus S|}.$$

To bound the last expression, let $\bar{C} = V \setminus (C \cup S)$. Since $C, \bar{C} \in \mathcal{F}_{k+1}$, we get

$$\frac{|W \setminus S|}{|W|} \;=\; \frac{|C \cap W|}{|W|} + \frac{|\bar{C} \cap W|}{|W|} \;\geq\; \left(\frac{|C|}{|V|} + \frac{|\bar{C}|}{|V|} - 2\delta\right) \;=\; \frac{|V \setminus S|}{|V|} - 2\delta.$$

A similar argument gives an upper bound for $\frac{|W \setminus S|}{|W|}$, which after taking reciprocals gives

$$\frac{1}{\frac{|V \setminus S|}{|V|} + 2\delta} \;\leq\; \frac{|W|}{|W \setminus S|} \;\leq\; \frac{1}{\frac{|V \setminus S|}{|V|} - 2\delta} \tag{11}$$

This gives the upper bound in (10) because

$$\frac{|W \cap C|}{|W \setminus S|} \;\leq\; \frac{c(S)\frac{|V\setminus S|}{|V|} + \delta}{\frac{|V\setminus S|}{|V|} - 2\delta} \;\leq\; \frac{\lambda c(S) + \delta}{\lambda - 2\delta},$$

where the last inequality follows by the fact that the middle expression is a decreasing function of $\frac{|V\setminus S|}{|V|}$ and $\frac{|V\setminus S|}{|V|} \geq \lambda$. This establishes the upper bound in (10). The lower bound follows by similar arguments. ■

Proof [Proof of Proposition 25] Let S^* be a minimizer of $c(S)$ among all $S \subset V$ such that $|S| \leq k+1$. By Proposition 17 $c(S^*) \leq \frac{1}{2}\frac{|V|-k}{|V|-(k+1)}$, which is further bounded by $11/20$ if $|V| \geq 6(k+1)$. By Lemma 27, if $C \in \mathcal{C}^{S^*}$ then

$$\frac{|W \cap C|}{|W \setminus S^*|} \;\leq\; \frac{\frac{11}{20}\lambda + \delta}{\lambda - 2\delta}.$$

The right-hand side is an increasing function of δ and the maximum for $\delta \leq \frac{1}{24}$ is $(\frac{11}{20}\lambda + \frac{1}{24})/(\lambda - \frac{1}{12})$, which is bounded by $\frac{2}{3}$ because $\lambda \geq \frac{5}{6}$ (use $|V| \geq |W| \geq 6(k+1)$). This shows that $W \setminus S^*$ can be partitioned into disjoint subsets $W \cap C$ for $C \in \mathcal{C}^{S^*}$ all of size at most $\frac{2}{3}|W \setminus S^*|$. By Lemma 26, we can group these sets into two groups A' and B' each of size at most $\frac{2}{3}|W \setminus S^*|$. To show the first claim let A, B be any two sets partitioning W that satisfy $A \setminus S^* = A'$, $B \setminus S^* = B'$. We next show that there is a choice of A, B that gives $\max(|A|, |B|) \leq \frac{2}{3}|W|$. This is done by allocating the elements of $W \cap S^*$ in a balanced way between A' and B' so that both A' and B' get at most $\frac{2}{3}$ of the elements in $W \cap S^*$. This can be always done if $W \cap S^*$ has at least two elements. If $W \cap S^*$ is empty, the statement is trivial. If $|W \cap S^*| = 1$ we consider two cases (i) $|A'| < |B'|$ and (ii) $|A'| = |B'|$. In case (i) we allocate the element in $W \cap S^*$ to A'. In that case

$$|A| \;=\; |A'| + 1 \;\leq\; |B'| \;=\; |B| \;\leq\; \frac{2}{3}|W \setminus S^*| \;\leq\; \frac{2}{3}|W|.$$

In case (ii), we again allocate the element in $W \cap S^*$ to A' and use the fact that $|A'| = |B'| = \frac{1}{2}|W \setminus S^*|$, which gives

$$|B| \;\leq\; |A| \;=\; |A'| + 1 \;=\; \frac{1}{2}|W \setminus S^*| + 1 \;=\; \frac{1}{2}|W| + \frac{1}{2} \;\leq\; \frac{2}{3}|W|,$$

where the last inequality holds always if $|W| \geq 3$. This proves the first claim.

To show the second claim, assume $\max(|A|, |B|) \leq \frac{2}{3}|W|$. Since $|W| \geq 6(k+1)$ we have $|W \setminus S| \geq 5(k+1) \geq 5|W \cap S|$. Now

$$\max(|A \setminus S|, |B \setminus S|) \leq \frac{2}{3}|W| \leq \frac{2}{3}(|W \setminus S| + |W \cap S|) \;\leq\; \frac{2}{3}\left(1 + \frac{1}{5}\right)|W \setminus S|,$$

which completes the argument. ■

4.3 Separating and splitting

We now propose a procedure `Separator` that finds a balanced separator in G. When a separator is found, decomposing the graph into connected components is straightforward and is given in the procedure `Components`.

The procedure starts by choosing a sample $W \subset V$. Then the algorithm looks for a partition of W into two sets A, B so that $|A|, |B| \leq \frac{2}{3}|W|$ and the rank of $\Sigma_{A,B}$ is small. In Proposition 28 we argue why such a partition exists with high probability. Then the algorithm uses ideas of Section 4.1 to efficiently find a minimal separator S of A and B in G; by construction $|S| = \mathrm{rank}(\Sigma_{A,B})$. At this moment a purely deterministic part of the process begins. Given S, the algorithm decomposes V into connected components in $\mathcal{C}^S$. This is done using rank conditions like in the tree-like case.

Algorithm 7: `Separator`

Pick a set W by taking m vertices uniformly at random, where m satisfies (9) with $r = 11k$ and $\delta = 1/24$;
Search exhaustively through all partitions of W into sets A, B with $|A|, |B| \leq \frac{2}{3}|W|$, minimizing $\mathrm{rank}(\Sigma_{A,B})$;
If no balanced split exists, output any partition A, B of W;
$S \leftarrow$ `ABSeparator`(A, B);
return S

Algorithm 8: `Components(V)`

```
S ← Separator(V);
r ← |S|;
R ← ∅;                                  // will contain one vertex from each C ∈ C^S
for v ∈ V \ S do
    notFound ← True;
    for u ∈ R do
        if rank(Σ_{uS,vS}) = r + 1 then
            C_u ← C_u ∪ {v};
            notFound ← False;
    if notFound then
        create C_v = {v};
        R ← R ∪ {v};
return S and all C_v for v ∈ R;
```

The next proposition shows that Algorithm 8 outputs a balanced separator with high probability.

Proposition 28 *Let $G = (V, E)$ be a graph with $\mathrm{tw}(G) \leq k$, and $|V| \geq 6(k+1)$ vertices and let $\Sigma \in \mathcal{M}(G)$ be generic. Let $\tau \in (0, 1)$. Then, with probability at least $1 - \tau$, Algorithm 7 finds a separator S in G such that $|S| \leq k + 1$ and $|C| \leq 0.93|V|$ for each connected component $C \in \mathcal{C}^S$.*

Proof The procedure starts by choosing a sample $W \subset V$. The size of the sample m is chosen so that, with probability at least $1-\tau$, W is a δ-sample for $\mathcal{F}_{k+1}$. A sufficient condition for m follows by Theorem 22 and Lemma 23. Note that this condition also assures that $|W| \geq 6(k+1)$.

