

A Junction Tree Framework for Undirected Graphical Model Selection

Divyanshu Vats¹
Electrical and Computer Engineering
Rice University
dvats@rice.edu

Robert D. Nowak
Electrical and Computer Engineering
University of Wisconsin - Madison
nowak@ece.wisc.edu

Abstract

An undirected graphical model is a joint probability distribution defined on an undirected graph G^* , where the vertices in the graph index a collection of random variables and the edges encode conditional independence relationships amongst random variables. The undirected graphical model selection (UGMS) problem is to estimate the graph G^* given observations drawn from the undirected graphical model. This paper proposes a framework for decomposing the UGMS problem into multiple subproblems over clusters and subsets of the separators in a junction tree. The junction tree is constructed using a graph that contains a superset of the edges in G^* . We highlight three main properties of using junction trees for UGMS. First, different regularization parameters or different UGMS algorithms can be used to learn different parts of the graph. This is possible since the subproblems we identify can be solved independently of each other. Second, under certain conditions, a junction tree based UGMS algorithm can produce consistent results with exponentially fewer observations than the usual requirements of existing algorithms. Third, both our theoretical and experimental results show that the junction tree framework does a significantly better job at finding the weakest edges in a graph than existing methods. This property is a consequence of both the first and second properties. Finally, we note that our framework is independent of the choice of the UGMS algorithm and can be used as a wrapper around standard UGMS algorithms for more accurate graph estimation.

Keywords: Graphical models; Markov random fields; Junction trees; model selection; graphical model selection; high-dimensional statistics; graph decomposition.

1. Introduction

An undirected graphical model is a joint probability distribution P_X of a random vector X defined on an undirected graph G^* . The graph G^* consists of a set of vertices $V = \{1, \dots, p\}$ and a set of edges $E(G^*) \subseteq V \times V$. The vertices index the p random variables in X and the edges $E(G^*)$ characterize conditional independence relationships amongst the random variables in X [25]. We study undirected graphical models (also known as Markov random fields) so that the graph G^* is undirected, i.e., if an edge $(i, j) \in E(G^*)$, then $(j, i) \in E(G^*)$. The undirected graphical model selection (UGMS) problem is to estimate G^* given n observations $\mathfrak{X}^n = (X^{(1)}, \dots, X^{(n)})$ drawn from P_X . This problem is of interest in many areas including biological data analysis, financial analysis, and social network analysis; see [22] for some more examples.

This paper studies the following problem: *Given the observations \mathfrak{X}^n drawn from P_X and a graph H that contains all the true edges $E(G^*)$, and possibly some extra edges, estimate the graph G^* .*

A natural question to ask is how can the graph H be selected in the first place? One way of doing so is to use screening algorithms, such as [13] and [47], to eliminate edges that are clearly non-existent in G^* . Another method can be to use partial prior information about X to remove unnecessary edges. For example, this could be based on (i) prior knowledge about statistical properties of genes when analyzing gene expressions, (ii) prior knowledge about companies when analyzing stock returns, or (iii) demographic information when modeling social networks. Yet another method can be to use clever model selection algorithms that estimate more edges than desired. Assuming an initial graph H has been computed, our main contribution in this paper is to show how a junction tree representation of H can be used as a wrapper around UGMS algorithms for more accurate graph estimation.

¹Also affiliated with the Institute for Mathematics and its Applications, University of Minnesota - Twin Cities.

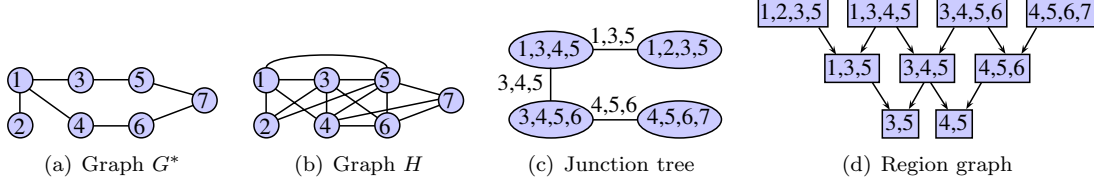


Figure 1: We want to estimate the graph in (a) using the graph (b). Our framework computes the junction tree in (c) and uses a region graph representation in (d) of the junction tree to decompose the UGMS problem into multiple subproblems.

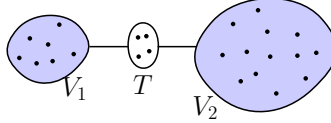


Figure 2: Structure of the graph G^* we assume to analyze the junction tree framework for UGMS.

1.1 Overview of the Junction Tree Framework

A junction tree is a tree-structured representation of an arbitrary graph [39]. The vertices in a junction tree are clusters of vertices from the original graph. An edge in a junction tree connects two clusters. Junction trees are used in many applications to reduce the computational complexity of solving various graph related problems [4]. Figure 1(c) shows an example of a junction tree for the graph in Figure 1(b). Notice that each edge in the junction tree is labeled by the set of vertices common to both clusters connected by the edge. These set of vertices are referred to as a *separator*.

Let H be a graph that contains all the edges in G^* . We show that the UGMS problem can be decomposed into multiple subproblems over *clusters* and *subsets of the separators* in a junction tree representation of H . In particular, using the junction tree, we construct a region graph, which is a directed graph over clusters of vertices. An example of a region graph for the junction tree in Figure 1(c) is shown in Figure 1(d). The first two rows in the region graph are the clusters and separators of the junction tree, respectively. The rest of the rows contain subsets of the separators². The multiple subproblems we identify correspond to estimating a subset of edges over each cluster in the region graph. For example, the subproblem over the cluster $\{1, 2, 3, 5\}$ in Figure 1(d) estimates the edges $(1, 2)$, $(1, 3)$, and $(1, 5)$.

We solve the subproblems over the region graph in an iterative manner. First, all subproblems in the first row of the region graph are solved in parallel. Second, the region graph is updated taking into account the edges removed in the first step. We keep solving subproblems over rows in the region graph and update the region graph until all the edges in the graph H have been estimated.

1.2 Advantages of Using Junction Trees

We highlight three main advantages of the junction tree framework for UGMS.

Choosing Regularization Parameters and UGMS Algorithms: UGMS algorithms typically depend on a regularization parameter that controls the number of estimated edges. This regularization parameter is usually chosen using model selection algorithms such as cross validation or stability selection. Since each subproblem we identify in the region graph is solved independently, different regularization parameters can be used to learn different parts of the graph. This has advantages when the true graph G^* has different characteristics in different parts of the graph. Further, since the subproblems are independent, different UGMS algorithms can be used to learn different parts of the graph. Our numerical simulations clearly show the advantages of this property.

Reduced Sample Complexity: One of the key results of our work is to show that in many cases, the junction tree framework is capable of consistently estimating a graph under significantly weaker conditions than

²see Algorithm 1 for details on how to exactly construct the region graph.

required by previously proposed methods. For example, we show that if G^* consists of two main components that are separated by a relatively small number of vertices (see Figure 2 for a general example), then, under certain conditions, the number of observations needed for consistent estimation scales like $\log(p_{\min})$, where p_{\min} is the number of vertices in the smaller of the two components. In contrast, existing methods are consistent if the observations scale like $\log p$, where p is the total number of vertices. If the smaller component were, for example, exponentially smaller than the larger component, then the junction tree framework is consistent with about $\log \log p$ observations. For generic problems, without structure that can be exploited by the junction tree framework, we recover the standard conditions for consistency.

Learning Weak Edges: A direct consequence of choosing different regularization parameters and the reduced sample complexity is that certain weak edges, not estimated using standard algorithms, can be estimated when using the junction tree framework. We show this theoretically and using extensive numerical simulations on both synthetic and real world data.

1.3 Related Work

Several algorithms have been proposed in the literature for learning undirected graphical models. Some examples include References [1, 5, 9, 15, 20, 33, 43] for learning Gaussian graphical models, References [23, 27, 29, 30] for learning non-Gaussian graphical models, and References [2, 7, 8, 18, 19, 34, 37] for learning discrete graphical models. Although all of the above algorithms can be modified to take into account prior knowledge about a graph H that contains all the true edges (see Appendix B for some examples), our junction tree framework is fundamentally different than the standard modification of these algorithms. The main difference is that the junction tree framework allows for using the *global Markov property* of undirected graphical models (see Definition 2.1) when learning graphs. This allows for improved graph estimation, as illustrated in both our theoretical results and numerical results. Finally, we note that all of the above algorithms can be used in conjunction with the junction tree framework.

Junction trees have been used for performing probabilistic inference in graphical models [24]. This problem differs from the UGMS problem since the graph is assumed to be known in the inference problem. The use of junction trees for learning graphical models is limited to learning the direction of edges in *directed* graphical models [53]. These methods cannot be used to learn undirected graphical models. Recent work [31, 52] has shown that solutions to the graphical lasso (gLasso) [15] problem for UGMS over Gaussian graphical models can be computed, under certain conditions, by decomposing the problem over connected components of the graph computed by thresholding the empirical covariance matrix. The methods in [31, 52] are useful for computing solutions to gLasso for particular choices of the regularization parameter and not for accurately estimating graphs. Thus, when using gLasso for UGMS, we can use the methods in [31, 52] to solve gLasso when performing model selection for choosing suitable regularization parameters.

2. Preliminaries

In this Section, we review some necessary background on graphs and graphical models that we use in this paper. Section 2.1 reviews some graph theoretic concepts. Section 2.2 reviews undirected graphical models. Section 2.3 formally defines the undirected graphical model selection (UGMS) problem. Section 2.4 reviews junction trees, which we use as a tool for decomposing UGMS into multiple subproblems.

2.1 Graph Theoretic Concepts

A graph is a tuple $G = (V, E(G))$, where V is a set of vertices and $E(G) \subseteq V \times V$ are edges connecting vertices in V . For any graph H , we use the notation $E(H)$ to denote its edges. We only consider undirected graphs where if $(v_1, v_2) \in E(G)$, then $(v_2, v_1) \in E(G)$ for $v_1, v_2 \in V$. Some graph theoretic notations that we use in this paper are summarized as follows:

- Neighbor $ne_G(i)$: Set of nodes connected to i .
- Path $\{i, s_1, \dots, s_d, j\}$: A sequence of nodes such that $(i, s_1), (s_d, j), (s_k, s_{k+1}) \in E$ for $k = 1, \dots, d-1$.
- Separator S : A set of nodes such that all paths from i to j contain at least one node in S . The separator S is *minimal* if no proper subset of S separates i and j .

- Induced Subgraph $G[A] = (A, E(G[A]))$: A graph over the nodes A such that $E(G[A])$ contains the edges only involving the nodes in A .
- Complete graph K_A : A graph that contains all possible edges over the nodes A .

For two graphs $G_1 = (V_1, E(G_1))$ and $G_2 = (V_2, E(G_2))$, define the following operations:

- Graph Union: $G_1 \cup G_2 = (V_1 \cup V_2, E_1 \cup E_2)$.
- Graph Difference: $G_1 \setminus G_2 = (V_1, E_1 \setminus E_2)$.

2.2 Undirected Graphical Models

Definition 2.1 (Undirected Graphical Model [25]). *An undirected graphical model is a probability distribution P_X defined on a graph $G^* = (V, E(G^*))$, where $V = \{1, \dots, p\}$ indexes the random vector $X = (X_1, \dots, X_p)$ and the edges $E(G^*)$ encode the following Markov property: for a set of nodes A , B , and S , if S separates A and B , then $X_A \perp\!\!\!\perp X_B | X_S$.*

The Markov property outlined above is referred to as the *global Markov property*. Undirected graphical models are also referred to as Markov random fields or Markov networks in the literature. When the joint probability distribution P_X is non-degenerate, i.e., $P_X > 0$, the Markov properties in Definition 2.1 are equivalent to the pairwise and local Markov properties:

- Pairwise Markov property: For all $(i, j) \notin E$, $X_i \perp\!\!\!\perp X_j | X_{V \setminus \{i, j\}}$.
- Local Markov property: For all $i \in V$, $X_i \perp\!\!\!\perp X_{V \setminus \{ne_G(i) \cup \{i\}\}} | X_{ne_G(i)}$.

In this paper, we always assume $P_X > 0$ and say P_X is *Markov* on G to reflect the Markov properties. Examples of conditional independence relations conveyed by a probability distribution defined on the graph in Figure 3(d) are $X_1 \perp\!\!\!\perp X_6 | \{X_2, X_4\}$ and $X_4 \perp\!\!\!\perp X_6 | \{X_2, X_5, X_8\}$.

2.3 Undirected Graphical Model Selection (UGMS)

Definition 2.2 (UGMS). *The undirected graphical model selection (UGMS) problem is to estimate a graph G^* such that the joint probability distribution P_X is Markov on G^* , but not Markov on any subgraph of G^* .*

The last statement in Definition 2.2 is important, since, if P_X is Markov on G^* , then it is also Markov on any graph that contains G^* . For example, all probability distributions are Markov on the complete graph. Thus, the UGMS problem is to find the minimal graph that captures the Markov properties associated with a joint probability distribution. In the literature, this is also known as finding the minimal I-map.

Let Ψ be an abstract UGMS algorithm that takes as inputs a set of n i.i.d. observations $\mathfrak{X}^n = \{X^{(1)}, \dots, X^{(n)}\}$ drawn from P_X and a regularization parameter λ_n . The output of Ψ is a graph \hat{G}_n , where λ_n controls the number of edges estimated in \hat{G}_n . Note the dependence of the regularization parameter on n . We assume Ψ is consistent, which is formalized in the following assumption.

Assumption 1. *There exists a λ_n for which $P(\hat{G}_n = G^*) \rightarrow 1$ as $n \rightarrow \infty$, where $\hat{G}_n = \Psi(\mathfrak{X}^n, \lambda_n)$.*

We give examples of Ψ in Appendix B. Assumption 1 also takes into account the high-dimensional case when p depends on n in such a way that $p \rightarrow \infty$ and $n \rightarrow \infty$.

2.4 Junction Trees

Junction trees [39] are used extensively for efficiently solving various graph related problems, see [4] for some examples. Reference [24] shows how junction trees can be used for exact inference (computing marginal distribution given a joint distribution) over graphical models. We use junction trees as a tool for decomposing the UGMS problem into multiple subproblems.

Definition 2.3 (Junction tree). *For an undirected graph $G = (V, E(G))$, a junction tree $\mathcal{J} = (\mathcal{C}, E(\mathcal{J}))$ is a graph over clusters of nodes in V such that*

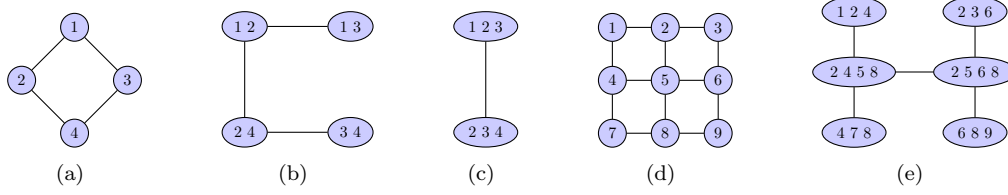


Figure 3: (a) An undirected graph, (b) Not a valid junction tree since $\{1, 2\}$ separates $\{1, 3\}$ and $\{3, 4\}$, but $3 \notin \{1, 2\}$. (c) A valid junction tree for the graph in (a). (d) A grid graph. (e) Junction tree representation of (d).

- (i) Each node in V is associated with at least one cluster in \mathcal{C} .
- (ii) For every edge $(i, j) \in E(G)$, there exists a cluster $C_k \in \mathcal{C}$ such that $i, j \in C_k$.
- (iii) \mathcal{J} satisfies the running intersection property: For all clusters C_u , C_v , and C_w such that C_w separates C_u and C_v in the tree defined by $E(\mathcal{J})$, $C_u \cap C_v \subset C_w$.

The first property in Definition 2.3 says that all nodes must be mapped to at least one cluster of the junction tree. The second property states that each edge of the original graph must be contained within a cluster. The third property, known as the running intersection property, is the most important since it restricts the clusters and the trees that can be formed. For example [49], consider the graph in Figure 3(a). By simply clustering the nodes over edges, as done in Figure 3(b), we can *not* get a valid junction tree. By making appropriate clusters of size three, we get a valid junction tree in Fig. 3(c). In other words, the running intersection property says that for two clusters with a common node, all the clusters on the path between the two clusters must contain that common node.

