Journal of Machine Learning Research 15 (2014) 147-191 Submitted 4/13; Revised 9/13; Published 1/14

# A Junction Tree Framework for Undirected Graphical Model Selection

**Divyanshu Vats** DVATS@RICE.EDU
*Department of Electrical and Computer Engineering*
*Rice University*
*Houston, TX 77005, USA*

**Robert D. Nowak** NOWAK@ECE.WISC.EDU
*Department of Electrical and Computer Engineering*
*University of Wisconsin–Madison*
*Madison, WI 53706, USA*

**Editor:** Sebastian Nowozin

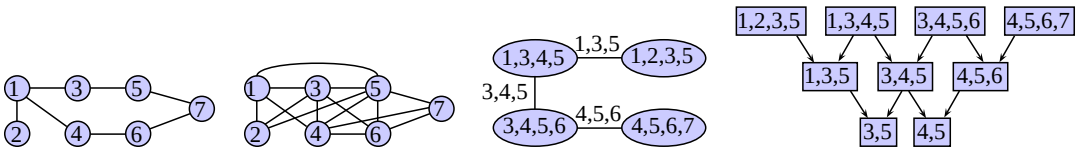
## Abstract

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 among 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 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 $V = \{1, \ldots, p\}$ 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 among the random variables in $X$ (Lauritzen, 1996). We study undirected graphical models (also known as Markov random fields) so that the graph $G^*$ is undirected, that is, if an edge $(i, j) \in E(G^*)$,




(a) Graph $G^*$ (b) Graph $H$ (c) Junction tree (d) Region graph

Figure 1: Our framework for estimating the graph in (a) using (b) 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.

then $(j, i) \in E(G^*)$. The undirected graphical model selection (UGMS) problem is to estimate $G^*$ given $n$ observations $\mathfrak{X}^n = \left(X^{(1)}, \ldots, X^{(n)}\right)$ drawn from $P_X$. This problem is of interest in many areas including biological data analysis, financial analysis, and social network analysis; see Koller and Friedman (2009) 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 in Fan and Lv (2008) or in Vats (to appear), 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.

### 1.1 Overview of the Junction Tree Framework

A junction tree is a tree-structured representation of an arbitrary graph (Robertson and Seymour, 1986). 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 graph related problems (Arnborg and Proskurowski, 1989). 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

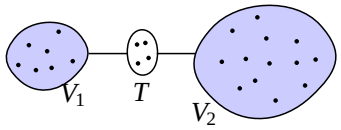



Figure 2: Structure of the graph used to analyze the junction tree framework for UGMS.

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 separators.[1] 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 $(2, 3)$ and $(2, 5)$.

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.

As illustrated above, our framework depends on a junction tree representation of the graph $H$ that contains a superset of the true edges. Given any graph, there may exist several junction tree representations. An optimal junction tree is a junction tree representation such that the maximum size of the cluster is as small as possible. Since we apply UGMS algorithms to the clusters of the junction tree, and the complexity of UGMS depends on the number of vertices in the graph, it is useful to apply our framework using optimal junction trees. Unfortunately, it is computationally intractable to find optimal junction trees (Arnborg et al., 1987). However, there exists several computationally efficient greedy heuristics that compute close to optimal junction trees (Kjaerulff, 1990; Berry et al., 2003). We use such heuristics to find junction trees when implementing our algorithms in practice.

### 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 weaker conditions than required by previously proposed methods. For example, we show that if