Since $\delta = 1/24$, by Proposition 25, we can partition W into two sets A, B such that $|A|, |B| \leq \frac{2|W|}{3}$ and any minimal separator S of A and B has at most $k+1$ elements. Moreover, for any such partition $|A \setminus S|, |B \setminus S| \leq \frac{4|W \setminus S|}{5}$. Now we only need to show that for each connected component $C \in \mathcal{C}^S$ of G, $|C| \leq \frac{93}{100}|V|$. Indeed, if C^* is the maximal component in $\mathcal{C}^S$ then $\frac{|W \cap C^*|}{|W \setminus S|} \leq \frac{4}{5}$. Since C^* lies in $\mathcal{F}_{k+1}$, we get

$$\frac{|C^* \cap W|}{|W \setminus S|} \geq \left(\frac{|C^*|}{|V \setminus S|} \frac{|V \setminus S|}{|V|} - \delta \right) \frac{|W|}{|W \setminus S|} \overset{(11)}{\geq} \frac{c(S)\frac{|V \setminus S|}{|V|} - \delta}{\frac{|V \setminus S|}{|V|} + 2\delta}.$$

The expression on the right-hand side is an increasing function of $\frac{|V \setminus S|}{|V|}$ and $\frac{|V \setminus S|}{|V|} \geq \lambda$, which gives that

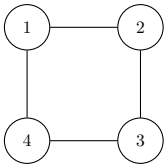
$$\frac{|C^* \cap W|}{|W \setminus S|} \geq \frac{\lambda \frac{|C^*|}{|V \setminus S|} - \delta}{\lambda + 2\delta}$$

Since $\frac{|C^* \cap W|}{|W \setminus S|} \leq \frac{4}{5}$, $\delta \leq \frac{1}{24}$, and $\lambda \geq \frac{5}{6}$, we get that $\frac{|C^*|}{|V \setminus S|} \leq \frac{93}{100}$. ∎

5. Recovery of bounded treewidth graphs

In this section we present an algorithm for reconstructing graphs with bounded treewidth. Let $G = ([n], E)$ be a graph and let $\Sigma \in \mathcal{M}(G)$ be generic. To recover G from Σ we follow a similar divide-and-conquer strategy as in the previous sections. First, a balanced separator S is chosen and then the algorithm recurses into all the components in $\mathcal{C}^S$. There is however a complication. If $C \in \mathcal{C}^S$, then $G[C \cup S]$ is not equal to the graph of Σ_{CS}, unless for *each* $C \in \mathcal{C}^S$ the set of vertices in S that are linked to C by an edge is a clique of G; see (Frydenberg, 1990, Theorem 3.3). The condition holds, in particular, when S is a clique, but in our case there is no way to assure that in general. If it does not hold, then the graph of $\mathcal{G}(\Sigma_{CS})$ is strictly bigger than the subgraph $G[C \cup S]$. The next example illustrates this phenomenon.

Example 1 *Consider the four-cycle given below together with the corresponding covariance and precision matrices*



$$\Sigma = \begin{bmatrix} 7 & -2 & 1 & -2 \\ -2 & 7 & -2 & 1 \\ 1 & -2 & 7 & -2 \\ -2 & 1 & -2 & 7 \end{bmatrix} \qquad K = \frac{1}{24} \begin{bmatrix} 4 & 1 & 0 & 1 \\ 1 & 4 & 1 & 0 \\ 0 & 1 & 4 & 1 \\ 1 & 0 & 1 & 4 \end{bmatrix}$$

Then $\{1,3\}$ *separates* 2 *and* 4 *but the graph* $\mathcal{G}(\Sigma_{123})$ *is the complete graph over* $\{1,2,3\}$ *because*

$$\begin{bmatrix} 7 & -2 & 1 \\ -2 & 7 & -2 \\ 1 & -2 & 7 \end{bmatrix}^{-1} = \frac{1}{96}\begin{bmatrix} 15 & 4 & -1 \\ 4 & 16 & 4 \\ -1 & 4 & 15 \end{bmatrix}.$$

An easy way around this problem is by noting that Gaussian graphical models are closed under conditioning and this probabilistic statement has a useful algebraic counterpart. The graph of the conditional covariance is obtained from G by removing the vertices in the conditioning set and all the incident edges. This means that the edges in $G[C]$ can be recovered from the conditional covariance matrix

$$\Sigma_{C|S} \;:=\; \Sigma_{C,C} - \Sigma_{C,S}\Sigma_{S,S}^{-1}\Sigma_{S,C}. \tag{12}$$

More concretely, we have the following basic result.

Lemma 29 *If* S *separates* C *from the rest of* $G = \mathcal{G}(\Sigma)$ *then* $K_{C,C} = (\Sigma_{C|S})^{-1}$.

This result can be argued by standard properties of the Gaussian distribution as K_C is the inverse of the conditional covariance matrix $\Sigma_{C|[n]\setminus C}$; see, for example, equation (C.3) in Lauritzen (1996). If S separates C from the remaining vertices then, by conditional independence, this conditional covariance matrix is equal to $\Sigma_{C|S}$. For clarity and completeness, we also include an independent purely algebraic argument.

Proof First note that, if S separates C from the rest $B = V \setminus (C \cup S)$ we have

$$\Sigma_{C,B} \;=\; \Sigma_{C,S}\Sigma_{S,S}^{-1}\Sigma_{S,B}.$$

Indeed, by Lemma 3, $\operatorname{rank}(\Sigma_{CS,SB}) = |S|$. Using the Guttman rank additivity formula given below in (14) we conclude that the matrix $\Sigma_{C,B|S} := \Sigma_{C,B} - \Sigma_{C,S}\Sigma_{S,S}^{-1}\Sigma_{S,B}$ is zero. Moreover, the matrix equation

$$\begin{bmatrix} \Sigma_{C,C} & \Sigma_{C,S} & \Sigma_{C,S}\Sigma_{S,S}^{-1}\Sigma_{S,B} \\ \Sigma_{C,C} & \Sigma_{C,S} & \Sigma_{C,B} \\ \Sigma_{B,S}\Sigma_{S,S}^{-1}\Sigma_{S,C} & \Sigma_{C,S} & \Sigma_{C,B} \end{bmatrix} \cdot \begin{bmatrix} \mathbf{A} & \mathbf{B} & \mathbf{0} \\ \mathbf{B}^T & \mathbf{C} & \mathbf{D} \\ \mathbf{0} & \mathbf{D}^T & \mathbf{E} \end{bmatrix} \;=\; I_n \tag{13}$$

is satisfied for

$$\mathbf{A} = (\Sigma_{C|S})^{-1}, \quad \mathbf{B} = -(\Sigma_{C|S})^{-1}\Sigma_{C,S}\Sigma_{S,S}^{-1},$$
$$\mathbf{E} = (\Sigma_{B|S})^{-1}, \quad \mathbf{D} = -(\Sigma_{B|S})^{-1}\Sigma_{B,S}\Sigma_{S,S}^{-1},$$

and

$$\mathbf{C} = \Sigma_{S,S}^{-1}\left(\Sigma_{S,S} + \Sigma_{S,C}(\Sigma_{C|S})^{-1}\Sigma_{C,S} + \Sigma_{S,B}(\Sigma_{B|S})^{-1}\Sigma_{B,S}\right)\Sigma_{S,S}^{-1}.$$

The first term in the product in (13) is Σ. Thus the second term is $K = \Sigma^{-1}$. In particular, $K_{C,C} = \mathbf{A} = (\Sigma_{C|S})^{-1}$. ■

This result shows that in order to keep information about the induced subgraph $G[C]$ once we regress on C, it is important to keep the information about the separating set.