Proposition 2.1 ([39]). *Let $\mathcal{J} = (\mathcal{C}, E(\mathcal{J}))$ be a junction tree of the graph G . Let $S_{uv} = C_u \cap C_v$. For each $(C_u, C_v) \in \mathcal{E}$, we have the following properties:*

1. $S_{uv} \neq \emptyset$.
2. S_{uv} separates $C_u \setminus S_{uv}$ and $C_v \setminus S_{uv}$.

The set of nodes S_{uv} on the edges are called the *separators* of the junction tree. Proposition 2.1 says that all clusters connected by an edge in the junction tree have at least one common node and the common nodes separate nodes in each cluster. For example, consider the junction tree in Figure 3(e) of the graph in Figure 3(d). We can infer that 1 and 5 are separated by 2 and 4. Similarly, we can also infer that 4 and 6 are separated by 2, 5, and 8. It is clear that if a graphical model is defined on the graph, then the separators can be used to easily define conditional independence relationships. For example, using Figure 3(e), we can conclude that $X_1 \perp\!\!\!\perp X_5$ given X_2 and X_4 . As we will see in later Sections, Proposition 2.1 allow the decomposition of UGMS into multiple subproblems over clusters and subsets of the separators in a junction tree.

3. Paper Organization

The rest of the paper is organized as follows:

- Section 4 shows how junction trees can be represented as region graphs and outlines an algorithm for constructing a region graph from a junction tree.
- Section 5 shows how the region graphs can be used to apply a UGMS algorithm to the clusters and separators of a junction tree.
- Section 6 presents our main framework for using junction trees for UGMS. In particular, we show how the methods in Sections 4-5 can be used iteratively to estimate a graph.
- Section 7 reviews the PC-Algorithm, which we use to study the theoretical properties of the junction tree framework.

- Section 8 presents theoretical results on the sample complexity of learning graphical models using the junction tree framework. We also highlight advantages of using the junction tree framework as summarized in Section 1.2.
- Section 9 presents numerical simulations to highlight the advantages of using junction trees for UGMS in practice.
- Section 10 summarizes the paper and outlines some future work.

4. Overview of Region Graphs

In this Section, we show how junction trees can be represented as region graphs. As we will see in Section 6, region graphs allow us to easily decompose the UGMS problem into multiple subproblems. There are many different types of region graphs and we refer the readers to [54] for a comprehensive discussion about region graphs and how they are useful for characterizing graphical models. The region graph we present in this Section differs slightly from the standard definition of region graphs. This is mainly because our goal is to estimate edges, while the classical region graphs defined in the literature are used for computations over graphical models.

A *region* is a collection of nodes, which in our context can be the clusters of the junction tree, separators of the junction tree, or subsets of the separators. A region graph $\mathcal{G} = (\mathcal{R}, \vec{E}(\mathcal{G}))$ is a directed graph where the vertices are regions and the edges represent directed edges from one region to another. We use the notation $\vec{E}(\cdot)$ to emphasize that region graphs contain directed edges. A description of region graphs is given as follows.

- The set $\vec{E}(\mathcal{G})$ contains directed edges so that if $(R, S) \in \vec{E}(\mathcal{G})$, there exists a directed edge from region R to region S .
- Whenever $R \longrightarrow S$, then $S \subseteq R$.

Algorithm 1 outlines an algorithm to construct region graphs given a junction tree representation of some graph H . We associate a label l with every region in \mathcal{R} and group regions with the same label to partition \mathcal{R} into L groups $\mathcal{R}_1, \dots, \mathcal{R}_L$. In Algorithm 1, we initialize \mathcal{R}_1 and \mathcal{R}_2 to be the clusters and separators of a junction tree \mathcal{J} , respectively, and then iteratively find $\mathcal{R}_3, \dots, \mathcal{R}_L$ by computing all possible intersections of regions with the same label. The edges in $\vec{E}(\mathcal{G})$ are only drawn from a region in \mathcal{R}_l to a region in \mathcal{R}_{l+1} . Figure 4(c) shows an example of a region graph computed using the junction tree in Figure 4(b).

Algorithm 1: Constructing region graphs

Input: A junction tree $\mathcal{J} = (\mathcal{C}, E(\mathcal{J}))$ of a graph H .

Output: A region graph $\mathcal{G} = (\mathcal{R}, \vec{E}(\mathcal{G}))$.

- 1 $\mathcal{R}_1 = \mathcal{C}$, where \mathcal{C} are the clusters of the junction tree \mathcal{J} .
 - 2 Let \mathcal{R}_2 be all the separators of \mathcal{J} , i.e., $\mathcal{R}_2 = \{S_{uv} = C_u \cap C_v : (C_u, C_v) \in E(\mathcal{J})\}$.
 - 3 To find \mathcal{R}_3 , find all possible pairwise intersections of regions in \mathcal{R}_2 . Add all intersecting regions with cardinality greater than one to \mathcal{R}_3 .
 - 4 Repeat previous step to construct $\mathcal{R}_4, \dots, \mathcal{R}_L$ until there are no more intersecting regions of cardinality greater than one.
 - 5 For $R \in \mathcal{R}_l$ and $S \in \mathcal{R}_{l+1}$, add the edge (R, S) to $\vec{E}(\mathcal{G})$ if $S \subseteq R$.
 - 6 Let $\mathcal{R} = \{\mathcal{R}_1, \dots, \mathcal{R}_L\}$.
-

Remark 4.1. For every junction tree, Algorithm 1 outputs a unique region graph. The junction tree only characterizes the relationship between the clusters in a junction tree. A region graph extends the junction tree representation to characterize the relationships between the clusters as well as the separators. For example, in Figure 4(c), the region $\{5, 6\}$ is in the third row and is a subset of two separators of the junction tree. Thus, the only difference between the region graph and the junction tree is the additional set of regions introduced in $\mathcal{R}_3, \dots, \mathcal{R}_L$.

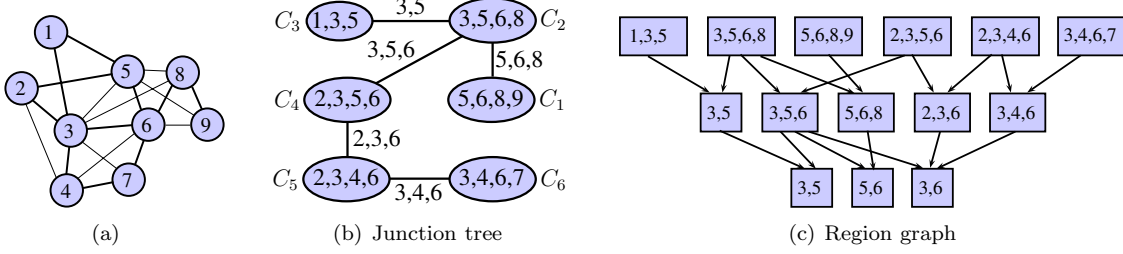


Figure 4: (a) An example of H . (b) A junction tree representation of H . (c) A region graph representation of (b) computed using Algorithm 1.

Remark 4.2. Using methods in [48], we can always construct junction trees such that the region graph will only have two labels. However, in this case, the size of the regions or clusters may be too large. This may not be desirable since the computational complexity of applying UGMS algorithms to region graphs, as shown in Section 6, depends on the size of the regions.

Remark 4.3. From the construction in Algorithm 1, \mathcal{R} may have two or more regions that are the same but have different labels. For example, in Figure 4(c), the region $\{3, 5\}$ is in both \mathcal{R}_2 and \mathcal{R}_3 . We can avoid this situation by removing $\{3, 5\}$ from \mathcal{R}_2 and adding an edge from the region $\{1, 3, 5\}$ in \mathcal{R}_1 to the region $\{3, 5\}$ in \mathcal{R}_3 . For notational simplicity and for the purpose of illustration, we allow for duplicate regions. This does not change the theory or the algorithms that we develop.

5. Applying UGMS to Region Graphs

Before presenting our framework for decomposing UGMS into multiple subproblems, we first show how UGMS algorithms can be applied to estimate a subset of edges in a region of a region graph. In particular, for a region graph $\mathcal{G} = (\mathcal{R}, \vec{E}(\mathcal{G}))$, we want to identify a set of edges in the induced subgraph $H[R]$ that can be estimated by applying a UGMS algorithm to either R or a set of vertices that contains R . With this goal in mind, define the children $ch(R)$ of a region R as follows:

$$\text{Children: } ch(R) = \left\{ S : (R, S) \in \vec{E} \right\}. \quad (1)$$

We say R connects to S if $(R, S) \in \vec{E}(\mathcal{G})$. Thus, the children in (1) contains all regions that R connects to. For example, in Figure 4(c),

$$ch(\{2, 3, 4, 6\}) = \{\{2, 3, 6\}, \{3, 4, 6\}\}.$$

If there exists a direct path from S to R , we say S is an *ancestor* of R . The set of all ancestors of R is denoted by $an(R)$. For example, in Figure 4(c),

$$\begin{aligned} an(\{5, 6, 8, 9\}) &= \emptyset, \\ an(\{3, 5, 6\}) &= \{\{3, 5, 6, 8\}, \{2, 3, 5, 6\}\}, \text{ and} \\ an(\{3, 6\}) &= \{\{3, 5, 6\}, \{2, 3, 6\}, \{3, 4, 6\}, \{2, 3, 5, 6\}, \{2, 3, 4, 6\}, \{3, 4, 6, 7\}, \{3, 5, 6, 8\}\}. \end{aligned}$$

The notation \overline{R} takes the union of all regions in $an(R)$ and R so that

$$\overline{R} = \bigcup_{S \in \{an(R), R\}} S. \quad (2)$$

Thus, \overline{R} contains the union of all clusters in the junction tree that contains R . An illustration of some of the notations defined on region graphs is shown in Figure 5. Using $ch(R)$, define the subgraph H'_R as ³

$$H'_R = H[R] \setminus \left\{ \bigcup_{S \in ch(R)} K_S \right\}, \quad (3)$$

³For graphs G_1 and G_2 , $E(G_1 \setminus G_2) = E(G_1) \setminus E(G_2)$ and $E(G_1 \cup G_2) = E(G_1) \cup E(G_2)$

Algorithm 2: UGMS over regions in a region graph

- 1: **Input:** Region graph $\mathcal{G} = (\mathcal{R}, \vec{E}(\mathcal{G}))$, a region R , observations \mathfrak{X}^n , and a UGMS algorithm Ψ .
 - 2: Compute H'_R using (3) and \bar{R} using (2).
 - 3: Apply Ψ to $\mathfrak{X}^n_{\bar{R}}$ to estimate edges in H'_R . See Appendix B for examples.
 - 4: **Return** the estimated edges \hat{E}_R .
-

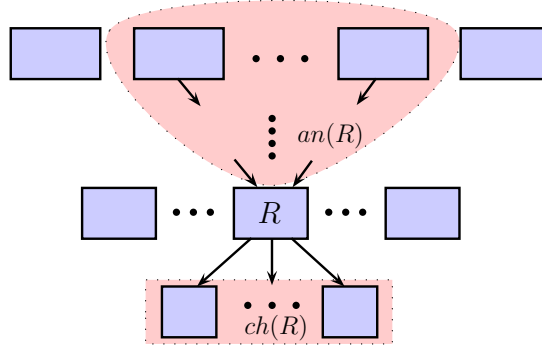


Figure 5: Notations defined on region graphs. The children $ch(R)$ are the set of regions that R connects to. The ancestors $an(R)$ are all the regions that have a directed path to the region R . The set \bar{R} takes the union of all regions in $an(R)$ and R .

where $H[R]$ is the induced subgraph that contains all edges in H over the region R and K_S is the complete graph over S . In words, H'_R is computed by removing all edges from $H[R]$ that are contained in another separator. For example, in Figure 4(c), when $R = \{5, 6, 8\}$, $E(H'_R) = \{(5, 8), (6, 8)\}$. The subgraph H'_R is important since it identifies the edges that can be estimated when applying a UGMS algorithm to the set of vertices \bar{R} .

Proposition 5.1. *Suppose $E(G^*) \subseteq E(H)$. All edges in H'_R can be estimated by solving a UGMS problem over the vertices \bar{R} .*

Proof See Appendix C. □

Proposition 5.1 says that all edges in H'_R can be estimated by applying a UGMS algorithm to the set of vertices \bar{R} . The intuition behind the result is that only those edges in the region R can be estimated whose Markov properties can be deduced using the vertices in \bar{R} . Moreover, the edges *not estimated* in $H[R]$ share an edge with another region that does not contain all the vertices in R . Algorithm 2 summarizes the steps involved in estimating H'_R using the UGMS algorithm Ψ defined in Section 2.3. Some examples on how to use Algorithm 2 to estimate some edges of the graph in Figure 4(a) using the region graph in Figure 4(c) are described as follows.

1. Let $R = \{1, 3, 5\}$. This region only connects to $\{3, 5\}$. This means that all edges, except the edge $(3, 5)$ in $H[R]$, can be estimated by applying Ψ to R .
2. Let $R = \{3, 5, 6\}$. The children of this region are $\{3, 5\}$, $\{5, 6\}$, and $\{3, 6\}$. This means that $H'_R = \emptyset$, i.e., no edge over $H[R]$ can be estimated by applying Ψ to $\{3, 5, 6\}$.
3. Let $R = \{3, 4, 6\}$. This region only connects to $\{3, 6\}$. Thus, all edges except $(3, 6)$ can be estimated. The regions $\{2, 3, 4, 6\}$ and $\{3, 4, 6, 7\}$ connect to R , so Ψ needs to be applied to $\bar{R} = \{2, 3, 4, 6, 7\}$.

6. UGMS Using Junction Trees: A General Framework

In this Section, we present the main junction tree framework for UGMS using the results from Sections 4-5. Section 6.1 presents the junction tree framework. Section 6.2 discusses the computational complexity of the

Notation	Description
$G^* = (V, E(G^*))$	Unknown graph that we want to estimate.
H	Known graph such that $E(G^*) \subseteq E(H)$.
$\mathcal{G} = (\mathcal{R}, \vec{E}(\mathcal{G}))$	Region graph of \mathcal{J} constructed using Algorithm 1.
$\mathcal{R} = (\mathcal{R}_1, \dots, \mathcal{R}_L)$	Partitioning of the regions in \mathcal{R} into L labels.
\bar{R}	The set of vertices used when applying Ψ to estimate edges over R .
H'_R	Edges in $H[R]$ that can be estimated using Algorithm 2. See (3).

Table 1: A summary of some notations.

framework. Section 6.3 highlights the advantages of using junction trees for UGMS using some examples. We refer to Table 1 for a summary of all the notations that we use in this Section.

6.1 Description of Framework

Recall that Algorithm 2 shows that to estimate a subset of edges in $H[R]$, where R is a region in the region graph \mathcal{G} , the UGMS algorithm Ψ in Assumption 1 needs to be applied to the set \bar{R} defined in (2). Given this result, a straightforward approach to decomposing the UGMS problem is to apply Algorithm 2 to each region R and combine all the estimated edges. This will work since for any $R, S \in \mathcal{R}$ such that $R \neq S$, $E(H'_R) \cap E(H'_S) = \emptyset$. This means that each application of Algorithm 2 estimates a different set of edges in the graph. However, for some edges, this may require applying a UGMS algorithm to a large set of nodes. For example, in Figure 4(c), when applying Algorithm 2 to $R = \{3, 6\}$, the UGMS algorithm needs to be applied to $\bar{R} = \{2, 3, 4, 5, 6, 7, 8\}$, which is almost the full set of vertices. To reduce the problem size of the subproblems, we apply Algorithms 1 and 2 in an iterative manner as outlined in Algorithm 3.

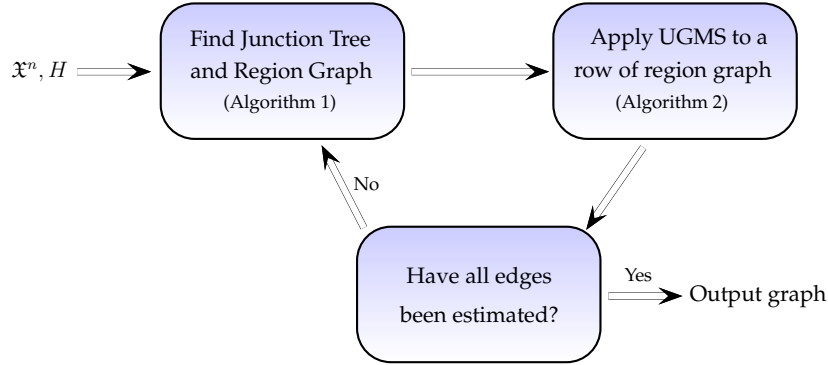


Figure 6: A high level overview of the junction tree framework for UGMS in Algorithm 3.

Figure 6 shows a high level description of Algorithm 3. We first find a junction tree and then a region graph of the graph H using Algorithm 1. We then find the row in the region graph over which edges can be estimated and apply Algorithm 2 to each region in that row. We note that when estimating edges over a region, we use model selection algorithms to choose an appropriate regularization parameter to select the number of edges to estimate. Next, all estimated edges are added to \hat{G} and all edges that are estimated are removed from H . Thus, H now represents all the edges that are left to be estimated and $\hat{G} \cup H$ contains all the edges in G^* . We repeat the above steps on a new region graph computed using $\hat{G} \cup H$ and stop the algorithm when H is an empty graph.

An example illustrating the junction tree framework is shown in Figure 7. The region graph in Figure 7(b) is constructed using the graph H in Figure 7(a). The true graph G^* we want to estimate is shown in Figure 1(a). The top and bottom in Figure 7(c) show the graphs \hat{G} and H , respectively, after estimating all the edges in \mathcal{R}_1 of Figure 7(b). The edges in \hat{G} are represented by double lines to distinguish them from the edges in H . Figure 7(d) shows the region graph of $\hat{G} \cup H$. Figure 7(e) shows the updated \hat{G} and H where only the edges (4, 5) and (5, 6) are left to be estimated. This is done by applying Algorithm 2 to the regions

Algorithm 3: Junction Tree Framework for UGMS

See Table 1 for some notations.

Step 1. Initialize \hat{G} so that $E(\hat{G}) = \emptyset$ and find the region graph \mathcal{G} of H .

Step 2. Find the smallest ℓ such that there exists a region $R \in \mathcal{R}_\ell$ such that $E(H'_R) \neq \emptyset$.

Step 3. Apply Algorithm 2 to each region in \mathcal{R}_ℓ .

Step 4. Add all estimated edges to \hat{G} and *remove edges* from H that have been estimated. Now $H \cup \hat{G}$ contains all the edges in G^* .

Step 5. Compute a new junction tree and region graph \mathcal{G} using the graph $\hat{G} \cup H$.

Step 6. If $E(H) = \emptyset$, stop the algorithm, else go to Step 2.

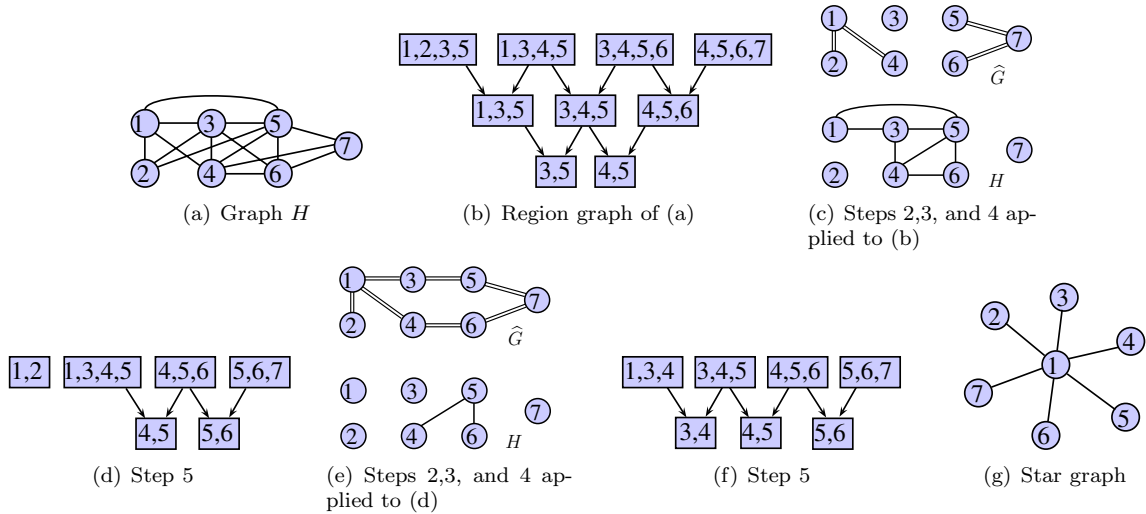


Figure 7: Example to illustrate the junction tree framework in Section 6.1.

in \mathcal{R}_2 of Figure 7(f). Notice that we did not include the region $\{1, 2\}$ in the last region graph since we know all edges in this region have already been estimated. In general, if $E(H[R]) = \emptyset$ for any region R , we can remove this region and thereby reduce the computational complexity of constructing region graphs.

6.2 Computational Complexity

In this Section, we discuss the computational complexity of the junction tree framework. It is difficult to write down a closed form expression since the computational complexity depends on the structure of the junction tree. Moreover, merging clusters in the junction tree can easily control the computations. With this in mind, the main aim in this Section is to show that the complexity of the framework is roughly the same as that of applying a standard UGMS algorithm. Consider the following observations.

1. *Computing H* : Assuming no prior knowledge about H is given, this graph needs to be computed from the observations. This can be done using standard screening algorithms, such as those in [13, 47], or by applying a UGMS algorithm with a regularization parameter that selects a larger number of edges (than that computed by using a standard UGMS algorithm). Thus, the complexity of computing H is roughly the same as that of applying a UGMS algorithm to all the vertices in the graph.
2. *Applying UGMS to regions*: Recall from Algorithm 2 that we apply a UGMS algorithm to observations over \bar{R} to estimate edges over the vertices R , where R is a region in a region graph representation of

H . Since $|\overline{R}| \leq p$, it is clear that the complexity of Algorithm 2 is less than that of applying a UGMS algorithm to estimate all edges in the graph.

3. *Computing junction trees*: There are many different junction tree representations of a given graph. In the literature, an optimal junction tree is defined as having the minimal maximum width, where the width of a junction tree is the maximum size of the cluster. Finding an optimal junction tree is known to be computationally intractable [3]. However, several tractable algorithms have been proposed in the literature that compute a close to optimal junction tree in time at most $O(p^3)$ [6, 21]. This time complexity is less than that of standard UGMS algorithms.

It is clear that the complexity of all the intermediate steps in the framework is less than that of applying a standard UGMS algorithm. The overall complexity of the framework depends on the number of clusters in the junction tree and the size of the separators in the junction tree. The size of the separators in a junction tree can be controlled by merging clusters that share a large separator. This step can be done in linear time. Removing large separators also reduces the total number of clusters in a junction tree. In the worst case, if all the separators in H are too large, the junction tree will only have one cluster that contains all the vertices. In this case, using the junction tree framework will be no different than using a standard UGMS algorithm.

6.3 Advantages of using Junction Trees and Region Graphs

An alternative approach to estimating G^* using H is to modify some current UGMS algorithms (see Appendix B for some concrete examples). For example, neighborhood selection based algorithms first estimate the neighborhood of each vertex and then combine all the estimated neighborhoods to construct an estimate \hat{G} of G^* [7, 33, 34, 37]. Two ways in which these algorithms can be modified when given H are described as follows:

1. A straightforward approach is to decompose the UGMS problem into p different subproblems of estimating the neighborhood of each vertex. The graph H can be used to restrict the estimated neighbors of each vertex to be subsets of the neighbors in H . For example, in Figure 7(a), the neighborhood of 1 is estimated from the set $\{2, 3, 4, 5\}$ and the neighborhood of 3 is estimated from the set $\{1, 4, 5, 6\}$. This approach can be compared to independently applying Algorithm 2 to each region in the region graph. For example, when using the region graph, the edge $(1, 4)$ can be estimated by applying a UGMS algorithm to $\{1, 3, 4, 5\}$. In comparison, when not using region graphs, the edge $(1, 4)$ is estimated by applying a UGMS algorithm to $\{1, 2, 3, 4, 5\}$. In general, using region graphs results in smaller subproblems. A good example to illustrate this is the star graph in Figure 7(g). The junction tree framework only requires applying a UGMS algorithm to a pair of nodes. On the other hand, neighborhood selection needs to be applied to all the nodes to estimate the neighbors of the central node 1 which is connected to all other nodes.
2. An alternative approach is to estimate the neighbors of each vertex in an iterative manner. However, it is not clear what ordering should be chosen for the vertices. The region graph approach outlined in Section 6.1 leads to a natural choice for choosing which edges to estimate in the graph so as to reduce the problem size of subsequent subproblems. Moreover, iteratively applying neighborhood selection may still lead to large subproblems. For example, suppose the star graph in Figure 7(g) is in fact the true graph. In this case, using neighborhood selection always leads to applying UGMS to all the nodes in the graph.

From the above discussion, it is clear that using junction trees for UGMS leads to smaller subproblems and a natural choice of an ordering for estimating edges in the graph. We will see in Section 8 that the smaller subproblems lead to weaker conditions on the number of observations required for consistent graph estimation. Moreover, our numerical simulations in Section 9 empirically show the advantages of using junction tree over neighborhood selection based algorithms.

Algorithm 4: PC-Algorithm for UGMS: $\text{PC}(\kappa, \mathfrak{X}^n, H, L)$

Inputs:

- κ : An integer that controls the computational complexity of PC.
- \mathfrak{X}^n : n i.i.d. observations.
- H : A graph that contains all the true edges G^* .
- L : A graph that contains the edges that need to be estimated.

Output: A graph \hat{G} that contains edges in H' that are estimated to be in G^* .

```

1  $\hat{G} \leftarrow L$ 
2 for each  $k \in \{0, 1, \dots, \kappa\}$  do
3   for each  $(i, j) \in E(\hat{G})$  do
4      $\mathcal{S}_{ij} \leftarrow$  Neighbors of  $i$  or  $j$  in  $H$  depending on which one has lower cardinality.
5     if  $\exists S \subset \mathcal{S}_{ij}, |S| = k, \text{ s.t. } X_i \perp\!\!\!\perp X_j | X_S$  (computed using  $\mathfrak{X}^n$ ) then
6        $\perp$  Delete edge  $(i, j)$  from  $\hat{G}$  and  $H$ .
7 Return  $\hat{G}$ .
```

7. PC-Algorithm for UGMS

So far, we have presented the junction tree framework using an abstract undirected graphical model selection (UMGS) algorithm. This shows that our framework can be used in conjunction with any UGMS algorithm. In this Section, we review the PC-Algorithm, since we use it to analyze the junction tree framework in Section 8. The PC-Algorithm was originally proposed in the literature for learning directed graphical models [43]. The first stage of the PC-Algorithm, which we refer to as PC, estimates an undirected graph using conditional independence tests. The second stage orients the edges in the undirected graph to estimate a directed graph. We use the first stage of the PC-Algorithm for UGMS. Algorithm 4 outlines PC. Variants of the PC-Algorithm for learning undirected graphical models have recently been analyzed in [1, 2]. The main property used in PC is the global Markov property of undirected graphical models which states that if a set of vertices S separates i and j , then $X_i \perp\!\!\!\perp X_j | X_S$. As seen in Line 5 of Algorithm 4, PC deletes an edge (i, j) if it identifies a conditional independence relationship. Some properties of PC are summarized as follows:

1. **Parameter κ :** PC iteratively searches for separators for an edge (i, j) by searching for separators of size $0, 1, \dots, \kappa$. This is reflected in Line 2 of Algorithm 4. Theoretically, the algorithm can automatically stop after searching for all possible separators for each edge in the graph. However, this may not be computationally tractable, which is why κ needs to be specified.
2. **Conditional Independence Test:** Line 5 of Algorithm 4 uses a conditional independence test to determine if an edge (i, j) is in the true graph. This makes PC extremely flexible since nonparametric independence tests may be used, see [16, 36, 56] for some examples. In this paper, for simplicity, we only consider Gaussian graphical models. In this case, conditional independence can be tested using the conditional correlation coefficient defined as

$$\text{Conditional correlation coefficient: } \rho_{ij|S} = \frac{\Sigma_{ij} - \Sigma_{i,S} \Sigma_{S,S}^{-1} \Sigma_{S,j}}{\sqrt{\Sigma_{i,i|S} \Sigma_{j,j|S}}}, \quad (4)$$

where $P_X \sim \mathcal{N}(0, \Sigma)$, $\Sigma_{A,B}$ is the covariance matrix of X_A and X_B , and $\Sigma_{A,B|S}$ is the conditional covariance defined by

$$\Sigma_{A,B|S} = \Sigma_{A,B} - \Sigma_{A,S} \Sigma_{S,S}^{-1} \Sigma_{B,S}. \quad (5)$$

Whenever $X_i \perp\!\!\!\perp X_j | X_S$, then $\rho_{ij|S} = 0$. This motivates the following test for independence:

$$\text{Conditional Independence Test: } |\hat{\rho}_{ij|S}| < \lambda_n \implies X_i \perp\!\!\!\perp X_j | X_S, \quad (6)$$

where $\hat{\rho}_{ij|S}$ is computed using the empirical covariance matrix from the observations \mathfrak{X}^n . The regularization parameter λ_n controls the number of edges estimated in \hat{G} .

3. **The graphs H and L :** Recall that H contains all the edges in G^* . The graph L contains edges that need to be estimated since, as seen in Algorithm 2, we apply UGMS to only certain parts of the graph instead of the whole graph. As an example, to estimate edges in a region R of a region graph representation of H , we apply Algorithm 4 as follows:

$$\hat{G}_R = \text{PC}(\eta, \mathcal{X}^n, H, H'_R), \quad (7)$$

where H'_R is defined in (3). Notice that we do not use \overline{R} in (7). This is because Line 4 of Algorithm 4 automatically finds the set of vertices to apply the PC algorithm to. Alternatively, we can apply Algorithm 4 using \overline{R} as follows:

$$\hat{G}_R = \text{PC}(\eta, \mathcal{X}_{\overline{R}}^n, K_{\overline{R}}, H'_R), \quad (8)$$

where $K_{\overline{R}}$ is the complete graph over \overline{R} .

4. **The set \mathcal{S}_{ij} :** An important step in Algorithm 4 is specifying the set \mathcal{S}_{ij} in Line 4 to restrict the search space for finding separators for an edge (i, j) . This step significantly reduces the computational complexity of PC and differentiates PC from the first stage of the SGS-Algorithm [44], which specifies $\mathcal{S}_{ij} = V \setminus \{i, j\}$.

8. Theoretical Analysis of Junction Tree based PC

We use the PC-algorithm to analyze the junction tree based UGMS algorithm. Our main result, stated in Theorem 8.1 shows that when using the PC-Algorithm with the junction tree framework, we can potentially estimate the graph using fewer number of observations than what is required by the standard PC-Algorithm. As we shall see in Theorem 8.1, the particular gain in performance depends on the structure of the graph.

Section 8.1 discusses the assumptions we place on the graphical model. Section 8.2 presents the main theoretical result highlighting the advantages of using junction trees. We use standard asymptotic notation so that $f(n) = \Omega(g(n))$ implies that there exists an N and a constant c such that for all $n \geq N$, $f(n) \geq cg(n)$. For $f(n) = O(g(n))$, replace \geq by \leq .

8.1 Assumptions

- (A1) **Gaussian graphical model:** We assume $X = (X_1, \dots, X_p) \sim P_X$, where P_X is a multivariate normal distribution with mean zero and covariance Σ . Further, P_X is Markov on G^* and not Markov on any subgraph of G^* . It is well known this assumption translates into the fact that $\Sigma_{ij}^{-1} = 0$ if and only if $(i, j) \notin G^*$ [41].
- (A2) **Faithfulness:** If $X_i \perp\!\!\!\perp X_j | X_S$, then i and j are separated by⁴ S . This assumption is important for the PC algorithm to output the correct graph. Further, note that the Markov assumption is different since it goes the other way: if i and j are separated by S , then $X_i \perp\!\!\!\perp X_j | X_S$. Thus, when both (A1) and (A2) hold, we have that $X_i \perp\!\!\!\perp X_j | X_S \iff (i, j) \notin G^*$.
- (A3) **Separator Size η :** For all $(i, j) \notin G^*$, there exists a subset of nodes $S \subset V \setminus \{i, j\}$, where $|S| \leq \eta$, such that S is a separator for i and j in G^* . This assumption allows us to use $\kappa = \eta$ when using PC.
- (A4) **Conditional Correlation Coefficient $\rho_{ij|S}$ and Σ :** Under (A3), we assume that $\rho_{ij|S}$ satisfies

$$\sup\{|\rho_{ij|S}| : i, j \in V, S \subset V, |S| \leq \eta\} \leq M < 1, \quad (9)$$

where M is a constant. Further, we assume that $\max_{i, S, |S| \leq \eta} \Sigma_{i, i|S} \leq L < \infty$.