To see how this is done, it is helpful to study the situation in Figure 5. Suppose that S separates G into several components one of which is C. We then recurse our algorithm on C by conditioning on S. In the next step we use the matrix $\Sigma_{C|S}$ to find a balanced separator S' of $G[C]$. We then recurse on the corresponding components C_1, C_2, C_3, C_4. Note that in the next step it is not enough to condition on S' to study $G[C_2]$ because it is connected to the rest of the graph through S. Therefore, in this recursive call we need to work with the conditional covariance matrix $\Sigma_{C_2|SS'}$.

The dependence on separating sets requires a modification of the algorithms that we use to decompose the graph. Instead of working on the covariance matrix, they should be working on the conditional covariance matrix. Note however, that rank queries for $\Sigma_{A,B|S}$ with $A, B \subset C$ are equivalent to rank queries on $\Sigma_{AS,BS}$. Indeed, by the Guttman rank additivity formula (see e.g. (Zhang, 2005, Section 0.9))

$$\operatorname{rank}(\Sigma_{AS,BS}) \;=\; \operatorname{rank}(\Sigma_S) + \operatorname{rank}(\Sigma_{A,B|S}) \;=\; |S| + \operatorname{rank}(\Sigma_{A,B|S}). \tag{14}$$

Therefore the algorithms $\texttt{ABSeparator}(A, B)$, $\texttt{Separator}$, $\texttt{Components}$ have their simple modifications $\texttt{ABSeparator}(V, A, B, S)$, $\texttt{Separator}(V, S)$, $\texttt{Components}(V, S)$, where a set S disjoint from V is added to both the row and the column set in all the rank queries. For completeness we explicitly provide these algorithms in Appendix A.

In Algorithm 9 we present the complete algorithm which relies on routine $\texttt{Reconstruct}$, which is then called recursively in Algorithm 10. With a fixed bound k on the treewidth of G, the main algorithm returns the precision matrix $K = \Sigma^{-1}$.

Algorithm 9: *Main algorithm*

$\widehat{K} \leftarrow 0 \in \mathbb{R}^{n \times n}$
Fix m satisfying (9) with $r = 11k$ and $\delta = \frac{1}{24}$
$\texttt{Reconstruct}([\texttt{n}], \emptyset)$

At each call of $\texttt{Reconstruct}(V, S)$, if the input vertex set V is larger than the fixed threshold m, then $\texttt{Separator}(V)$ finds a balanced separator S' of $G[V]$. Then the procedure $\texttt{Components}$ finds all connected components C_i in $G[V] \setminus S'$. Subsequently, $\texttt{Reconstruct}$ recurses in all these components replacing S with $S \cup S'$ as in Figure 5.

Most edges in K are only reconstructed in the final recursive calls. Consider the situation in Figure 5. Suppose that $\texttt{Reconstruct}(C, S)$ recurses to $\texttt{Reconstruct}(C_1, S \cup S')$. If $|C_1| \leq m$ then $\texttt{Reconstruct}(C_1, S \cup S')$ computes K_{C_1}, which by Lemma 29 is equal to the inverse of $\Sigma_{C_1|SS'}$. The matrices $K_{C_1S'}$ and $K_{S'S'}$ can be computed in a similar way as described in the lemma below.

Lemma 30 *Suppose $C \in \mathcal{C}^S$ and that C is further decomposed into S' and the connected components $\{C_1, \ldots, C_d\}$ (as in Figure 5). Let $K = \Sigma^{-1}$. The submatrix K_C has a block structure with $K_{C_i,C_j} = 0$ for $i \neq j$ and*

$$K_{C_i,S'} = -K_{C_i}\Sigma_{C_i,S'|S}\Sigma^{-1}_{S'|S}, \qquad K_{S'} = \left(\mathbb{I}_{|S'|} - \sum_{i=1}^{d} K_{S',C_i}\Sigma_{C_i,S'|S}\right)\Sigma^{-1}_{S'|S},$$

where $\mathbb{I}_m$ denotes the $m \times m$ identity matrix.

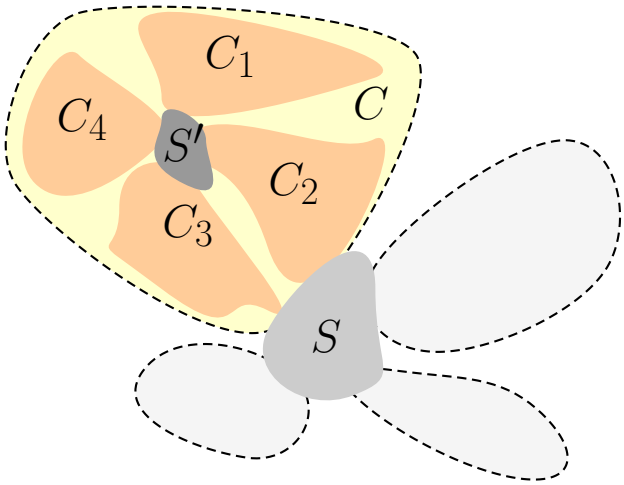


Figure 5: Components found during subsequent recursive calls to `Reconstruct`. The algorithm needs to keep track of the separators found at all levels, as C_2 and C_3 are connected by both S' and S.

Proof There are no direct links between C_i and C_j in G and so $K_{C_i,C_j} = 0$ for $i \neq j$. Lemma 29 gives the identity $K_C \Sigma_{C|S} = \mathbb{I}_{|C|}$. Taking the C_i-rows of K_C and the S'-columns of $\Sigma_{C|S}$ we get from this identity that

$$K_{C_i} \Sigma_{C_i,S'|S} + K_{C_i,S'} \Sigma_{S',S'|S} \;=\; 0,$$

which implies the first equality. Taking the S'-rows and S'-columns we get

$$\sum_{i=1}^{d} K_{S',C_i} \Sigma_{C_i,S'|S} + K_{S'} \Sigma_{S'|S} \;=\; \mathbb{I}_{|S'|},$$

which implies the second formula. ■

Algorithm 10: `Reconstruct`(V, S)

if $|V| > m$ **then**
  $C_1, \ldots, C_d, S' \leftarrow$ `Components`(V, S);
  for i from 1 to d **do**
    $\widehat{K}_{C_i} \leftarrow$ `Reconstruct`$(C_i, S \cup S')$;
    $\widehat{K}_{C_i,S'} \leftarrow -\widehat{K}_{C_i} \Sigma_{C_i,S'|S} \Sigma^{-1}_{S'|S}$
  $\widehat{K}_{S'} \leftarrow \left(\mathbb{I}_{|S'|} - \sum_{i=1}^{d} \widehat{K}_{S',C_i} \Sigma_{C_i,S'|S} \right) \Sigma^{-1}_{S'|S}$;
else
  return $\widehat{K}_V$

Theorem 31 `Reconstruct`$([n], \emptyset)$ *correctly recovers* $K = \Sigma^{-1}$.