- (A5) **High-Dimensionality** We assume that the number of vertices in the graph p scales with n so that $p \rightarrow \infty$ as $n \rightarrow \infty$. Furthermore, both $\rho_{ij|S}$ and η are assumed to be functions of n and p unless mentioned otherwise.

⁴If S is the empty set, this means that there is no edge between i and j .

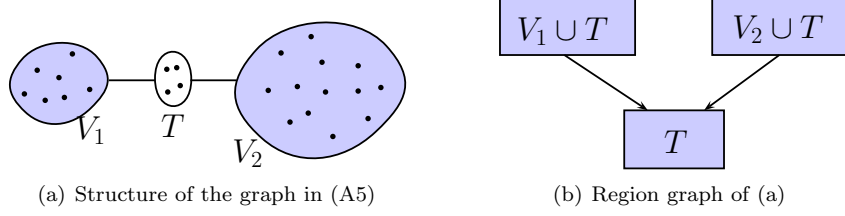


Figure 8: General Structure of the graph we use in showing the advantages of the junction tree framework.

(A6) **Structure of G^* :** Under (A3), we assume that there *exists* a set of vertices V_1 , V_2 , and T such that T separates V_1 and V_2 in G^* and $|T| < \eta$. Figure 8(a) shows the general structure of this assumption. As we will see in the next Section, this structure of the graph will allow us to apply the junction tree framework to the region graph representation in Figure 8(b).

8.2 Theoretical Result and Analysis

Recall PC in Algorithm 4. Since we assume (A1), the conditional independence test in (6) can be used in Line 5 of Algorithm 4. To analyze the junction tree framework, consider the following steps to construct \hat{G} using PC when given n i.i.d. observations \mathfrak{X}^n :

Step 1. Compute H : Apply PC using a regularization parameter λ_n^0 such that

$$H = \text{PC}(|T|, \mathfrak{X}^n, K_V, K_V), \quad (10)$$

where K_V is the complete graph over the nodes V . In the above equation, we apply PC to remove all edges in G^* for which there exists a separator of size less than or equal $|T|$.

Step 2. Estimate a subset of edges over $V_1 \cup T$ and $V_2 \cup T$ using regularization parameters λ_n^1 and λ_n^2 , respectively, such that

$$\hat{G}_{V_k} = \text{PC}(\eta, \mathfrak{X}^n, H[V_k \cup T] \cup K_T, H'_{V_k \cup T}), \text{ for } k = 1, 2. \quad (11)$$

where $H'_{V_k \cup T} = H[V_k \cup T] \setminus K_T$ as defined in (3).

Step 3. Estimate edges over T using a regularization parameter λ_n^T :

$$\hat{G}_T = \text{PC}(\eta, \mathfrak{X}^n, H[T \cup \text{ne}_{G^*}(T)], H[T]). \quad (12)$$

Step 4. Final estimate is $\hat{G} = \hat{G}_{V_1} \cup \hat{G}_{V_2} \cup \hat{G}_T$.

For the region graph in Figure 8(b), Steps 2 and 3 correspond to applying PC to the regions $V_1 \cup T$ and $V_2 \cup T$. Step 4 corresponds to applying PC to the region T and all neighbors of T in G^* . Step 4 corresponds to applying PC to the region T and all neighbors of T in G^* . Although the neighbors of T are sufficient to estimate all the edges in T , in general, depending on the graph, a smaller set of vertices is required to estimate edges in T . The main result is stated using the following terms defined on the graphical model:

$$p_1 = |V_1| + |T| \quad (13)$$

$$p_2 = |V_2| + |T| \quad (14)$$

$$p_T = |T \cup \text{ne}_{G^*}(T)| \quad (15)$$

$$\eta_T = |T| \quad (16)$$

$$\rho_0 = \inf\{|\rho_{ij|S}| : i, j \text{ s.t. } |S| \leq \eta_T \text{ \& } |\rho_{ij|S}| > 0\} \quad (17)$$

$$\rho_1 = \inf\{|\rho_{ij|S}| : i \in V_1, j \in V_1 \cup T \text{ s.t. } (i, j) \in E(G^*), S \subseteq V_1 \cup T, |S| \leq \eta\} \quad (18)$$

$$\rho_2 = \inf\{|\rho_{ij|S}| : i \in V_2, j \in V_2 \cup T \text{ s.t. } (i, j) \in E(G^*), S \subseteq V_2 \cup T, |S| \leq \eta\} \quad (19)$$

$$\rho_T = \inf\{|\rho_{ij|S}| : i, j \in T \text{ s.t. } (i, j) \in E, S \subseteq T \cup ne_G(T), \eta_T < |S| \leq \eta\}, \quad (20)$$

The term ρ_0 is a measure of how hard it is to learn the graph H in Step 1 so that $E(G^*) \subseteq E(H)$ and all edges that have a separator of size less than $|T|$ are deleted in H . The terms ρ_1 and ρ_2 are measures of how hard it is to learn the edges in $G^*[V_1 \cup T] \setminus K_T$ and $G^*[V_2 \cup T] \setminus K_T$ (Step 2), respectively, given that $E(G^*) \subseteq E(H)$. The term ρ_T is a measure of how hard it is to learn the graph over the nodes T given that we know the edges that connect V_1 to T and V_2 to T .

Theorem 8.1. *Under Assumptions (A1)-(A6), there exist a conditional independence test such that if*

$$n = \Omega\left(\max\left\{\rho_0^{-2}\eta_T \log(p), \rho_1^{-2}\eta \log(p_1), \rho_2^{-2}\eta \log(p_2), \rho_T^{-2}\eta \log(p_T)\right\}\right), \quad (21)$$

then $P(\hat{G} \neq G) \rightarrow 0$ as $n \rightarrow \infty$.

Proof See Appendix E. □

Remark 8.1 (Choice of Regularization Parameters). We use the conditional independence test in (6) that thresholds the conditional correlation coefficient. From the proof in Appendix E, the thresholds, which we refer to as the regularization parameter, are chosen as follows:

$$\lambda_n^0 = O(\rho_0) \text{ and } \lambda_n^0 = \Omega\left(\sqrt{\eta_T \log(p)/n}\right) \quad (22)$$

$$\lambda_n^k = O(\rho_k) \text{ and } \lambda_n^k = \Omega\left(\sqrt{\eta \log(p_k)/n}\right), k = 1, 2 \quad (23)$$

$$\lambda_n^T = O(\rho_T) \text{ and } \lambda_n^T = \Omega\left(\sqrt{\eta \log(p_T)/n}\right). \quad (24)$$

We clearly see that different regularization parameters are used to estimate different parts of the graph.

Remark 8.2 (Weaker Condition). If we do not use the junction tree based approach outlined in Steps 1-4 and instead directly apply PC, the sufficient condition on the number of observations will be $n = \Omega(\rho_{min}^{-2}\eta \log(p))$, where

$$\rho_{min} := \inf\{|\rho_{ij|S}| : (i, j) \in E(G^*), |S| \leq \eta\}. \quad (25)$$

This result is proved in Appendix D using results from [1, 20]. Since $\rho_{min} \leq \min\{\rho_0, \rho_1, \rho_2, \rho_T\}$, it is clear that (21) is a weaker condition. The main reason for this difference is that the junction tree approach defines an *ordering* on the edges to test if an edge belongs to the true graph. This ordering allows for a reduction in separator search space (see \mathcal{S}_{ij} in Algorithm 4) for testing edges over the set T . Standard analysis of PC assumes that the edges are tested randomly, in which case, the separator search space is always upper bounded by the full set of nodes.

Remark 8.3 (Reduced Sample Complexity). Suppose η , ρ_0 , and ρ_T are constants and $\rho_1 < \rho_2$. In this case, (21) reduces to

$$n = \Omega\left(\max\left\{\log(p), \rho_1^{-2} \log(p_1), \rho_2^{-2} \log(p_2)\right\}\right). \quad (26)$$

If $\rho_1^{-2} = \Omega\left(\max\left\{\rho_2^{-2} \log(p_2)/\log(p_1), \log(p)\right\}\right)$, then (26) reduces to

$$n = \Omega\left(\rho_1^{-2} \log(p_1)\right). \quad (27)$$

On the other, if we do not use junction trees, $n = \Omega(\rho_{min}^{-2} \log(p))$, where $\rho_{min} \leq \rho_1$. Thus, if $p_1 \ll p$, for example $p_1 = \log(p)$, then using the junction tree based PC requires lower number of observations for consistent UGMS. Informally, the above condition says that if the graph structure in (A6) is easy to identify, $p_1 \ll p_2$, and the minimal conditional correlation coefficient over the true edges lies in the smaller cluster (but not over the separator), the junction tree framework can accurately learn the graph using significantly less number of observations.

Remark 8.4 (Learning Weak Edges). We now analyze Theorem 8.1 to see how the partial correlations scale for high-dimensional consistency. Under the assumption in Remark 8.3, it is easy to see that the minimal partial correlation scales as $\Omega(\sqrt{\log(p_1)/n})$ when using junction trees and as $\Omega(\sqrt{\log(p)/n})$ when not using junction trees. Thus, it is clear that when $p_1 \ll p$, it is possible to learn edges with weaker partial correlation when using junction trees.

Remark 8.5 (Computational complexity). It is easy to see that the worst case computational complexity of the PC-Algorithm is $O(p^{\eta+2})$ since there are $O(p^2)$ edges and testing for each edges requires a search over at most $O(p^\eta)$ separators. The worst case computational complexity of Steps 1-4 is roughly $O(p^{|T|+2} + p_1^{\eta+2} + p_2^{\eta+2} + p_T^{\eta+2})$. Under the conditions in Remark 8.3 and when $p_1 \ll p$, this complexity is roughly $O(p^{\eta+2})$, which is the same as the standard PC-Algorithm. In practice, especially when the graph is sparse, the computational complexity is much less than $O(p^{\eta+2})$ since the PC-Algorithm restricts the search space for finding separators.

Remark 8.6 (Extensions). We have analyzed the junction tree framework assuming that the junction tree of H only has two clusters. Extending the analysis to junction trees with more than two clusters is trivial. Moreover, the analysis can be easily extended to graphical models that takes values in some discrete space. In this case, the conditional independence test can be done using the empirical conditional mutual information. Further, we presented theoretical results only for the PC-Algorithm. Similar results, under different assumptions, can be obtained when analyzing other UGMS algorithms.

9. Numerical Simulations

In this Section, we present numerical simulations that highlight the advantages of using the junction tree framework for UGMS. Throughout this Section, we assume a Gaussian graphical model such that $P_X \sim \mathcal{N}(0, \Theta^{-1})$ is Markov on G^* . It is well known that this implies that $(i, j) \notin G^* \iff \Theta_{ij} = 0$ [42]. Some algorithmic details used in the simulations are described as follows.

Computing H : We apply Algorithm 4 with a suitable value of κ in such a way that the separator search space \mathcal{S}_{ij} (see Line 4) is restricted to be small. We use the conditional partial correlation to test for conditional independence and choose a *separate threshold* to test for each edge in the graph. The thresholds for the conditional independence test are computed using 5-fold cross-validation. The computational complexity of this step is roughly $O(p^2)$ since there are $O(p^2)$ edges to be tested. Note that this method for computing H is equivalent to Step 1 in Section 8.2 with $|T| = \kappa$. Finally, we note that the above method does not guarantee that all edges in G^* will be included in H . This can result in false edges being included in the junction tree estimated graphs. To avoid this situation, once a graph estimate \hat{G} has been computed using the junction tree based UGMS algorithm, we apply conditional independence tests again to prune the estimated edge set.

Computing the junction tree: We use standard algorithms in the literature for computing close to optimal junction tree⁵. Once the junction tree is computed, we merge clusters so that the maximum size of the separator is at most $\kappa + 1$, where κ is the parameter used when computing the graph H .

UGMS Algorithms: We apply the junction tree framework in conjunction with graphical Lasso (gL) [5], neighborhood selection using Lasso (nL) [33], and the PC-Algorithm (PC) [43]. See Appendix B for a review of gL and nL and Algorithm 4 for PC. When using nL, we use the intersection rule to combine neighborhood estimates. Further, we use the adaptive Lasso [57] for finding neighbors of a vertex since this is known to give superior results for variable selection [46].

Choosing Regularization Parameters: An important step when applying UGMS algorithms is to choose a suitable regularization parameter. It is now well known that classical methods, such as cross-validation and information criterion based methods, tend to choose a much larger number of edges when compared to an oracle estimator for high-dimensional problems [28, 32]. Several alternative methods have been proposed in the literature; see for example stability selection [28, 32] and extended Bayesian information (EBIC) criterion [11, 14]. In all our simulations, we use EBIC since it is much faster than stability based methods

⁵We use the GreedyFillin heuristic. This is known to give good results with reasonable computational time [21].

when the distribution is Gaussian. EBIC selects a regularization parameter $\hat{\lambda}_n$ as follows:

$$\hat{\lambda}_n = \min_{\lambda_n > 0} \left\{ n \left[\log \det \hat{\Theta}_{\lambda_n} - \text{trace}(\hat{S}\mathbf{x}^n) \right] + |E(\hat{G}_{\lambda_n})| \log n + 4\gamma |E(\hat{G}_{\lambda_n})| \log p \right\}, \quad (28)$$

where $\hat{\Theta}_{\lambda_n}$ is the estimate of the inverse covariance matrix and $|E(\hat{G}_{\lambda_n})|$ is the number of edges in the estimated graph. The estimate $\hat{\lambda}_n$ depends on a parameter $\gamma \in [0, 1]$ such that $\gamma = 0$ results in the BIC estimate and increasing γ produces sparser graphs. The authors in Reference [14] suggest that $\gamma = 0.5$ is a reasonable choice for high-dimensional problems. When solving subproblems using Algorithm 2, the $\log p$ term is replaced by $\log |\bar{R}|$, $\hat{\Theta}_{\lambda_n}$ is replaced by the inverse covariance over the vertices \bar{R} , and $|E(\hat{G}_{\lambda_n})|$ is replaced by the number of edges estimated from the graph H'_R .

Small subproblems: Whenever $|\bar{R}|$ is small (less than 8 in our simulations), we independently test whether each edge is in G^* using hypothesis testing. This shows the application of using different algorithms to learn different parts of the graph.

9.1 Results on Synthetic Graphs

We assume that $\Theta_{ii} = 1$ for all $i = 1, \dots, p$. We refer to all edges connected to the first p_1 vertices as *weak edges* and the rest of the edges are referred to as *strong edges*. The different types of synthetic graphical models we study are described as follows:

- Chain (CH₁ and CH₂): $\Theta_{i,i+1} = \rho_1$ for $i = 1, \dots, p_1 - 1$ (weak edges) and $\Theta_{i,i+1} = \rho_2$ for $i = p_1, p - 1$ (strong edges). For CH₁, $\rho_1 = 0.15$ and $\rho_2 = 0.245$. For CH₂, $\rho_1 = 0.075$ and $\rho_2 = 0.245$. Let $\Theta_{ij} = \Theta_{ji}$.
- Cycle (CY₁ and CY₂): $\Theta_{i,i+1} = \rho_1$ for $i = 1, \dots, p_1 - 1$ (weak edges) and $\Theta_{i,i+1} = \rho_2$ for $i = p_1, p - 1$ (strong edges). In addition, $\Theta_{i,i+3} = \rho_1$ for $i = 1, \dots, p_1 - 3$ and $\Theta_{i,i+3} = \rho_2$ for $i = p_1, p_1 + 1, \dots, p - 3$. This introduces multiple cycles in the graph. For CY₁, $\rho_1 = 0.15$ and $\rho_2 = 0.245$. For CY₂, $\rho_1 = 0.075$ and $\rho_2 = 0.245$.
- Hub (HB₁ and HB₂): For the first p_1 vertices, construct as many star⁶ graphs of size d_1 as possible. For the remaining vertices, construct star graphs of size d_2 (at most one may be of size less than d_2). The hub graph G^* is constructed by taking a union of all star graphs. For $(i, j) \in G^*$ s.t. $i, j \leq p_1$, let $\Theta_{i,j} = 1/d_1$. For the remaining edges, let $\Theta_{ij} = 1/d_2$. For HB₁, $d_1 = 8$ and $d_2 = 5$. For HB₂, $d_1 = 12$ and $d_2 = 5$.
- Neighborhood graph (NB₁ and NB₂): Randomly place vertices on the unit square at coordinates y_1, \dots, y_p . Let $\Theta_{ij} = 1/\rho_1$ with probability $(\sqrt{2\pi})^{-1} \exp(-4\|y_i - y_j\|_2^2)$, otherwise $\Theta_{ij} = 0$ for all $i, j \in \{1, \dots, p_1\}$ such that $i > j$. For all $i, j \in \{p_1 + 1, \dots, p\}$ such that $i > j$, $\Theta_{ij} = \rho_2$. For edges over the first p_1 vertices, delete edges so that each vertex is connected to at most d_1 other vertices. For the vertices $p_1 + 1, \dots, p$, delete edges such that the neighborhood of each vertex is at most d_2 . Finally, randomly add four edges from a vertex in $\{1, \dots, p_1\}$ to a vertex in $\{p_1, p_1 + 1, \dots, p\}$ such that for each such edge, $\Theta_{ij} = \rho_1$. We let $\rho_2 = 0.245$, $d_1 = 6$, and $d_2 = 4$. For NB₁, $\rho_1 = 0.15$ and for NB₂, $\rho_2 = 0.075$.

Notice that the parameters associated with the weak edges are lower than the parameters associated with the strong edges. Some comments regarding notation and usage of various algorithms is given as follows.

- The junction tree versions of the UGMS algorithms are denoted by JgL, JPC, and JnL.
- We use EBIC with $\gamma = 0.5$ to choose regularization parameters when estimating graphs using JgL and JPC. To objectively compare JgL (JPC) and gL (PC), we make sure that the number of edges estimated by gL (PC) is roughly the same as the number of edges estimated by JgL (JPC).

⁶A star is a tree where one vertex is connected all other vertices.

- The **nL** and **JnL** estimates are computed differently since it is difficult to control the number of edges estimated using both these algorithms⁷. We apply both **nL** and **JnL** for multiple different values of γ (the parameter for EBIC) and choose graphs so that the number of edges estimated is closest to the number of edges estimated by **gL**.
- When applying **PC** and **JPC**, we choose κ as 1, 2, 1, and 3 for Chain, Cycle, Hub, and Neighborhood graphs, respectively. When computing H , we choose κ as 0, 1, 0, and 2 for Chain, Cycle, Hub, and Neighborhood graphs, respectively.

Tables 2-5 summarize the results for the different types of synthetic graphical models. For an estimate \hat{G} of G^* , we evaluate \hat{G} using the weak edge discovery rate (WEDR), false discovery rate (FDR), true positive rate (TPR), and the edit distance (ED).

$$\text{WEDR} = \frac{\# \text{ weak edges in } \hat{G}}{\# \text{ of weak edges in } G^*} \quad (29)$$

$$\text{FDR} = \frac{\# \text{ of edges in } \hat{G} \setminus G^*}{\# \text{ of edges in } \hat{G}} \quad (30)$$

$$\text{TPR} = \frac{\# \text{ of edges in } \hat{G} \cap G^*}{\# \text{ of edges in } G^*} \quad (31)$$

$$\text{ED} = \{ \# \text{ edges in } \hat{G} \setminus G^* \} + \{ \# \text{ edges in } G^* \setminus \hat{G} \}, \quad (32)$$

Recall that the weak edges are over the first p_1 vertices in the graph. Naturally, we want WEDR and TPR to be large and FDR and ED to be small. Each entry in the table shows the mean value and standard error (in brackets) over 50 observations. We now make some remarks regarding the results.

Remark 9.1 (Graphical Lasso). Of all the algorithms, graphical Lasso (**gL**) performs the worst. On the other hand, junction tree based **gL** significantly improves the performance of **gL**. Moreover, the performance of **JgL** is comparable, and sometimes even better, when compared to **JPC** and **JnL**. This suggests that when using **gL** in practice, it is beneficial to apply a screening algorithm to remove some edges and then use the junction tree framework in conjunction with **gL**.

Remark 9.2 (PC-Algorithm and Neighborhood Selection). Although using junction trees in conjunction with the PC-Algorithm (**PC**) and neighborhood selection (**nL**) does improve the graph estimation performance, the difference is not as significant as **gL**. The reason is because both **PC** and **nL** make use of the local Markov property in the graph H . The junction tree framework further improves the performance of these algorithms by making use of the global Markov property, in addition to the local Markov property.

Remark 9.3 (Chain Graph). Although the chain graph does not satisfy the conditions in (A6), the junction tree estimates still outperforms the non-junction tree estimates. This suggests the advantages of using junction trees beyond the graphs considered in (A6). We suspect that correlation decay properties, which have been studied extensively in [1, 2], can be used to weaken the assumption in (A6).

Remark 9.4 (Hub Graph). For the hub graph HB_1 , the junction tree estimate does not result in a significant difference in performance, especially for the **PC** and **nL** algorithms. This is mainly because this graph is extremely sparse with multiple components. For the number of observations considered, H removes a significant number of edges. However, for HB_2 , the junction tree estimate, in general, performs slightly better. This is because the parameters associated with the weak edges in HB_2 are smaller than that of HB_1 .

Remark 9.5 (General Conclusion). We see that, in general, the WEDR and TPR are higher, while the FDR and ED are lower, for junction tree based algorithms. This clearly suggests that using junction trees results in more accurate graph estimation. Moreover, the higher WEDR suggest that the main differences between the two algorithms are over the weak edges, i.e., junction tree based algorithms are estimating more weak edges when compared to a non junction tree based algorithm.

⁷Recall that both these algorithms use different regularization parameters, so there may exist multiple different estimates with the same number of edges

Table 2: Results for Chain graphs: $p = 100$ and $p_1 = 20$

Model	n	Alg	WEDR	FDR	TPR	ED	$ \hat{G} $
CH ₁ $p = 100$	300	JgL	0.305 (0.00473)	0.048 (0.000972)	0.767 (0.00158)	27 (0.176)	79.8
		gL	0.18 (0.00426)	0.0611 (0.00113)	0.757 (0.00135)	29 (0.153)	79.8
		JPC	0.312 (0.00441)	0.0466 (0.000954)	0.775 (0.00135)	26 (0.162)	80.5
		PC	0.264 (0.00513)	0.0473 (0.00107)	0.781 (0.00138)	25.6 (0.169)	81.2
		JnL	0.306 (0.00477)	0.0723 (0.00109)	0.769 (0.00149)	28.8 (0.188)	82.1
		nL	0.271 (0.00446)	0.0729 (0.00126)	0.757 (0.00147)	30 (0.197)	80.9
CH ₂ $p = 100$	300	JgL	0.0516 (0.00199)	0.0672 (0.00121)	0.727 (0.00141)	32.2 (0.173)	77.3
		gL	0.00947 (0.00103)	0.0619 (0.000974)	0.733 (0.00146)	31.3 (0.162)	77.4
		JPC	0.0484 (0.00189)	0.0637 (0.00114)	0.735 (0.00136)	31.2 (0.169)	77.8
		PC	0.0337 (0.00183)	0.0545 (0.00124)	0.748 (0.00114)	29.3 (0.144)	78.4
		JnL	0.0516 (0.00204)	0.077 (0.00113)	0.733 (0.0014)	32.5 (0.186)	78.7
		nL	0.0389 (0.00201)	0.086 (0.00144)	0.723 (0.00143)	34.2 (0.216)	78.4
CH ₁ $p = 100$	500	JgL	0.596 (0.00551)	0.0206 (0.000597)	0.916 (0.00117)	10.2 (0.133)	92.6
		gL	0.44 (0.00516)	0.0502 (0.000773)	0.889 (0.00106)	15.6 (0.132)	92.7
		JPC	0.612 (0.00507)	0.0215 (0.000705)	0.921 (0.000976)	9.86 (0.128)	93.2
		PC	0.577 (0.0048)	0.0324 (0.000746)	0.916 (0.000956)	11.4 (0.124)	93.7
		JnL	0.623 (0.00483)	0.0588 (0.00092)	0.922 (0.000925)	13.5 (0.133)	97
		nL	0.596 (0.00474)	0.0689 (0.00112)	0.918 (0.000953)	14.9 (0.164)	97.6
CH ₂ $p = 100$	500	JgL	0.0768 (0.00257)	0.0435 (0.000974)	0.816 (0.000581)	22 (0.107)	84.5
		gL	0.0211 (0.00143)	0.0533 (0.000974)	0.808 (0.000347)	23.5 (0.0824)	84.6
		JPC	0.0726 (0.00228)	0.042 (0.000966)	0.817 (0.000532)	21.7 (0.0822)	84.5
		PC	0.0516 (0.00226)	0.0489 (0.00111)	0.815 (0.000504)	22.5 (0.0918)	84.9
		JnL	0.0758 (0.00243)	0.0702 (0.00109)	0.818 (0.000536)	24.2 (0.102)	87.2
		nL	0.0663 (0.00232)	0.0767 (0.00123)	0.815 (0.000555)	25.1 (0.126)	87.5

Table 3: Results for Cycle graphs, $p = 100$ and $p_1 = 20$

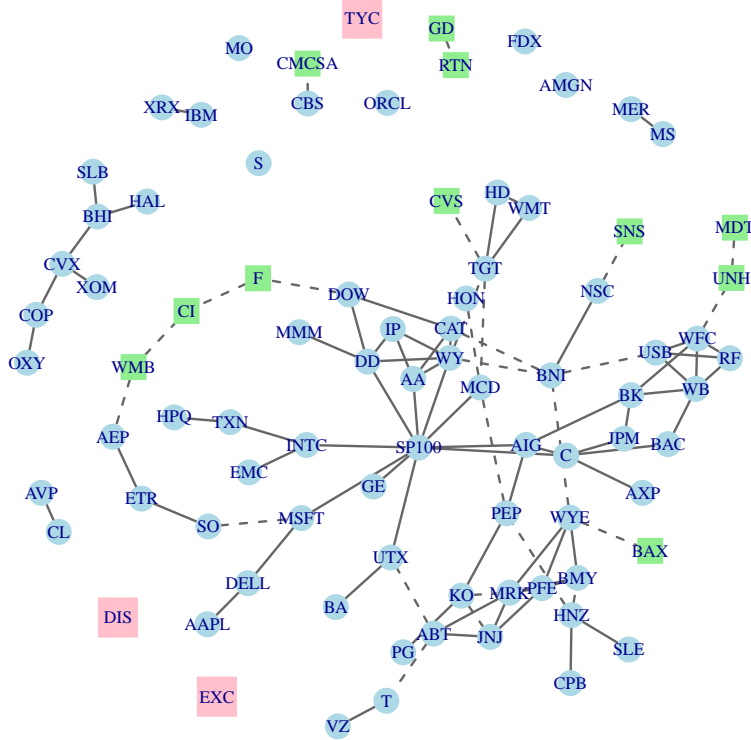
Model	n	Alg	WEDR	FDR	TPR	ED	$ \hat{G} $
CY ₁ $p = 100$	300	JgL	0.314 (0.00356)	0.0355 (0.000617)	0.814 (0.00102)	28.5 (0.142)	111
		gL	0.105 (0.00309)	0.0556 (0.000806)	0.798 (0.000995)	32.9 (0.16)	112
		JPC	0.326 (0.00401)	0.0302 (0.000661)	0.819 (0.00129)	27.2 (0.18)	112
		PC	0.307 (0.00427)	0.0266 (0.000707)	0.826 (0.00125)	26 (0.169)	112
		JnL	0.342 (0.00373)	0.0429 (0.000803)	0.813 (0.00113)	29.5 (0.175)	112
		nL	0.299 (0.00363)	0.0443 (0.000974)	0.793 (0.00131)	32.3 (0.192)	110
CY ₂ $p = 100$	300	JgL	0.0472 (0.00164)	0.0445 (0.000906)	0.762 (0.000961)	36.2 (0.163)	105
		gL	0.0008 (0.000253)	0.0488 (0.000775)	0.759 (0.00109)	37 (0.172)	105
		JPC	0.0432 (0.00176)	0.0424 (0.000877)	0.764 (0.000869)	35.6 (0.174)	105
		PC	0.0272 (0.00147)	0.0355 (0.000758)	0.773 (0.000767)	33.7 (0.137)	106
		JnL	0.042 (0.00209)	0.0575 (0.00108)	0.754 (0.00117)	38.6 (0.21)	106
		nL	0.035 (0.00241)	0.0569 (0.00118)	0.743 (0.00129)	39.9 (0.228)	104
CY ₁ $p = 100$	500	JgL	0.532 (0.0045)	0.0222 (0.000549)	0.907 (0.000933)	15.1 (0.139)	122
		gL	0.278 (0.00526)	0.0707 (0.000723)	0.862 (0.00102)	26.9 (0.178)	122
		JPC	0.61 (0.0042)	0.0157 (0.000575)	0.925 (0.000825)	11.9 (0.15)	124
		PC	0.609 (0.00398)	0.0203 (0.000547)	0.925 (0.000786)	12.5 (0.134)	125
		JnL	0.612 (0.00449)	0.028 (0.000605)	0.924 (0.000825)	13.6 (0.151)	125
		nL	0.584 (0.00449)	0.0406 (0.000726)	0.919 (0.000929)	15.9 (0.171)	126
CY ₂ $p = 100$	500	JgL	0.0864 (0.00271)	0.0389 (0.000766)	0.821 (0.000571)	28.1 (0.116)	113
		gL	0.004 (0.000542)	0.0578 (0.000768)	0.805 (0.000359)	32.3 (0.0883)	113
		JPC	0.0872 (0.00233)	0.0343 (0.000682)	0.825 (0.000467)	27 (0.0988)	113
		PC	0.0744 (0.00234)	0.0399 (0.000689)	0.823 (0.000497)	27.9 (0.0995)	113
		JnL	0.085 (0.00315)	0.0451 (0.00102)	0.824 (0.00061)	28.4 (0.147)	114
		nL	0.069 (0.00309)	0.053 (0.00103)	0.821 (0.000638)	29.8 (0.158)	114