Proof The correctness of `Components`$([n], \emptyset)$ was already shown in Proposition 28. Using the discussion given in the beginning of this section, we can easily adjust this proof for any call of `Components`(V, S). This, together with Lemma 29, implies that in each call of `Components`(V, S):

(i) If u, v are put in different components components then $K_{uv} = 0$.

(ii) It holds that $K_{C_i} = (\Sigma_{C_i|SS'})^{-1}$.

By (i) all (zero) entries in K_{C_i,C_j} are correctly recovered. By (ii) all the entries of K_{C_i} obtained by inverting $\Sigma_{C_i|SS'}$ are correct. By Lemma 30, also $K_{C_i,S'}$ and $K_{S',S'}$ are correctly recovered. ■

Theorem 32 *Let $G = ([n], E)$ be a graph with treewidth $\mathrm{tw}(G) \leq k$ and maximum degree $\Delta(G) \leq d$. Let $\Sigma \in \mathcal{M}(G)$ be generic and let m in* `Separator` *satisfy (9) with $r \geq 11k$, $\delta \leq \frac{1}{24}$, and $\tau \leq \frac{1}{3}$. Then, with probability at least $1 - \frac{1}{n^8}$, the query complexity of* `Reconstruct` *is of the order*

$$\mathcal{O}\left((2^{\mathcal{O}(k \log k)} + dk \log n)k^2 n \log^3 n\right),$$

and the time complexity is of the order

$$\mathcal{O}\left((2^{\mathcal{O}(k \log k)} + dk \log n)k^3 n \log^4 n\right).$$

Before we prove this result, a number of remarks are in order.

Remark 33 *The choice of $\tau \leq 1/3$ in Theorem 32 is arbitrary; any other choice would only change the constant factors in the complexity bounds and the probability of exceeding the stated query and time complexity. Moreover, we note that the total probability of error can be made arbitrarily small. In the following proof we show that the recursion depth is $\mathcal{O}(\log n)$ with high probability. Then, executing the algorithm $\mathcal{O}(\log \frac{1}{\epsilon})$ times (each time stopping if it does not finish in the time limit stated in Theorem 32) we get at least one timely execution of the algorithm with probability $1-\epsilon$, regardless of n. The complexity only changes by a factor of $\mathcal{O}(\log \frac{1}{\epsilon})$.*

Remark 34 *As it is seen from the proof, the assumption of bounded degree can be relaxed. It can be substituted by the assumption that removing $\mathcal{O}(k \log n)$ vertices decomposes the graph into at most a polylogarithmic number of connected components. In other words, one may weaken the bounded-degree assumption by suitable assumptions on the* fragmentation *of the graph. We refer to Hajiaghayi and Hajiaghayi (2003) and Hajiaghayi and Nishimura (2007) for more information on this notion. An example of a graph with unbounded degree for which our reconstruction method works is the wheel graph (i.e., the graph formed by connecting a central vertex to all vertices of a cycle of $n-1$ vertices). This graph has treewidth 3, maximum degree $n-1$ but low fragmentation.*

Remark 35 *The problem of computing the treewidth of a graph given its adjacency matrix is NP-hard Arnborg et al. (1987) (see Bodlaender et al. (2016) for an account on the history of the problem and references on current results). Hence it is not surprising to have an exponential dependence on treewidth. Note however that in our case we do not have access to edge queries but to separation queries, in contrast to the traditional setting. We are not aware of algorithmic results in this setting. We expect that similar hardness results should hold but proving such hardness seems non-trivial and outside of the scope of this paper.*

Proof [Proof of Theorem 32] We refer to all operations in $\texttt{Reconstruct}([n], \emptyset)$ excluding operations in subsequent calls $\texttt{Reconstruct}(C_i, S')$ as the zeroth recursion level of the algorithm. Similarly, the operations of all $\texttt{Reconstruct}(C_i, S')$ for $i = 1, \dots, d$ apart from their subsequent calls are called the first recursion level. We extend this definition iteratively to the t-th recursion level for $t > 1$.

Assume initially that `Components` never fails, that is, $|S'| \leq k+1$ and for each connected component $C \in \mathcal{C}^S$ it holds that $|C| \leq 0.93|V|$, as stated in Proposition 28. We will bound the total probability of failure later in the proof. Since the algorithm recurses on sets C_i of size at most $0.93|V|$, the recursion depth (maximal number of recursion levels) is at most $\log_{100/93} n$, which is of order $\mathcal{O}(\log(n))$. Moreover, in each call of $\texttt{Reconstruct}(V, S)$ always $|S| = \mathcal{O}(k \log n)$.

We start with the analysis of $\texttt{Components}(V, S)$. Assume first that $|V| > m$ and write $s = |S|$ and $s' = |S'|$. Finding a balanced partition A, B of W in `Separator` is achieved by exhaustively searching all $< 2^m$ balanced partitions and computing the rank of the associated matrices $\Sigma_{ASS',BSS'}$, which gives query complexity $\mathcal{O}(2^m(m+s)^2)$ and time complexity $\mathcal{O}(2^m(m+s)^3)$ for this step. Taking into account that $m = \mathcal{O}(k \log k)$ and $s = \mathcal{O}(k \log n)$, we obtain complexity

$$\mathcal{O}(2^m s^2) \text{ and } \mathcal{O}(2^m s^3)$$

for queries and time, respectively. Then, given a balanced split, $\texttt{ABSeparator}(V, A, B, S)$ finds a separator of A and B in $\mathcal{O}(|V|(m+s)^2)$ queries and $\mathcal{O}(|V|(m+s)^3)$ time. This bound can be obtained by a simple modification of Proposition 20, which gives bounds for $\texttt{ABSeparator}(A, B)$. Hence we obtain complexity

$$\mathcal{O}(|V|s^2) \quad \text{and} \quad \mathcal{O}(|V|s^3)$$

for queries and time, respectively.

Since removing at most s' vertices of degree at most d splits the graph into at most ds' connected components, splitting V into the connected components C_i requires $\mathcal{O}(|V|s'ds)$ queries and $\mathcal{O}(|V|s'ds^3)$ time, and since $s = \mathcal{O}(k \log n)$, this gives query and time complexity bounds

$$\mathcal{O}(|V|dk^2 \log n) \text{ and } \mathcal{O}(|V|dk^4 \log^3 n).$$

In the case where $|V| \leq m$, we obtain queries and time of the order $\mathcal{O}(s^2)$ and $\mathcal{O}(s^3)$ respectively. These terms are dominated by the terms that appeared for earlier steps of the algorithm and will be ignored in what follows. Overall, *at each recursion level* this part of the algorithm requires

$$\mathcal{O}\left(n2^m k^2 \log^2 n + nk^2 \log^2 n + ndk^2 \log n\right)$$

queries and

$$\mathcal{O}(n2^m k^3 \log^3 n + nk^3 \log^3 n + ndk^4 \log^3 n)$$

time, since the vertex sets V are disjoint and there can be up to $O(n)$ calls to `Separator` at the bottom levels of recursion. Using the fact that $m = \mathcal{O}(k \log k)$ and simplifying, we obtain:

$$\mathcal{O}\left(2^{\mathcal{O}(k \log k)} k^2 n \log^2 n + dk^2 n \log^2 n\right) \tag{15}$$

queries and

$$\mathcal{O}(2^{\mathcal{O}(k \log k)} k^3 n \log^3 n + dk^4 n \log^3 n) \tag{16}$$

time.