Table 4: Results for Hub graphs: $p = 100$ and $p_1 = 20$

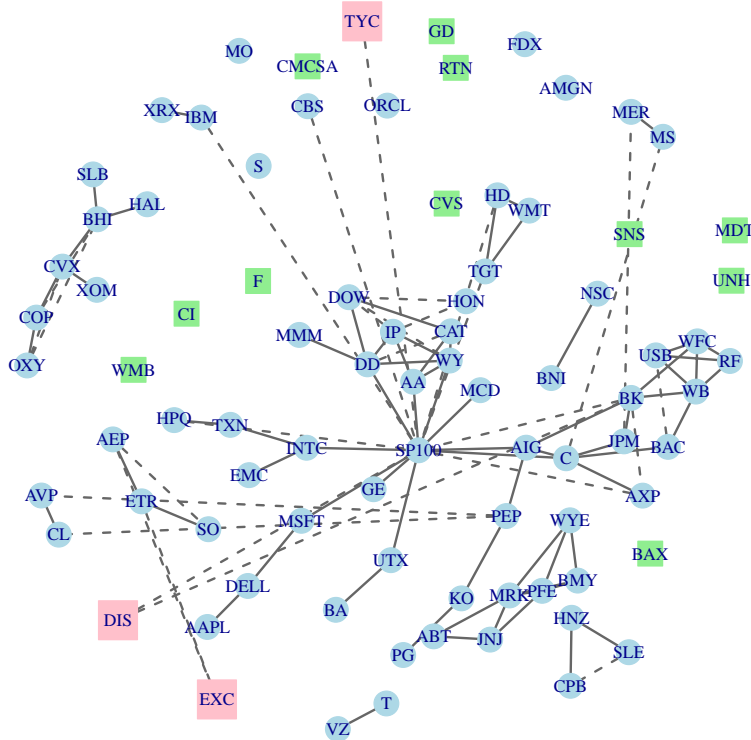
Model	n	Alg	WEDR	FDR	TPR	ED	$ \widehat{G} $
HB ₁ $p = 100$	300	JgL	0.204 (0.00398)	0.0389 (0.00107)	0.755 (0.00174)	22.3 (0.151)	63.7
		gL	0.154 (0.00408)	0.0377 (0.00107)	0.758 (0.00161)	22.1 (0.13)	63.8
		JPC	0.204 (0.00419)	0.038 (0.00106)	0.753 (0.00175)	22.4 (0.16)	63.4
		PC	0.193 (0.00422)	0.0377 (0.000953)	0.762 (0.00177)	21.7 (0.143)	64.2
		JnL	0.245 (0.00464)	0.0887 (0.00131)	0.75 (0.00178)	26.2 (0.174)	66.7
		nL	0.247 (0.00448)	0.0983 (0.00155)	0.752 (0.00182)	26.8 (0.198)	67.6
HB ₂ $p = 100$	300	JgL	0.0444 (0.00194)	0.0471 (0.00131)	0.71 (0.00134)	26.7 (0.116)	61.2
		gL	0.0133 (0.00118)	0.0425 (0.00127)	0.716 (0.00142)	26 (0.121)	61.4
		JPC	0.0478 (0.00225)	0.0431 (0.00114)	0.709 (0.00128)	26.5 (0.108)	60.8
		PC	0.0289 (0.00189)	0.0381 (0.00105)	0.718 (0.00141)	25.5 (0.121)	61.3
		JnL	0.0544 (0.00248)	0.0833 (0.00142)	0.704 (0.0012)	29.6 (0.146)	63
		nL	0.0467 (0.00226)	0.0958 (0.00138)	0.7 (0.00116)	30.7 (0.138)	63.5
HB ₁ $p = 100$	500	JgL	0.413 (0.00732)	0.0262 (0.000978)	0.87 (0.00159)	12.4 (0.156)	72.4
		gL	0.364 (0.00704)	0.0352 (0.000983)	0.863 (0.00149)	13.7 (0.144)	72.5
		JPC	0.438 (0.00677)	0.0269 (0.000842)	0.878 (0.00147)	11.9 (0.148)	73.1
		PC	0.448 (0.00678)	0.0268 (0.000834)	0.882 (0.00143)	11.6 (0.141)	73.4
		JnL	0.507 (0.00615)	0.0764 (0.00133)	0.89 (0.00131)	14.9 (0.152)	78.2
		nL	0.52 (0.00706)	0.0907 (0.00139)	0.893 (0.00156)	15.9 (0.191)	79.6
HB ₂ $p = 100$	500	JgL	0.0856 (0.00276)	0.0416 (0.00111)	0.794 (0.000676)	19.8 (0.0855)	68
		gL	0.05 (0.00252)	0.0474 (0.00123)	0.789 (0.000633)	20.6 (0.0978)	68
		JPC	0.0967 (0.00288)	0.0395 (0.0012)	0.798 (0.000732)	19.3 (0.109)	68.2
		PC	0.0867 (0.0028)	0.0436 (0.00129)	0.797 (0.000687)	19.7 (0.111)	68.4
		JnL	0.123 (0.00366)	0.0843 (0.00155)	0.804 (0.000816)	22.2 (0.15)	72.1
		nL	0.106 (0.00341)	0.105 (0.00147)	0.801 (0.000743)	24.1 (0.143)	73.4

Table 5: Results for Neighborhood graph, $p = 300$ and $p_1 = 30$

Model	n	Alg	WEDR	FDR	TPR	ED	$ \widehat{G} $
NB ₁ $p = 100$	300	JgL	0.251 (0.00153)	0.0303 (0.000284)	0.813 (0.00049)	126 (0.329)	498
		gL	0.102 (0.0015)	0.0389 (0.00029)	0.806 (0.000506)	135 (0.345)	498
		JPC	0.259 (0.00159)	0.0313 (0.000231)	0.814 (0.000402)	126 (0.26)	499
		PC	0.255 (0.00157)	0.036 (0.000276)	0.813 (0.000466)	129 (0.33)	501
		JnL	0.254 (0.00244)	0.0354 (0.000378)	0.812 (0.000635)	129 (0.461)	500
		nL	0.226 (0.00249)	0.0389 (0.000451)	0.804 (0.000648)	136 (0.458)	497
NB ₁ $p = 100$	300	JgL	0.00457 (0.000335)	0.0429 (0.000364)	0.784 (0.000508)	149 (0.385)	486
		gL	0.000286 (7.64e - 05)	0.0361 (0.000283)	0.79 (0.000422)	142 (0.259)	487
		JPC	0.00371 (0.000263)	0.0416 (0.00031)	0.784 (0.000493)	148 (0.376)	486
		PC	0.00286 (0.000244)	0.0479 (0.000281)	0.782 (0.000377)	153 (0.239)	488
		JnL	0.00457 (0.000335)	0.0463 (0.000329)	0.783 (0.000448)	151 (0.356)	488
		nL	0.00314 (0.000274)	0.0504 (0.000403)	0.775 (0.000456)	158 (0.374)	485
NB ₁ $p = 100$	500	JgL	0.449 (0.00183)	0.0179 (0.000212)	0.921 (0.000279)	57.1 (0.199)	557
		gL	0.319 (0.00217)	0.0349 (0.00025)	0.905 (0.000295)	75.8 (0.242)	557
		JPC	0.489 (0.00181)	0.0148 (0.000171)	0.925 (0.000276)	52.8 (0.189)	558
		PC	0.496 (0.00169)	0.0226 (0.000235)	0.92 (0.000271)	60.2 (0.214)	559
		JnL	0.508 (0.00307)	0.0274 (0.000328)	0.929 (0.000438)	57.9 (0.348)	567
		nL	0.494 (0.00326)	0.0332 (0.000435)	0.927 (0.000435)	62.3 (0.4)	570
NB ₂ $p = 100$	500	JgL	0.008 (0.00041)	0.0329 (0.000252)	0.87 (0.000216)	95 (0.206)	534
		gL	0.000286 (7.64e - 05)	0.0341 (0.000275)	0.869 (0.000204)	96 (0.214)	534
		JPC	0.00886 (0.000448)	0.032 (0.000273)	0.87 (0.000205)	94.2 (0.215)	534
		PC	0.00543 (0.00036)	0.0404 (0.00027)	0.865 (0.000202)	102 (0.207)	536
		JnL	0.00829 (0.000464)	0.038 (0.000283)	0.871 (0.000198)	97.3 (0.22)	538
		nL	0.00486 (0.00032)	0.043 (0.000334)	0.87 (0.000177)	101 (0.234)	540

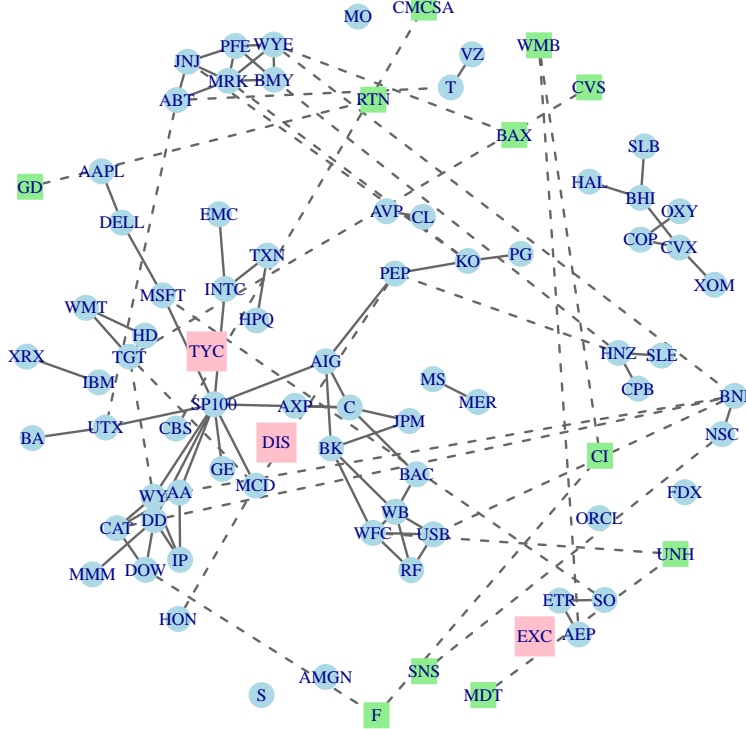


(a) Junction tree based graphical Lasso

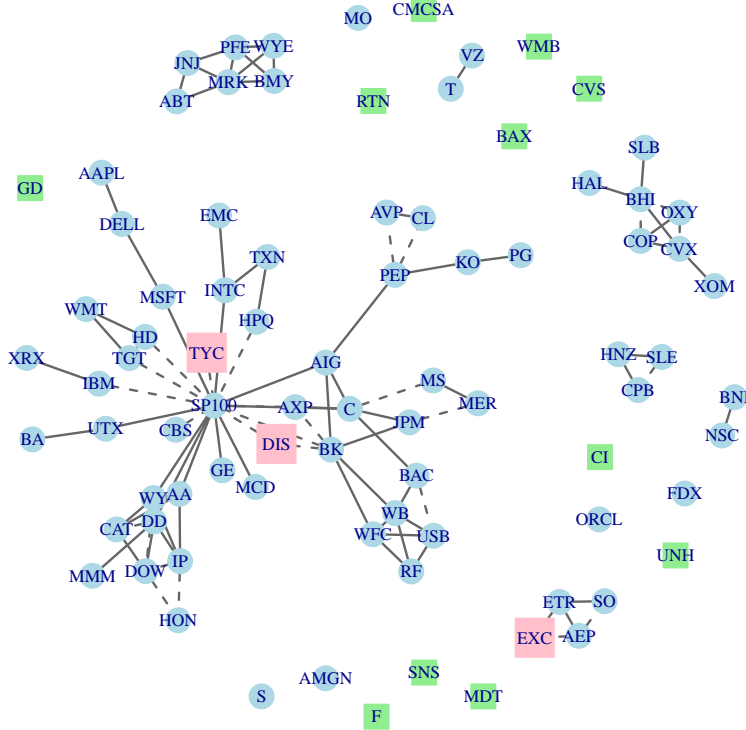


(b) Graphical Lasso

Figure 9: Graph over a subset of companies in the S&P 100. The positioning of the vertices is chosen so that the *junction tree based graph is aesthetically pleasing*. The edges common in (a) and (b) are marked by bold lines and the remaining edges are marked by dashed lines



(a) Junction tree based graphical Lasso



(b) Graphical Lasso

Figure 10: Graph over a subset of companies in the S&P 100. The positioning of the vertices is chosen so that the *graphical Lasso based graph is aesthetically pleasing*. The edges common in (a) and (b) are marked by bold lines and the remaining edges are marked by dashed lines.

9.2 Analysis of Stock Returns Data

We applied our methods to the data set in [12] of $n = 216$ monthly stock returns of $p = 85$ companies in the S&P 100. We computed H using $\kappa = 1$. We applied JgL using EBIC with $\gamma = 0.5$ and applied gL so that both graphs have the same number of edges. This allows us to objectively compare the gL and JgL graphs. Figure 9 shows the two estimated graphs in such a way that the vertices are positioned so that the JgL graph looks aesthetically pleasing. In Figure 10, the vertices are positioned so that gL looks aesthetically pleasing. In each graph, we mark the common edges by bold lines and the remaining edges by dashed lines. Some conclusions that we draw from the estimated graphs are summarized as follows:

- The gL graph in Figure 10(b) seems well structured with multiple different clusters of nodes with companies that seem to be related to each other. A similar clustering is seen for the JgL graph in Figure 9(a) with the exception that there are now connections between the clusters. As observed in [10, 12], it has been hypothesized that the “actual” graph over the companies is dense since there are several unobserved companies that induce conditional dependencies between the observed companies. These induced conditional dependencies can be considered to be the weak edges of the “actual” graph. Thus, our results suggest that the junction tree based algorithm is able to detect such weak edges.
- We now focus on some specific edges and nodes in the graphs. The 11 vertices represented by smaller squares and shaded in green are not connected to any other vertex in gL. On the other hand, all these 11 vertices are connected to at least one other vertex in JgL (see Figure 9). Moreover, several of these edges are meaningful. For example, CBS and CMCSA are in the television industry, TGT and CVS are stores, AEP and WMB are energy companies, GD and RTN are defense companies, and MDT and UNH are in the healthcare industry. Finally, the three vertices represented by larger squares and shaded in pink, are not connected to any vertex in JgL and are connected to at least one other vertex in gL. Only the edges associated with EXC seem to be meaningful.

9.3 Analysis of Gene Expression Data

Graphical models have been used extensively for studying gene interactions using gene expression data [35, 51]. The gene expression data we study is the Lymph node status data which contains $n = 148$ expression values from $p = 587$ genes [26]. Since there is no ground truth available, the main aim in this Section is to highlight the differences between the estimates JgL (junction tree estimate) and gL (non junction tree estimate). Just like in the stock returns data, we compute the graph H using $\kappa = 1$. Both the JgL and gL graphs contain 831 edges. Figure 11 shows the graphs JgL and gL under different placements of the vertices. We clearly see significant differences between the estimated graphs. This suggests that using the junction tree framework may lead to new scientific interpretations when studying biological data.

10. Summary and Future Work

We have outlined a general framework that can be used as a wrapper around any arbitrary undirected graphical model selection (UGMS) algorithm for improved graph estimation. Our framework takes as input a graph H that contains all (or most of) the edges in G^* , decomposes the UGMS problem into multiple subproblems using a junction tree representation of H , and then solves subproblems iteratively to estimate a graph. Our theoretical results show that certain weak edges, which cannot be estimated using standard algorithms, can be estimated when using the junction tree framework. We supported the theory with numerical simulations on both synthetic and real world data. All the data and code used in our numerical simulations can be found at <http://www.ima.umn.edu/~dvats/JunctionTreeUGMS.html>.

Our work motivates several interesting future research directions. In our framework, we used a graph H to decompose the UGMS problem into multiple subproblems. Alternatively, we can also focus on directly finding such decompositions. Another interesting research direction is to use the decompositions to develop parallel algorithms for UGMS for estimating extremely large graphs. Finally, motivated by the differences in the graphs obtained using gene expression data, another research problem of interest is to study the scientific consequences of using the junction tree framework on various computational biology data sets.

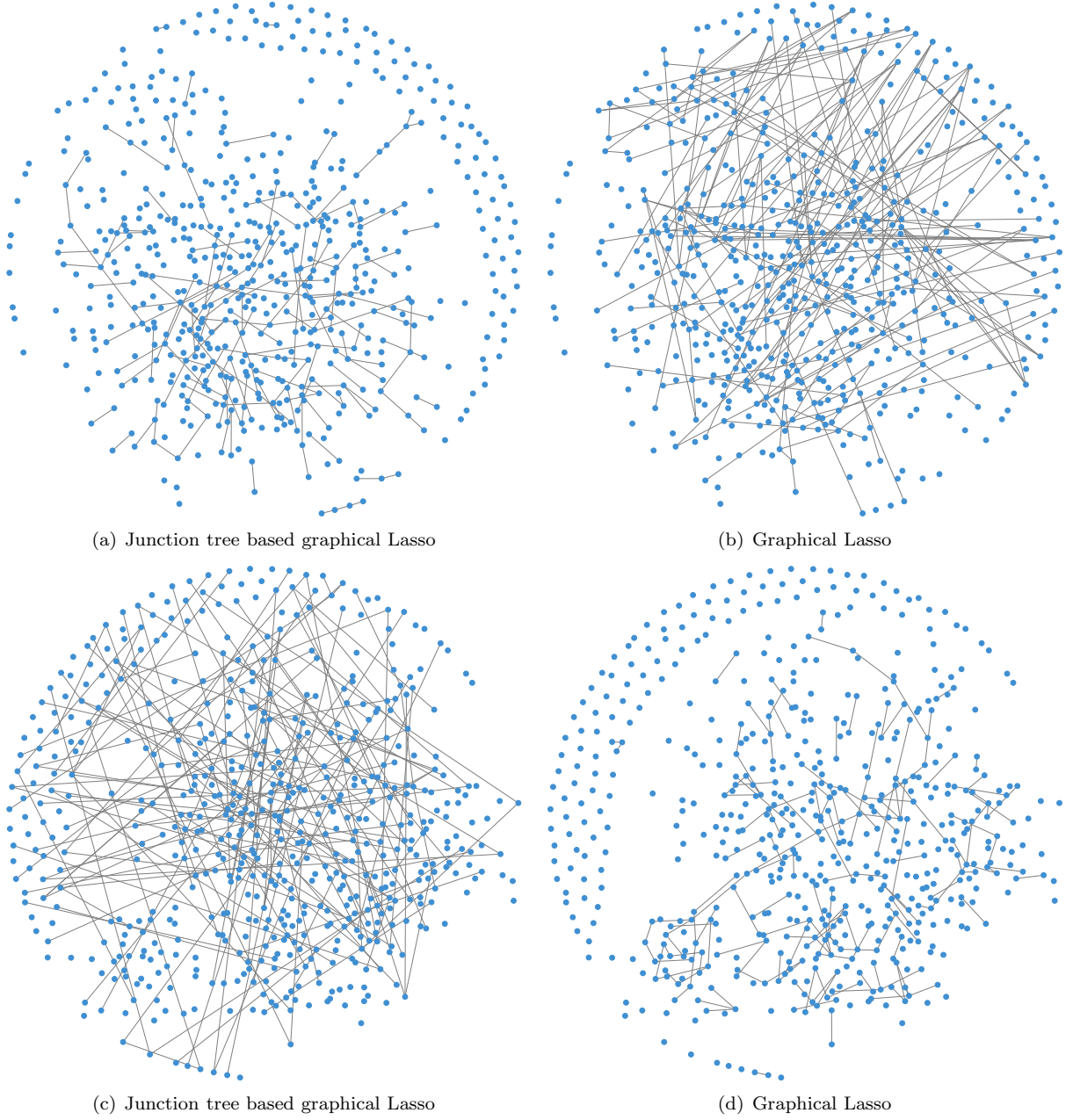


Figure 11: Graph over genes computed using gene expression data. For (a) and (b), the vertices are chosen so that the junction tree estimate is aesthetically pleasing. For (c) and (d), the vertices are chosen so that the graphical Lasso estimate is aesthetically pleasing. Further, in (a) and (c), we only show edges that are estimated in the junction tree estimate, but not estimated using graphical Lasso. Similarly, for (b) and (d), we only show edges that are estimated by graphical Lasso, but not by the junction tree estimate.