After calling `Components`(V, S), `Reconstruct` obtains $\widehat{K}_{C_i}$ (we focus on a fixed recursion level so we can ignore the recursive call of `Reconstruct`(V, S)) and computes the matrices $\widehat{K}_{C_i,S'}$. After these matrices are computed for all components C_i, the algorithm computes $\widehat{K}_{S',S'}$. For these computations we need to calculate the conditional covariance matrices $\Sigma_{C_i,S'|S}$ and $\Sigma_{S'|S}$. The time to compute each $\Sigma_{C_i,S'|S} = \Sigma_{C_i,S'} - \Sigma_{C_i,S}\Sigma_S^{-1}\Sigma_{S,S'}$ requires $\mathcal{O}(|C_i|s + s^2)$ and $\mathcal{O}(|C_i|s^2 + s^3)$ queries and time, respectively, hence

$$\mathcal{O}(|V|s + s^2 dk \log n) \text{ and } \mathcal{O}(|V|s^2 + s^3 dk \log n),$$

for all C_i. Computing $\widehat{K}_{C_i,S'} = \widehat{K}_{C_i}\Sigma_{C_i,S'|S}\Sigma_{S'}^{-1}$ requires only $\mathcal{O}(k^2)$ additional queries (we use $s' \leq k+1$). The time complexity of this computation is dominated by the time needed to compute $\widehat{K}_{C_i}\Sigma_{C_i,S'|S}$. A naive method of computing $\widehat{K}_{C_i}\Sigma_{C_i,S'|S}$ would require $\mathcal{O}(|C_i|^2 k)$ time, which is too time-consuming for our purposes. However, we can take advantage of the fact that the subgraph $G[C_i]$ has treewidth at most k (by the remark following Proposition 17) and the number of edges in such a graph is at most $|C_i|k$. This implies that there are at most $k|C_i|$ non-zero entries in $\widehat{K}_{C_i}$ and so the multiplication takes time $\mathcal{O}(k^2|C_i|)$. Considering all C_i, we obtain time complexity

$$\mathcal{O}(k^2|V|)$$

for this step. Finally, to compute $\widehat{K}_{S'}$ we need additional $\mathcal{O}(sk)$ queries for $\Sigma_{S,S'}$ and $\mathcal{O}(|V|k^2 + dk^3)$ time to compute $\widehat{K}_{S'}$. However, since $s = \mathcal{O}(k \log n)$, these terms are clearly dominated by complexity of the preceding steps in the algorithm and so they will be ignored in what follows. Overall, at each recursion level this part of the algorithm requires

$$\mathcal{O}\left(ns + ns^2 dk \log n\right) \qquad \text{and} \qquad \mathcal{O}(ns^2 + ns^3 dk \log n + k^2 n)$$

queries and time respectively. Using the fact that $m = \mathcal{O}(k \log k)$ and $s = \mathcal{O}(k \log n)$ and simplifying, we obtain:

$$\mathcal{O}\left(kn \log n + dk^3 n \log^3 n\right) \tag{17}$$

queries and

$$\mathcal{O}(k^2 n \log^2 n + dk^4 n \log^4 n) \tag{18}$$

time.

The total complexity for a fixed recursive level are obtained by combining (15)–(18). Taking into account the recursion depth $\mathcal{O}(\log(n))$ we get the stated overall complexity bounds.

Let I_i be the indicator variable that the i-th call of `Components` succeeds. This happens with $\mathbb{P}(I_i = 1) = 1 - \tau \geq 2/3$. Let $\alpha = 0.93$. Consider a given recursion path from the root to a leaf in the recursion tree. There are at most $\log_{1/\alpha} n$ calls with $I_i = 1$ in such a recursion path. Since the I_i are independent, we can use Hoeffding's inequality to bound the probability that we have less than $\log_{1/\alpha} n$ successes in $N = 3 \log_{1/\alpha} n$ calls:

$$\mathbb{P}\left(\sum_{i=1}^{N} I_i < \frac{1}{3}N\right) \;\leq\; e^{-\frac{2}{9}N} \;=\; n^{-\frac{2}{3\log(1/\alpha)}} \;\leq\; n^{-9}.$$

This argument implies that a fixed path from the root to a leaf in the recursion tree is logarithmic with high probability. There are at most n such paths. Hence, by the union bound and the condition for n, the probability that there exists one of them with more than logarithmic length is bounded by $1/n^8$. ■

6. Tree reconstruction using noisy covariance oracles

In this paper we focus on the noiseless setting when entries of a matrix $\Sigma \in \mathcal{M}(G)$ may be accessed by a learner and these values are available exactly, without error – this is our "covariance oracle." In other words, we have shown that in many cases it is possible to invert the positive definite matrix Σ after seeing a tiny, adaptively chosen, fraction of its entries.

In many learning problems, the entries of Σ are not available exactly. This is the case in statistical problems when Σ is an unknown covariance matrix of a random vector X and its entries may be estimated from data. In this section we discuss how the results of this paper may be extended to situations when covariances may not be observed exactly, for example, due to statistical fluctuations.

Here we limit ourselves to the study of the case when the underlying graph G is a tree. We show how Algorithm 2 may be modified to handle noise and establish sufficient conditions that guarantee correct recovery. Along similar lines, one may also modify the other algorithms introduced in this paper (for recovery of tree-like graphs and graphs of bounded treewidth). However, the details are somewhat tedious and go beyond the scope of this paper.

In order to simplify the presentation, we assume that all diagonal elements of Σ are equal to 1, that is, Σ is a correlation matrix with entries σ_{ij} for $i \neq j$. The extension of the general case is straightforward, at the price of visually more complicated formulas.

In the discussion that follows, we assume that the noisy covariance oracle, when queried for the (i, j)-th entry of Σ, returns a value $\widehat{\sigma}_{ij} \in [-1, 1]$ satisfying

$$\max_{ij} |\widehat{\sigma}_{ij} - \sigma_{ij}| < \epsilon \tag{19}$$

for some $\epsilon \in (0, 1)$. We assume that $\widehat{\sigma}_{ii} = 1$ for all i. In a statistical setting when Σ is the covariance matrix of a random vector X and one may obtain independent samples of X, it is easy to construct such a noisy covariance oracle. We discuss this in more detail at the end of the section.

In the noise-free case we required that the graph $\mathcal{G}(\Sigma)$ is connected and generic, or, equivalently, Σ does not have any entry in $\{-1, 0, 1\}$. In the presence of noise, because of problems of identifiability, we need a stronger assumptions on the entries of Σ corresponding to edges of the graph. In particular, we assume that there exist constants $0 < \delta < \gamma < 1$ such that

$$\delta \leq |\sigma_{ij}| \leq \gamma \qquad \text{for all } ij \in E \ . \tag{20}$$

By the product formula (2), this implies that for any two distinct $i, j \in V$,

$$|\sigma_{ij}| \geq \delta^D \ ,$$

where D is the diameter of the graph. Although the diameter of G played no role in the noise-free setting, in the noisy case the recovery guarantees crucially depend on D. In particular, our recovery guarantees are only meaningful when D is logarithmic in $1/\epsilon$. To see why this happens, note that under assumption (20), $|\sigma_{ij}| \leq \gamma^{d(i,j)}$ where $d(i, j)$ denotes the distance of vertex i and vertex j in the tree. This value becomes indistinguishable from zero by the noisy covariance oracle unless $d(i, j) < \log(1/\epsilon)/\log(1/\gamma)$.