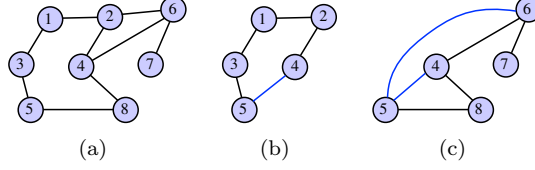


Figure 12: (a) A graph over eight nodes. (b) The marginal graph over $\{1, 2, 3, 4, 5\}$. (c) The marginal graph over $\{4, 5, 6, 7, 8\}$.

Acknowledgement

We thank Vincent Tan for discussions and comments on an earlier version of the paper. Divyanshu thanks the Institute for Mathematics and its Applications (IMA) for financial support in the form of a postdoctoral fellowship.

Appendix A. Marginal Graph

Definition A.1. The marginal graph $G^{*,m}[A]$ of a graph G^* over the nodes A is defined as a graph with the following properties

1. $E(G^*[A]) \subseteq E(G^{*,m}[A])$.
2. For an edge $(i, j) \in E(K_A) \setminus E(G^*[A])$, if all paths from i to j in G^* pass through a subset of the nodes in A , then $(i, j) \notin G^{*,m}[A]$.
3. For an edge $(i, j) \in E(K_A) \setminus E(G^*[A])$, if there exists a path from i to j in G^* such that all nodes in the path, except i and j , are in $V \setminus A$, then $(i, j) \in G^{*,m}[A]$.

The graph K_A is the complete graph over the vertices A . The first condition in Definition A.1 says that the marginal graph contains all edges in the induced subgraph over A . The second and third conditions say which edges not in $G^*[A]$ are in the marginal graph. As an example, consider the graph in Figure 12(a) and let $A = \{1, 2, 3, 4, 5\}$. From the second condition, the edge $(3, 4)$ is not in the marginal graph since all paths from 3 to 4 pass through a subset of the nodes in A . From the third condition, the edge $(4, 5)$ is in the marginal graph since there exists a path $\{4, 8, 5\}$ that does not go through any nodes in $A \setminus \{4, 5\}$. Similarly, the marginal graph over $A = \{4, 5, 6, 7, 8\}$ can be constructed as in Figure 12(c). The importance of marginal graphs is highlighted in the following proposition.

Proposition A.1. If $P_X > 0$ is Markov on $G^* = (V, E(G^*))$ and not Markov on any subgraph of G^* , then for any subset of vertices $A \subseteq V$, P_{X_A} is Markov on the marginal graph $G^{*,m}[A]$ and not Markov on any subgraph of $G^{*,m}[A]$.

Proof Suppose P_{X_A} is Markov on the graph \check{G}_A and not Markov on any subgraph of \check{G}_A . We will show that $\check{G}_A = G^{*,m}[A]$.

- If $(i, j) \in G$, then $X_i \not\perp\!\!\!\perp X_j | X_S$ for every $S \subseteq V \setminus \{i, j\}$. Thus, $G[A] \subset \check{G}_A$.
- For any edge $(i, j) \in K_A \setminus G[A]$, suppose that for every path from i to j contains at least one node from $A \setminus \{i, j\}$. Then, there exists a set of nodes $S \subseteq A \setminus \{i, j\}$ such that $X_i \perp\!\!\!\perp X_j | X_S$ and $(i, j) \notin \check{G}_A$.
- For any edge $(i, j) \in K_A \setminus G[A]$, suppose that there exists a path from i to j such that all nodes in the path, except i and j , are in $V \setminus A$. This means we cannot find a separator for i and j in the set A , so $(i, j) \in \check{G}_A$.

From the construction of \check{G}_A and Definition A.1, it is clear that $\check{G}_A = G^{*,m}[A]$. □

Using Proposition A.1, it is clear that if the UGMS algorithm Ψ in Assumption 1 is applied to a subset of vertices A , the output will be a consistent estimator of the marginal graph $G^{*,m}[A]$. Note that from Definition A.1, although the marginal graph contains all edges in $G^*[A]$, it may contain additional edges as well. Given only the marginal graph $G^{*,m}[A]$, it is not clear how to identify edges that are in $G^*[A]$. For example, suppose G^* is a graph over four nodes and let the graph be a single cycle. The marginal graph over any subset of three nodes is always the complete graph. Given the complete graph over three nodes, computing the induced subgraph over the three nodes is nontrivial.

Appendix B. Examples of UGMS Algorithms

We give examples of standard UGMS algorithms and show how they can be used to implement step 3 in Algorithm 2 when estimating edges in a region of a region graph. For simplicity, we review algorithms for UGMS when P_X is a Gaussian distribution with mean zero and covariance Σ^* . Such distributions are referred to as Gaussian graphical models. It is well known [41] that the inverse covariance matrix $\Theta^* = (\Sigma^*)^{-1}$, also known as precision matrix, is such that for all $i \neq j$, $\Theta_{ij}^* \neq 0$ if and only if $(i, j) \in E(G^*)$. In other words, the graph G^* can be estimated given an estimate of the covariance or inverse covariance matrix of X . We review two standard algorithms for estimating G^* : graphical Lasso and neighborhood selection using Lasso (nLasso).

B.1 Graphical Lasso (gLasso)

Define the empirical covariance matrix \hat{S}_A over a set of vertices $A \subset V$ as follows:

$$\hat{S}_A = \frac{1}{n} \sum_{k=1}^n X_A^{(k)} \left(X_A^{(k)} \right)^T. \quad (33)$$

Recall from Algorithm 2, we apply a UGMS algorithm \bar{R} to estimate edges in H'_R defined in (3). The graphical Lasso (gLasso) estimates \hat{E}_R by solving the following convex optimization problem:

$$\hat{\Theta} = \arg \min_{\Theta \succ 0, \Theta_{ij}=0 \forall (i,j) \notin H^m[\bar{R}]} \left\{ \log \det(\Theta) - \text{trace} \left(\hat{S}_{\bar{R}} \mathbf{x}_{\bar{R}}^n \right) - \lambda \sum_{(i,j) \in H'_R} \Theta_{ij} \right\} \quad (34)$$

$$\hat{E}_R = \{(i, j) \in H'_A : \hat{\Theta}_{ij} \neq 0\}. \quad (35)$$

The graph $H^m[\bar{R}]$ is the marginal graph over \bar{R} (see Appendix A). When $\bar{R} = V$, $H = K_V$, and $H'_A = K_V$, the above equations recover the standard gLasso estimator, which was first proposed in [5]. Equation (34) can be solved using algorithms in [5, 17, 40, 55]. Theoretical properties of the estimates $\hat{\Theta}$ and \hat{E}_R have been studied in [38]. Note that the regularization parameter in (34) controls the sparsity of \hat{E}_R . A larger λ corresponds to a sparser solution. Further, we only regularize the terms in Θ_{ij} corresponding to the edges that need to be estimated, i.e., the edges in H'_R . Finally, Equation (34) also accounts for the edges H by computing the marginal graph over \bar{R} . In general, $H^m[\bar{R}]$ can be replaced by any graph that is superset of $H^m[\bar{R}]$.

B.2 Neighborhood Selection (nLasso)

Using the local Markov property of undirected graphical models (see Definition 2.1), we know that if P_X is Markov on G^* , then $P(X_i | X_{V \setminus i}) = P(X_i | X_{ne_{G^*}(i)})$. This motivates an algorithm for estimating the neighborhood of each node and then combining all these estimates to estimate G^* . For Gaussian graphical models, this can be achieved by solving a Lasso problem [45] at each node [33]. Recall that we are interested in estimating all edges in H'_R by applying a UGMS algorithm to \bar{R} . The neighborhood selection using Lasso (nLasso) algorithm is given as follows:

$$H'' = K_{\bar{R} \setminus H^m[\bar{R}]} \quad (36)$$

$$\hat{\beta}^k = \arg \min_{\beta_i=0, i \in ne_{H''}(k) \cup k \cup V \setminus A} \left\{ \|\mathfrak{X}_k^n - \mathfrak{X}^n \beta\|_2^2 + \lambda \sum_{i \in ne_{H'_R}(k)} |\beta_i| \right\} \quad (37)$$

$$\widehat{ne}^k = \{i : \hat{\beta}_i^k \neq 0\} \quad (38)$$

$$\widehat{E}_R = \bigcup_{k \in \overline{R}} \{(k, i) : i \in \widehat{ne}^k\}. \quad (39)$$

Notice that in the above algorithm if i is estimated to be a neighbor of j , then we include the edge (i, j) even if j is not estimated to be a neighbor of i . This is called the union rule for combining neighborhood estimates. In our numerical simulations, we use the intersection rule to combine neighborhood estimates, i.e., (i, j) is estimated only if i is estimated to be a neighbor of j and j is estimated to be a neighbor of i . Theoretical analysis of nLasso has been carried out in [33, 50]. Note that, when estimating the neighbors of a node k , we only penalize the neighbors in H'_R . Further, we use prior knowledge about some of the edges by using the graph H in (37). References [7, 34, 37] extend the neighborhood selection based method to discrete valued graphical models.

Appendix C. Proof of Proposition 5.1

We first prove the following result.

Lemma C.1. *For any $(i, j) \in H'_R$, there either exists no non-direct path from i to j in H or all non-direct paths in H pass through a subset of \overline{R} .*

Proof We first show the result for $R \in \mathcal{R}_1$. This means that R is one of the clusters in the junction tree used to construct the region graph and $ch(R)$ is the set of all separators of cardinality greater than one connected to the cluster R in the junction tree. Subsequently, $\overline{R} = R$. If $ch(R) = \emptyset$, the claim trivially holds. Let $ch(R) \neq \emptyset$ and suppose there exists a non-direct path from i to j that passes through a vertex $k \notin R$. Then, there will exist a separator S in the junction tree such that S separates $\{i, j\}$ and k . Thus, all paths in H from i and j to k pass through S . This implies that either there is no non-direct path from i to j in H or else we have reached a contradiction.

Now, suppose $R \in \mathcal{R}_l$ for $l > 1$. The set $an(R)$ contains all the clusters in the junction tree than contain R . From the running intersection property of junction trees, all these clusters must form a subtree in the original junction tree. Merge \overline{R} into one cluster and find a new junction tree \mathcal{J}' by keeping the rest of the clusters the same. It is clear \overline{R} will be in the first row of the updated region graph. The arguments used above can be repeated to prove the claim. \square

We now prove Proposition 5.1.

Case 1: Let $(i, j) \in H'_R$ and $(i, j) \notin G^*$. If there exists no non-direct path from i to j in H , then the edge (i, j) can be estimated by solving a UGMS problem over i and j . By the definition of \overline{R} , $i, j \in \overline{R}$. Suppose there does exist non-direct paths from i to j in H . From Proposition C.1, all such paths pass through \overline{R} . Thus, the conditional independence of X_i and X_j can be determined from $X_{\overline{R} \setminus \{i, j\}}$.

Case 2: Let $(i, j) \in H'_R$ and $(i, j) \in G^*$. From Proposition C.1 and using the fact that $E(G^*) \subseteq E(H)$, we know that all paths from i to j pass through \overline{R} . This means that if $X_i \not\perp\!\!\!\perp X_j | X_{\overline{R} \setminus \{i, j\}}$, then $X_i \not\perp\!\!\!\perp X_j | X_{V \setminus \{i, j\}}$.

Appendix D. Analysis of the PC-Algorithm in Algorithm 4

In this Section, we present analysis of Algorithm 4 using results from [1] and [20]. The analysis presented here is for the non-junction tree based algorithm. Throughout this Section, assume

$$\widehat{G} = \text{PC}(\eta, \mathfrak{X}^n, K_V, K_V),$$

where K_V is the complete graph over the vertices V . Further, let the threshold for the conditional independence test in (6) be λ_n . We are interested in finding conditions under which $\widehat{G} = G^*$ with high probability.

Theorem D.1. *Under Assumptions (A1)-(A5), there exists a conditional independence test such that if*

$$n = \Omega(\rho_{\min}^{-2} \eta \log(p)), \text{ or} \quad (40)$$

$$\rho_{\min} = \Omega(\sqrt{\eta \log(p)/n}), \quad (41)$$

then $P(\widehat{G} \neq G) \rightarrow 0$ as $n \rightarrow \infty$.

Define the set B_η as follows

$$B_\eta = \{(i, j, S) : i, j \in V, i \neq j, S \subseteq V \setminus \{i, j\}, |S| \leq \eta\}. \quad (42)$$

The following concentration inequality follows from [1].

Lemma D.1. *Under Assumption (A4), there exists constants c_1 and c_2 such that for $\epsilon < M$,*

$$\sup_{(i,j,S) \in B_\eta} P(|\rho_{ij|S}| - |\widehat{\rho}_{ij|S}| > \xi) \leq c_1 \exp(-c_2(n-\eta)\xi^2), \quad (43)$$

where n is the number of vector valued measurements made of X_i, X_j , and X_S .