Remark 36 *The assumption that the diameter of G is small is relatively mild. Many important real life examples of complex networks have small diameter – these are the so-called small-world networks. For example the diameter of the world wide web, with way over billion nodes Van den Bosch et al. (2016), is around* 19 *Albert et al. (1999), while social networks with over six billion individuals are believed to have a diameter of around six Milgram (1967). Small-world networks have also found applications in brain study Bassett and Bullmore (2006).*

Next we show how Algorithm 2 may be modified so that it tolerates noise of magnitude ϵ – in the sense of (19).

Algorithm 2 uses the covariance oracle in order to check whether $\det(\Sigma_{ij,jk}) = 0$ for some triples of vertices i, j, k, or equivalently, whether $\sigma_{ij}\sigma_{jk} = \sigma_{ik}$. Also, the algorithm sorts all the correlations σ_{uw} for a fixed w. We show that both of these steps can be correctly executed with a noisy covariance oracle if

$$\epsilon \ \leq \ \frac{1}{8}\delta^D(1 - \gamma^2) \ . \tag{21}$$

Under (21), we may choose a value τ such that

$$\tau > 3\epsilon \quad \text{and} \quad \tau \leq \delta^D\left(1 - \gamma^2\right) - 3\epsilon \ .$$

In order to test whether $\sigma_{ij}\sigma_{jk} = \sigma_{ik}$, we use the decision

$$\begin{array}{lll} \text{if} & |\widehat{\sigma}_{ij}\widehat{\sigma}_{jk} - \widehat{\sigma}_{ik}| < \tau & \text{accept} \\ \text{if} & |\widehat{\sigma}_{ij}\widehat{\sigma}_{jk} - \widehat{\sigma}_{ik}| > \tau & \text{reject} \ . \end{array}$$

To see why the decision is correct, observe first that for all $i, j \in [n]$,

$$|\sigma_{ij}\sigma_{jk} - \widehat{\sigma}_{ij}\widehat{\sigma}_{jk}| = |\sigma_{jk}\left(\sigma_{ij} - \widehat{\sigma}_{ij}\right) + \widehat{\sigma}_{ij}\left(\sigma_{jk} - \widehat{\sigma}_{jk}\right)| \leq 2\epsilon \ . \tag{22}$$

Hence, if $\sigma_{ij}\sigma_{jk} = \sigma_{ik}$, then

$$|\widehat{\sigma}_{ij}\widehat{\sigma}_{jk} - \widehat{\sigma}_{ik}| \leq 3\epsilon < \tau$$

and the decision is correct.

We now treat the case where $\sigma_{ij}\sigma_{jk} \neq \sigma_{ik}$. We consider two cases.
Case I: There exists a vertex m that separates all i, j, k, that is,

$$\sigma_{im}\sigma_{mk} = \sigma_{ik} \quad \text{and} \quad \sigma_{jm}\sigma_{mk} = \sigma_{jk} \ .$$

In this case

$$\begin{aligned} |\sigma_{ij}\sigma_{jk} - \sigma_{ik}| &= \left|\sigma_{im}\sigma_{jm}^2\sigma_{km} - \sigma_{im}\sigma_{km}\right| \\ &= |\sigma_{ik}|\left|\sigma_{jm}^2 - 1\right| \\ &\geq \delta^D\left(1-\gamma^2\right) \ . \end{aligned} \tag{23}$$

But then, arguing as above,

$$\begin{aligned} |\widehat{\sigma}_{ij}\widehat{\sigma}_{jk} - \widehat{\sigma}_{ik}| &\geq |\sigma_{ij}\sigma_{jk} - \sigma_{ik}| - 3\epsilon \\ &\geq \delta^D\left(1-\gamma^2\right) - 3\epsilon > \tau \end{aligned}$$

and the decision is once again correct. It remains to consider
Case II: Either i separates j and k or k separates i and j. Without loss of generality, assume the latter, so that $\sigma_{ij} = \sigma_{ik}\sigma_{jk}$. Then

$$\begin{aligned} |\sigma_{ij}\sigma_{jk} - \sigma_{ik}| &= \left|\sigma_{ik}\sigma_{jk}^2 - \sigma_{ik}\right| \\ &= |\sigma_{ik}|\left|\sigma_{jk}^2 - 1\right| \\ &\geq \delta^D\left(1-\gamma^2\right) \ . \end{aligned}$$

Hence, similarly to Case I, we have

$$\begin{aligned} |\widehat{\sigma}_{ij}\widehat{\sigma}_{jk} - \widehat{\sigma}_{ik}| &\geq |\sigma_{ij}\sigma_{jk} - \sigma_{ik}| - 3\epsilon \\ &\geq \delta^D\left(1-\gamma^2\right) - 3\epsilon > \tau \ , \end{aligned}$$

which proves correctness of the testing procedure.

The only other ingredient of Algorithm 2 that uses the covariance oracle performs queries of the form $|\sigma_{uw}| < |\sigma_{vw}|$. In the presence of a noisy covariance oracle satisfying (19), we may use the following rule:

$$\begin{aligned} \text{if } |\widehat{\sigma}_{uw}| < |\widehat{\sigma}_{vw}| - 2\epsilon &\quad \text{accept} \\ \text{if } |\widehat{\sigma}_{vw}| < |\widehat{\sigma}_{uw}| - 2\epsilon &\quad \text{reject} \\ \text{otherwise} &\quad \text{do either} \ . \end{aligned}$$

We now show that the decision rule accepts and rejects correctly. In the first two cases, the decision is clearly correct. Hence, we only need to examine the case when

$$|\widehat{\sigma}_{uw}| \geq |\widehat{\sigma}_{vw}| - 2\epsilon \quad \text{and} \quad |\widehat{\sigma}_{vw}| \geq |\widehat{\sigma}_{uw}| - 2\epsilon$$

happen simultaneously. In such case,

$$||\widehat{\sigma}_{uw}| - |\widehat{\sigma}_{vw}|| \leq 2\epsilon \ .$$

which implies

$$||\sigma_{uw}| - |\sigma_{vw}|| \leq 4\epsilon \ .$$

In terms of the ordering of correlations, it is enough that for a central vertex w in Algorithm 3 the following holds: if

(i) w does not separate u and v and

(ii) v is the neighbor of w

then $|\widehat{\sigma}_{vw}| > |\widehat{\sigma}_{uw}|$. Indeed, note that the only thing that matters is that, at each step of the algorithm, all vertices in the same connected component of $G \setminus w$ are sorted after the unique neighbour of w that belongs in that component. But if (i) and (ii) hold then

$$||\sigma_{uw}| - |\sigma_{vw}|| \ = \ |\sigma_{vw}|(1 - |\sigma_{uv}|) \geq \delta(1-\gamma) > 4\epsilon \ ,$$

where the last inequality follows by (21).

This concludes the proof of correctness of the modified procedure under condition (21).

We close this section by noting that a noisy covariance oracle satisfying (19) may easily be constructed when Σ is the covariance matrix of a zero-mean random vector $X = (X_1, \ldots, X_n)$. All one needs is that, for each pair of indices $i, j \in [n]$ queried by the algorithm, one may obtain N i.i.d. samples of the pair (X_i, X_j). For example, if all components of X have a bounded fourth moment, say $\mathbb{E}X_i^4 \leq \kappa$ for all $i \in [n]$ for some $\kappa > 0$, then $\mathrm{Var}(X_i X_j) \leq \kappa$, and therefore one may use robust mean estimators (see, e.g., (Lugosi and Mendelson, 2019, Theorem 2)) to estimate $\sigma_{ij} = \mathbb{E}[X_i X_j]$ to obtain $\widehat{\sigma}_{ij}$ that satisfy (21) (simultaneously, for all $i, j \in [n]$), with probability at least $1 - \eta$, whenever

$$N \geq 32 \left(\frac{\kappa}{\epsilon}\right)^2 \log \frac{n}{\eta} \ .$$

Acknowledgments

We would like to thank Robert Castelo, Vida Dujmović, and David Rossell for helpful discussions. GL, VV, and PZ were supported by the Spanish Ministry of Economy and Competitiveness, Grant PGC2018-101643-B-I00 and FEDER, EU. GL and PZ acknowledge the support of "High-dimensional problems in structured probabilistic models - Ayudas Fundación BBVA a Equipos de Investigación Cientifica 2017". GL was supported by "Google Focused Award Algorithms and Learning for AI" and PZ by Beatriu de Pinós grant (BP-2016-00002) and Ramón y Cajal (RYC-2017-22544).

Appendix A. Modified algorithms

Here we present the modified versions of Algorithms 6, 7, and 8 that use conditional covariance information, as described in Section 5.

Algorithm 11: `ABSeparator`(V, A, B, S)

$U \leftarrow \emptyset$;
$r = \operatorname{rank}(\Sigma_{AS,BS})$;
forall $v \in V$ **do**
  if $\operatorname{rank}(\Sigma_{ASv,BSv}) = r$ **then**
    $U \leftarrow U \cup \{v\}$;
$C \leftarrow \{v_0\}$ for some $v_0 \in U$;
forall $u \in U \setminus \{v_0\}$ **do**
  if $\operatorname{rank}(\Sigma_{ASCu,BSCu}) = r$ **then**
    $C \leftarrow C \cup \{u\}$;
return C;

Algorithm 12: `Separator`(V, S)

Pick a set $W \subset V$ by taking m vertices uniformly at random, where m satisfies (9) with $r = 11k$ and $\delta = 1/24$;
Search exhaustively through all partitions of W into sets A, B with $|A|, |B| \leq \frac{2}{3}|W|$, minimizing $\operatorname{rank}(\Sigma_{AS,BS})$;
If no balanced split exists, output any partition A, B of W;
$S' \leftarrow$ `ABSeparator`(V, A, B, S);
return S'

Algorithm 13: `Components`(V, S)

$S' \leftarrow$ `Separator`(V);
$r \leftarrow |S'| + |S|$;
$R \leftarrow \emptyset$; `// will contain one vertex from each` $C \in \mathcal{C}^{S'}$
for $v \in V \setminus S'$ **do**
  $notFound \leftarrow True$;
  for $u \in R$ **do**
    if $\operatorname{rank}(\Sigma_{uSS',vSS'}) = r + 1$ **then**
      $C_u \leftarrow C_u \cup \{v\}$;
      $notFound \leftarrow False$;
  if $notFound$ **then**
    create $C_v = \{v\}$;
    $R \leftarrow R \cup \{v\}$;
return S' *and all* C_j;

References

Muad Abu-Ata and Feodor F Dragan. Metric tree-like structures in real-world networks: an empirical study. *Networks*, 67(1):49–68, 2016.

Aaron B Adcock, Blair D Sullivan, and Michael W Mahoney. Tree-like structure in large social and information networks. In *2013 IEEE 13th International Conference on Data Mining*, pages 1–10. IEEE, 2013.

Réka Albert, Hawoong Jeong, and Albert-László Barabási. Internet: Diameter of the world-wide web. *nature*, 401(6749):130, 1999.

Stefan Arnborg, Derek G Corneil, and Andrzej Proskurowski. Complexity of finding embeddings in ak-tree. *SIAM Journal on Algebraic Discrete Methods*, 8(2):277–284, 1987.

Onureena Banerjee, Laurent El Ghaoui, and Alexandre d'Aspremont. Model selection through sparse maximum likelihood estimation for multivariate gaussian or binary data. *Journal of Machine Learning Research*, 9(Mar):485–516, 2008.

Danielle Smith Bassett and ED Bullmore. Small-world brain networks. *The neuroscientist*, 12(6):512–523, 2006.

Kevin Bello and Jean Honorio. Computationally and statistically efficient learning of causal bayes nets using path queries. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL `https://proceedings.neurips.cc/paper/2018/file/a0b45d1bb84fe1bedbb8449764c4d5d5-Paper.pdf`.

Hans L Bodlaender. A partial k-arboretum of graphs with bounded treewidth. *Theoretical Computer Science*, 209(1-2):1–45, 1998.

Hans L Bodlaender, Pål Grønås Drange, Markus S Dregi, Fedor V Fomin, Daniel Lokshtanov, and Michał Pilipczuk. A $c^k n$ 5-approximation algorithm for treewidth. *SIAM Journal on Computing*, 45(2):317–378, 2016.

Daniel G Brown and Jakub Truszkowski. Fast error-tolerant quartet phylogeny algorithms. In *Annual Symposium on Combinatorial Pattern Matching*, pages 147–161. Springer, 2011.

Daniel G Brown and Jakub Truszkowski. Fast phylogenetic tree reconstruction using locality-sensitive hashing. In *International Workshop on Algorithms in Bioinformatics*, pages 14–29. Springer, 2012.

T Tony Cai, Weidong Liu, Harrison H Zhou, et al. Estimating sparse precision matrix: Optimal rates of convergence and adaptive estimation. *The Annals of Statistics*, 44(2): 455–488, 2016.

Thalia E Chan, Michael PH Stumpf, and Ann C Babtie. Network inference and hypotheses-generation from single-cell transcriptomic data using multivariate information measures. *bioRxiv*, 2016. doi: 10.1101/082099. URL `https://www.biorxiv.org/content/early/2016/10/20/082099`.

Venkat Chandrasekaran and Michael I Jordan. Computational and statistical tradeoffs via convex relaxation. *Proceedings of the National Academy of Sciences*, 110(13):E1181–E1190, 2013.

Venkat Chandrasekaran, Nathan Srebro, and Prahladh Harsha. Complexity of inference in graphical models. *arXiv preprint arXiv:1206.3240*, 2012.

C Chow and Cong Liu. Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, 14(3):462–467, 1968.

Gautamd Dasarathy, Aarti Singh, Maria-Florina Balcan, and Jong H Park. Active learning algorithms for graphical model selection. In *Artificial Intelligence and Statistics*, pages 1356–1364, 2016.

L. Devroye and G. Lugosi. *Combinatorial Methods in Density Estimation*. Springer-Verlag, New York, 2000.

Mathias Drton and Marloes H Maathuis. Structure learning in graphical modeling. *Annual Review of Statistics and Its Application*, 4:365–393, 2017.

David Edwards, Gabriel CG De Abreu, and Rodrigo Labouriau. Selecting high-dimensional mixed graphical models using minimal aic or bic forests. *BMC Bioinformatics*, 11(1):18, 2010.