Proof See [1]. □

Let $P_e = P(\widehat{G} \neq G)$, where the probability measure P is with respect to P_X . Recall that we threshold the empirical conditional partial correlation $\widehat{\rho}_{ij|S}$ to test for conditional independence, i.e., $\widehat{\rho}_{ij|S} \leq \lambda_n \implies X_i \perp\!\!\!\perp X_j | X_S$. An error may occur if there exists two distinct vertices i and j such that either $\rho_{ij|S} = 0$ and $|\widehat{\rho}_{ij|S}| > \lambda_n$ or $|\rho_{ij|S}| > 0$ and $|\widehat{\rho}_{ij|S}| \leq \lambda_n$. Thus, we have

$$P_e \leq P(\mathcal{E}_1) + P(\mathcal{E}_2), \quad (44)$$

$$P(\mathcal{E}_1) = P\left(\bigcup_{(i,j) \notin G} \{\exists S \text{ s.t. } |\widehat{\rho}_{ij|S}| > \lambda_n\}\right) \quad (45)$$

$$P(\mathcal{E}_2) = P\left(\bigcup_{(i,j) \in G} \{\exists S \text{ s.t. } |\widehat{\rho}_{ij|S}| \leq \lambda_n\}\right). \quad (46)$$

We will find conditions under which $P(\mathcal{E}_1) \rightarrow 0$ and $P(\mathcal{E}_2) \rightarrow 0$ which will imply that $P_e \rightarrow 0$. The term $P(\mathcal{E}_1)$, probability of including an edge in \widehat{G} that does not belong to the true graph, can be upper bounded as follows:

$$P(\mathcal{E}_1) \leq P\left(\bigcup_{(i,j) \notin G} \{\exists S \text{ s.t. } |\widehat{\rho}_{ij|S}| > \lambda_n\}\right) \leq P\left(\bigcup_{(i,j) \notin G, S \subseteq V \setminus \{i,j\}} \{|\widehat{\rho}_{ij|S}| > \lambda_n\}\right) \quad (47)$$

$$\leq p^{\eta+2} \sup_{(i,j,S) \in B_\eta} P(|\widehat{\rho}_{ij|S}| > \lambda_n) \quad (48)$$

$$\leq c_1 p^{\eta+2} \exp(-c_2(n-\eta)\lambda_n^2) = c_1 \exp((\eta+2)\log(p) - c_2(n-\eta)\lambda_n^2) \quad (49)$$

The terms $p^{\eta+2}$ comes from the fact that there are at most p^2 number of edges and the algorithm searches over at most p^η number of separators for each edge. Choosing λ_n such that

$$\lim_{n,p \rightarrow \infty} \frac{(n-\eta)\lambda_n^2}{(\eta+2)\log(p)} = \infty \quad (50)$$

ensures that $P(\mathcal{E}_1) \rightarrow 0$ as $n, p \rightarrow \infty$. Further, choose λ_n such that for $c_3 < 1$

$$\lambda_n < c_3 \rho_{\min}. \quad (51)$$

The term $P(\mathcal{E}_2)$, probability of not including an edge in \hat{G} that does belong to the true graph, can be upper bounded as follows:

$$P(\mathcal{E}_2) \leq P\left(\bigcup_{(i,j) \in G} \{\exists S \text{ s.t. } |\hat{\rho}_{ij|S}| \leq \lambda_n\}\right) \leq P\left(\bigcup_{(i,j) \in G, S \subset V \setminus \{i,j\}} |\rho_{ij|S}| - |\hat{\rho}_{ij|S}| > |\rho_{ij|S}| - \lambda_n\right) \quad (52)$$

$$\leq p^{\eta+2} \sup_{(i,j,S) \in B_\eta} P(|\rho_{ij|S}| - |\hat{\rho}_{ij|S}| > |\rho_{ij|S}| - \lambda_n) \quad (53)$$

$$\leq p^{\eta+2} \sup_{(i,j,S) \in B_\eta} P(|\rho_{ij|S}| - |\hat{\rho}_{ij|S}| > \rho_{\min} - \lambda_n) \quad (54)$$

$$\leq c_1 p^{\eta+2} \exp(-c_2(n-\eta)(\rho_{\min} - \lambda_n)^2) = c_1 \exp((\eta+2)\log(p) - c_4(n-\eta)\rho_{\min}^2). \quad (55)$$

To get (55), we use (51) so that $(\rho_{\min} - \lambda_n) > (1 - c_3)\rho_{\min}$. For some constant $c_5 > 0$, suppose that for all $n > n'$ and $p > p'$,

$$c_4(n-\eta)\rho_{\min}^2 > (\eta+2+c_5)\log(p). \quad (56)$$

Given (56), $P(\mathcal{E}_2) \rightarrow 0$ as $n, p \rightarrow \infty$. In asymptotic notation, we can write (56) as

$$n = \Omega(\rho_{\min}^{-2} \eta \log(p)) \quad (57)$$

which proves the Theorem. The conditional independence test is such that λ_n is chosen to satisfy (50) and (51). In asymptotic notation, we can show that $\lambda_n = O(\rho_{\min})$ and $\lambda_n^2 = \Omega(\eta \log(p)/n)$ satisfies (50) and (51).

Appendix E. Proof of Theorem 8.1

To prove the theorem, it is sufficient to establish that

$$\rho_0 = \Omega\left(\sqrt{\eta_T \log(p)/n}\right) \quad (58)$$

$$\rho_1 = \Omega\left(\sqrt{\eta \log(p_1)/n}\right) \quad (59)$$

$$\rho_2 = \Omega\left(\sqrt{\eta \log(p_2)/n}\right) \quad (60)$$

$$\rho_T = \Omega\left(\sqrt{\eta \log(p_T)/n}\right). \quad (61)$$

Let H be the graph estimated in Step 1. An error occurs if for an edge $(i, j) \in G^*$ there exists a subset of vertices S such that $|S| \leq \eta_T$ and $\hat{\rho}_{ij|S} > \lambda_n^0$. Using the proof of Theorem D.1 (see analysis of $P(\mathcal{E}_2)$), it is easy to see that $n = \Omega(\rho_0^{-2} \eta_T \log(p))$ is sufficient for $P(E(G^*) \not\subset E(H)) \rightarrow 0$ as $n \rightarrow \infty$. Further, the threshold is chosen such that $\lambda_n^0 = O(\rho_0)$ and $(\lambda_n^0)^2 = \Omega(\eta_T \log(p)/n)$. This proves (58).

In Step 2, we estimate the graphs \hat{G}_1 and \hat{G}_2 by applying the PC-Algorithm to the vertices $V_1 \cup T$ and $V_2 \cup T$, respectively. For \hat{G}_1 , given that all edges that have a separator of size η_T have been removed, we can again use the analysis in the proof of Theorem D.1 to show that for $\lambda_n^1 = O(\rho_1)$ and $(\lambda_n^1)^2 = \Omega(\eta \log(p_1)/n)$, $n = \Omega(\rho_1^{-2} \eta \log(p_1))$ is sufficient for $P(\hat{G}_1 \neq G^*[V_1 \cup T] \setminus K_T | G^* \subset H) \rightarrow 0$ as $n \rightarrow \infty$. This proves (59). Using similar analysis, we can prove (60) and (61).

The probability of error can be written as

$$\begin{aligned} P_e &\leq P(G^* \not\subset H) + \sum_{k=1}^2 P(\hat{G}_k \neq G^*[V_k \cup T] \setminus K_T | G^* \subset H) \\ &\quad + P(\hat{G}_T \neq G^*[T] | G^* \subset H, \hat{G} = G[V_1 \cup T] \setminus K_T, G^*[V_2 \cup T] = G[V_2 \cup T] \setminus K_T). \end{aligned} \quad (62)$$

Given (58)-(61), each term on the right goes to 0 as $n \rightarrow \infty$, so $P_e \rightarrow 0$ as $n \rightarrow \infty$.

References

- [1] A. Anandkumar, V. Y. F. Tan, F. Huang, and A. S. Willsky. High-dimensional Gaussian graphical model selection: Walk summability and local separation criterion. *Journal of Machine Learning Research*, 13:2293–2337, 2012.
- [2] A. Anandkumar, V. Y. F. Tan, F. Huang, and A. S. Willsky. High-dimensional structure learning of ising models: Local separation criterion. *Annals of Statistics*, 40(3):1346–1375, 2012.
- [3] S. Arnborg, D. Corneil, and A. Proskurowski. Complexity of finding embeddings in ak-tree. *SIAM Journal on Algebraic Discrete Methods*, 8(2):277–284, 1987.
- [4] S. Arnborg and A. Proskurowski. Linear time algorithms for NP-hard problems restricted to partial k-trees. *Discrete Applied Mathematics*, 23(1):11–24, April 1989.
- [5] O. Banerjee, L. El Ghaoui, and A. d’Aspremont. Model selection through sparse maximum likelihood estimation for multivariate Gaussian or binary data. *J. Mach. Learn. Res.*, 9:485–516, 2008.
- [6] A. Berry, P. Heggernes, and G. Simonet. The minimum degree heuristic and the minimal triangulation process. In *Graph-Theoretic Concepts in Computer Science*, pages 58–70. Springer, 2003.
- [7] G. Bresler, E. Mossel, and A. Sly. Reconstruction of Markov random fields from samples: Some observations and algorithms. In A. Goel, K. Jansen, J. Rolim, and R. Rubinfeld, editors, *Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques*, volume 5171 of *Lecture Notes in Computer Science*, pages 343–356. Springer Berlin, 2008.
- [8] F. Bromberg, D. Margaritis, and V. Honavar. Efficient markov network structure discovery using independence tests. *J. Artif. Intell. Res. (JAIR)*, 35:449–484, 2009.
- [9] T. Cai, W. Liu, and X. Luo. A constrained ℓ_1 minimization approach to sparse precision matrix estimation. *Journal of the American Statistical Association*, 106(494):594–607, 2011.
- [10] V. Chandrasekaran, P. Parrilo, and A. Willsky. Latent variable graphical model selection via convex optimization. *Ann. Statist.*, 40(4):1935–1967, 2012.
- [11] J. Chen and Z. Chen. Extended bayesian information criteria for model selection with large model spaces. *Biometrika*, 95(3):759–771, 2008.
- [12] M. J. Choi, V. Y. F. Tan, A. Anandkumar, and A. S. Willsky. Learning latent tree graphical models. *Journal of Machine Learning Research*, 2011.
- [13] J. Fan and J. Lv. Sure independence screening for ultrahigh dimensional feature space. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 70(5):849–911, 2008.
- [14] R. Foygel and M. Drton. Extended bayesian information criteria for Gaussian graphical models. *NIPS*, 2010.
- [15] J. Friedman, T. Hastie, and R. Tibshirani. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9(3):432–441, July 2008.
- [16] W. Hoeffding. A non-parametric test of independence. *The Annals of Mathematical Statistics*, 19(4):546–557, 1948.
- [17] C.-J. Hsieh, M. A. Sustik, I. S. Dhillon, and P. Ravikumar. Sparse inverse covariance matrix estimation using quadratic approximation. In J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pages 2330–2338. 2011.
- [18] A. Jalali, C. Johnson, and P. Ravikumar. On learning discrete graphical models using greedy methods. In *NIPS*, 2011.

- [19] A. Jalali, C. Johnson, and P. Ravikumar. High-dimensional sparse inverse covariance estimation using greedy methods. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2012.
- [20] M. Kalisch and P. Bühlmann. Estimating high-dimensional directed acyclic graphs with the PC algorithm. *Journal of Machine Learning Research*, 8:613–636, 2007.
- [21] U. B. Kjaerulff. Triangulation of graphs - algorithms giving small total state space. Technical Report Research Report R-90-09, Department of Mathematics and Computer Science, Aalborg University, Denmark, 1990.
- [22] D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. The MIT Press, 2009.
- [23] J. Lafferty, H. Liu, and L. Wasserman. Sparse nonparametric graphical models. *Statistical Science*, 27(4):519–537, 2012.
- [24] S. L. Lauritzen and D. J. Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society. Series B (Methodological)*, 50(2):157–224, 1988.
- [25] S. L. Lauritzen. *Graphical Models*. Oxford University Press, USA, 1996.
- [26] L. Li and K. Toh. An inexact interior point method for l_1 -regularized sparse covariance selection. *Mathematical Programming Computation*, 2(3):291–315, 2010.
- [27] H. Liu, F. Han, M. Yuan, J. Lafferty, and L. Wasserman. High dimensional semiparametric Gaussian copula graphical models. *Annals of Statistics*, 40(4):2293–2326, 2012.
- [28] H. Liu, K. Roeder, and L. Wasserman. Stability approach to regularization selection (stars) for high dimensional graphical models. *NIPS*, 2010.
- [29] H. Liu, F. Han, and C.-H. Zhang. Transelliptical graphical models. In P. Bartlett, F. Pereira, C. Burges, L. Bottou, and K. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 809–817. 2012.
- [30] H. Liu, J. Lafferty, and L. Wasserman. The nonparanormal: Semiparametric estimation of high dimensional undirected graphs. *The Journal of Machine Learning Research*, 10:2295–2328, 2009.
- [31] R. Mazumder and T. Hastie. Exact covariance thresholding into connected components for large-scale graphical lasso. *J. Mach. Learn. Res.*, 13:781–794, March 2012.
- [32] N. Meinshausen and P. Bühlmann. Stability selection. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 72(4):417–473, 2010.
- [33] N. Meinshausen and P. Bühlmann. High-dimensional graphs and variable selection with the lasso. *Ann. Statist.*, 34(3):1436–1462, 2006.
- [34] P. Netrapalli, S. Banerjee, S. Sanghavi, and S. Shakkottai. Greedy learning of markov network structure. In *Allerton Conf. on Communication, Control and Computing*, Sept. 2010.
- [35] J. R. Nevins, M. West, A. Dobra, A. Dobra, C. Hans, C. Hans, B. Jones, B. Jones, J. R. Nevins, and M. W. Abstract. Sparse graphical models for exploring gene expression data. *Journal of Multivariate Analysis*, 90:196–212, 2004.
- [36] M. Rasch, A. Gretton, Y. Murayama, W. Maass, N. Logothetis, L. Wiskott, G. Kempermann, L. Wiskott, G. Kempermann, B. Schölkopf, et al. A kernel two-sample test. *Journal of Machine Learning Research*, 2:299, 2012.
- [37] P. Ravikumar, M. J. Wainwright, and J. Lafferty. High-dimensional Ising model selection using l_1 -regularized logistic regression. *Annals of Statistics*, 38(3):1287–1319, 2010.

- [38] P. Ravikumar, M. Wainwright, G. Raskutti, and B. Yu. High-dimensional covariance estimation by minimizing ℓ_1 -penalized log-determinant divergence. *Electronic Journal of Statistics*, 5:935–980, 2011.
- [39] N. Robertson and P. D. Seymour. Graph minors. II. Algorithmic aspects of tree-width. *Journal of Algorithms*, 7(3):309 – 322, 1986.
- [40] K. Scheinberg, S. Ma, and D. Goldfarb. Sparse inverse covariance selection via alternating linearization methods. In J. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems 23*, pages 2101–2109. 2010.
- [41] T. P. Speed and H. T. Kiiveri. Gaussian Markov distributions over finite graphs. *The Annals of Statistics*, 14(1):138–150, 1986.
- [42] T. Speed and H. Kiiveri. Gaussian Markov distributions over finite graphs. *The Annals of Statistics*, 14(1):138–150, 1986.
- [43] P. Spirtes and C. Glymour. An algorithm for fast recovery of sparse causal graphs. *Social Science Computer Review*, 9:62–72, 1991.
- [44] P. Spirtes, C. Glymour, and R. Scheines. Causality from probability. In *Advanced Computing for the Social Sciences*, 1990.
- [45] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, Series B*, 58(1):267–288, 1996.
- [46] S. van de Geer, P. Bühlmann, and S. Zhou. The adaptive and the thresholded lasso for potentially misspecified models (and a lower bound for the lasso). *Electronic Journal of Statistics*, 5:688–749, 2011.
- [47] D. Vats. High-dimensional screening using multiple grouping of variables. *arXiv preprint arXiv:1208.2043*, 2013.
- [48] D. Vats and J. M. F. Moura. Finding non-overlapping clusters for generalized inference over graphical models. *IEEE Transactions on Signal Processing*, 60(12):6368 –6381, Dec. 2012.
- [49] M. J. Wainwright. *Stochastic Processes on Graphs: Geometric and Variational Approaches*. PhD thesis, Department of EECS, Massachusetts Institute of Technology, 2002.
- [50] M. J. Wainwright. Sharp thresholds for high-dimensional and noisy sparsity recovery using ℓ_1 -constrained quadratic programming (Lasso). *IEEE Trans. Inf. Theory*, 55(5):2183–2202, 2009.
- [51] A. Wille, P. Zimmermann, E. Vranová, A. Fürholz, O. Laule, S. Bleuler, L. Hennig, A. Prelic, P. Von Rohr, L. Thiele, et al. Sparse graphical Gaussian modeling of the isoprenoid gene network in arabidopsis thaliana. *Genome Biol*, 5(11):R92, 2004.
- [52] D. Witten, J. Friedman, and N. Simon. New insights and faster computations for the graphical lasso. *Journal of Computational and Graphical Statistics*, 20(4):892–900, 2011.
- [53] X. Xie and Z. Geng. A recursive method for structural learning of directed acyclic graphs. *Journal of Machine Learning Research*, 9:459–483, 2008.
- [54] J. Yedidia, W. Freeman, and Y. Weiss. Constructing free-energy approximations and generalized belief propagation algorithms. *IEEE Transactions on Information Theory*, 51(7):2282–2312, 2005.
- [55] M. Yuan and Y. Lin. Model selection and estimation in the Gaussian graphical model. *Biometrika*, 94(1):19–35, 2007.
- [56] K. Zhang, J. Peters, D. Janzing, and B. Schölkopf. Kernel-based conditional independence test and application in causal discovery. *Arxiv preprint arXiv:1202.3775*, 2012.
- [57] H. Zou. The adaptive lasso and its oracle properties. *Journal of the American Statistical Association*, 101(476):1418–1429, 2006.