Uriel Feige and Mohammad Mahdian. Finding small balanced separators. In *Proceedings of the Thirty-eighth Annual ACM Symposium on Theory of Computing*, pages 375–384. ACM, 2006.

Morten Frydenberg. Marginalization and collapsibility in graphical interaction models. *The Annals of Statistics*, pages 790–805, 1990.

Mohammad Taghi Hajiaghayi and Mahdi Hajiaghayi. A note on the bounded fragmentation property and its applications in network reliability. *European Journal of Combinatorics*, 24(7):891–896, 2003.

MohammadTaghi Hajiaghayi and Naomi Nishimura. Subgraph isomorphism, log-bounded fragmentation, and graphs of (locally) bounded treewidth. *Journal of Computer and System Sciences*, 73(5):755–768, 2007.

Frank Harary. *Graph theory*. Addison-Wesley, Reading, MA, 1969.

Søren Højsgaard, David Edwards, and Steffen Lauritzen. *Graphical models with R*. Springer Science & Business Media, 2012.

Cho-Jui Hsieh, Mátyás A Sustik, Inderjit S Dhillon, Pradeep K Ravikumar, and Russell Poldrack. Big & quic: Sparse inverse covariance estimation for a million variables. In *Advances in neural information processing systems*, pages 3165–3173, 2013.

Shuai Huang, Jing Li, Liang Sun, Jieping Ye, Adam Fleisher, Teresa Wu, Kewei Chen, Eric Reiman, Alzheimer's Disease NeuroImaging Initiative, et al. Learning brain connectivity of alzheimer's disease by sparse inverse covariance estimation. *NeuroImage*, 50(3):935–949, 2010.

Byungjin Hwang, Ji Hyun Lee, and Duhee Bang. Single-cell rna sequencing technologies and bioinformatics pipelines. *Experimental & Molecular Medicine*, 50(8):96, 2018. doi: 10.1038/s12276-018-0071-8. URL `https://doi.org/10.1038/s12276-018-0071-8`.

M. Jagadish and Anindya Sen. Learning a bounded-degree tree using separator queries. In Sanjay Jain, Rémi Munos, Frank Stephan, and Thomas Zeugmann, editors, *Algorithmic Learning Theory*, pages 188–202, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

David Karger and Nathan Srebro. Learning markov networks: Maximum bounded tree-width graphs. In *Proceedings of the Twelfth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 392–401. Society for Industrial and Applied Mathematics, 2001.

Johan Kwisthout, Hans L Bodlaender, and Linda C van der Gaag. The necessity of bounded treewidth for efficient inference in bayesian networks. In *ECAI*, volume 215, pages 237–242, 2010.

Steffen L Lauritzen. *Graphical models*. Clarendon Press, 1996.

Han Liu, John Lafferty, and Larry Wasserman. The nonparanormal: Semiparametric estimation of high dimensional undirected graphs. *Journal of Machine Learning Research*, 10(Oct):2295–2328, 2009.

Han Liu, Fang Han, Ming Yuan, John Lafferty, Larry Wasserman, et al. High-dimensional semiparametric gaussian copula graphical models. *The Annals of Statistics*, 40(4):2293–2326, 2012.

Po-Ling Loh and Martin J. Wainwright. Structure estimation for discrete graphical models: generalized covariance matrices and their inverses. *Ann. Statist.*, 41(6):3022–3049, 2013. ISSN 0090-5364. doi: 10.1214/13-AOS1162. URL `https://doi.org/10.1214/13-AOS1162`.

Gábor Lugosi and Shahar Mendelson. Mean estimation and regression under heavy-tailed distributions: A survey. *Foundations of Computational Mathematics*, 19(5):1145–1190, 2019.

Silviu Maniu, Pierre Senellart, and Suraj Jog. An experimental study of the treewidth of real-world graph data (extended version). *arXiv preprint arXiv:1901.06862*, 2019.

Nicolai Meinshausen and Peter Bühlmann. Stability selection. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 72(4):417–473, 2010.

Stanley Milgram. The small world problem. *Psychology today*, 2(1):60–67, 1967.

Konstantinos Panagiotou and Angelika Steger. Maximal biconnected subgraphs of random planar graphs. *ACM Transactions on Algorithms (TALG)*, 6(2):31, 2010.

Judea Pearl. *Causality*. Cambridge University Press, Cambridge, second edition, 2009. ISBN 978-0-521-89560-6; 0-521-77362-8. doi: 10.1017/CBO9780511803161. URL `https://doi.org/10.1017/CBO9780511803161`. Models, reasoning, and inference.

Morgan N Price, Paramvir S Dehal, and Adam P Arkin. Fasttree 2–approximately maximum-likelihood trees for large alignments. *PloS one*, 5(3):e9490, 2010.

David Rossell and Piotr Zwiernik. Dependence in elliptical partial correlation graphs. *arXiv preprint arXiv:2004.13779*, 2020.

Alessandro Rudi, Raffaello Camoriano, and Lorenzo Rosasco. Less is more: Nyström computational regularization. In *Advances in Neural Information Processing Systems*, pages 1657–1665, 2015.

Richard Scheines, Peter Spirtes, Clark Glymour, Christopher Meek, and Thomas Richardson. The tetrad project: Constraint based aids to causal model specification. *Multivariate Behavioral Research*, 33(1):65–117, 1998.

Peter Spirtes, Clark Glymour, and Richard Scheines. *Causation, prediction, and search*. Adaptive Computation and Machine Learning. MIT Press, Cambridge, MA, second edition, 2000. ISBN 0-262-19440-6. With additional material by David Heckerman, Christopher Meek, Gregory F. Cooper and Thomas Richardson, A Bradford Book.

Seth Sullivant, Kelli Talaska, and Jan Draisma. Trek separation for Gaussian graphical models. *The Annals of Statistics*, pages 1665–1685, 2010.

Antal Van den Bosch, Toine Bogers, and Maurice De Kunder. Estimating search engine index size variability: a 9-year longitudinal study. *Scientometrics*, 107(2):839–856, 2016.

V.N. Vapnik and A.Ya. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications*, 16:264–280, 1971.

Martin J Wainwright, Michael I Jordan, et al. Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 1(1–2):1–305, 2008.

Ming Yuan and Yi Lin. Model selection and estimation in the gaussian graphical model. *Biometrika*, 94(1):19–35, 2007.

Fuzhen Zhang, editor. *The Schur complement and its applications*, volume 4 of *Numerical Methods and Algorithms*. Springer-Verlag, New York, 2005. ISBN 0-387-24271-6. doi: 10.1007/b105056. URL `https://doi.org/10.1007/b105056`.

Richard Y Zhang, Salar Fattahi, and Somayeh Sojoudi. Large-scale sparse inverse covariance estimation via thresholding and max-det matrix completion. *arXiv preprint arXiv:1802.04911*, 2018.

Xiujun Zhang, Xing-Ming Zhao, Kun He, Le Lu, Yongwei Cao, Jingdong Liu, Jin-Kao Hao, Zhi-Ping Liu, and Luonan Chen. Inferring gene regulatory networks from gene expression data by path consistency algorithm based on conditional mutual information. *Bioinformatics*, 28(1):98–104, 2011.

Piotr Zwiernik. Latent tree models. In *Handbook of graphical models*, Chapman & Hall/CRC Handb. Mod. Stat. Methods, pages 265–288. CRC Press, Boca Raton, FL, 2019.