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

$G^*$ 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 known to be 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, may be estimated when using the junction tree framework. We show this theoretically and using 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 Spirtes and Glymour (1991), Kalisch and Bühlmann (2007), Banerjee et al. (2008), Friedman et al. (2008), Meinshausen and Bühlmann (2006), Anandkumar et al. (2012a) and Cai et al. (2011) for learning Gaussian graphical models, references Liu et al. (2009), Xue and Zou (2012), Liu et al. (2012a), Lafferty et al. (2012) and Liu et al. (2012b) for learning non-Gaussian graphical models, and references Bresler et al. (2008), Bromberg et al. (2009), Ravikumar et al. (2010), Netrapalli et al. (2010), Anandkumar et al. (2012b), Jalali et al. (2011), Johnson et al. (2012) and Yang et al. (2012) 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 1) when learning graphs. This allows for improved graph estimation, as illustrated by both our theoretical and numerical results. 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 *exact* probabilistic inference in graphical models (Lauritzen and Spiegelhalter, 1988). In particular, given a graphical model, and its junction tree representation, the computational complexity of exact inference is exponential in the size of the cluster in the junction tree with the most of number of vertices. This has motivated a line of research for learning *thin junction trees* so that the maximum size of the cluster in the estimated junction tree is small so that inference is computationally tractable (Chow and Liu, 1968; Bach and Jordan, 2001; Karger and Srebro, 2001; Chechetka and Guestrin, 2007; Kumar and Bach, 2013). We also make note of algorithms for learning decomposable graphical models where the graph structure is assumed to triangulated (Malvestuto, 1991; Giudici and Green, 1999). In general, the goal in the above algorithms is to learn a joint probability distribution that approximates a more complex probability distribution so that computations, such as inference, can be done in a tractable manner. On the other hand, this paper considers the problem of learning the *structure of*

*the graph* that best represents the conditional dependencies among the random variables under consideration.

There are two notable algorithms in the literature that use junction trees for learning graphical models. The first is an algorithm presented in Xie and Geng (2008) that uses junction trees to find the direction of edges for learning *directed* graphical models. Unfortunately, this algorithm cannot be used for UGMS. The second is an algorithm presented in Ma et al. (2008) for learning chain graphs, that are graphs with both directed and undirected edges. The algorithm in Ma et al. (2008) uses a junction tree representation to learn an undirected graph before orienting some of the edges to learn a chain graph. Our proposed algorithm, and subsequent analysis, differs from the work in Ma et al. (2008) in the following ways:

(i) Our algorithm identifies an ordering on the edges, which subsequently results in a lower sample complexity and the possibility of learning weak edges in a graph. The ordering on the edges is possible because of our novel region graph interpretation for learning graphical models. For example, when learning the graph in Figure 1(a) using Figure 1(b), the algorithm in Ma et al. (2008) learns the edge $(3, 5)$ by applying a UGMS algorithm to the vertices $\{1, 2, 3, 4, 5, 6\}$. In contrast, our proposed algorithm first estimates all edges in the second layer of the region graph in Figure 1(d), re-estimates the region graph, and then only applies a UGMS algorithm to $\{3, 4, 5\}$ to determine if the edge $(3, 4)$ belongs to the graph. In this way, our algorithm, in general, requires applying a UGMS algorithm to a smaller number of vertices when learning edges over separators in a junction tree representation.

(ii) Our algorithm for using junction trees for UGMS is independent of the choice of the UGMS algorithm, while the algorithm presented in Ma et al. (2008) uses conditional independence tests for UGMS.

(iii) Our algorithm, as discussed in (i), has the additional advantage of learning certain weak edges that may not be estimated when using standard UGMS algorithms. We theoretically quantify this property of our algorithm, while no such theory was presented in Ma et al. (2008).

Recent work has shown that solutions to the graphical lasso (gLasso) (Friedman et al., 2008) 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 (Witten et al., 2011; Mazumder and Hastie, 2012). The methods in Witten et al. (2011) and Mazumder and Hastie (2012) 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 Witten et al. (2011) and Mazumder and Hastie (2012) to solve gLasso when performing model selection for choosing suitable regularization parameters. Finally, we note that recent work in Loh and Wainwright (2012) uses properties of junction trees to learn discrete graphical models. The algorithm in Loh and Wainwright (2012) is designed for learning discrete graphical models and our methods can be used to improve its performance.

### 1.4 Paper Organization

The rest of the paper is organized as follows:

- Section 2 reviews graphical models and formulates the undirected graphical model selection (UGMS) problem.
- Section 3 shows how junction trees can be represented as region graphs and outlines an algorithm for constructing a region graph from a junction tree.
- Section 4 shows how the region graphs can be used to apply a UGMS algorithm to the clusters and separators of a junction tree.
- Section 5 presents our main framework for using junction trees for UGMS. In particular, we show how the methods in Sections 3-4 can be used iteratively to estimate a graph.
- Section 6 reviews the PC-Algorithm, which we use to study the theoretical properties of the junction tree framework.
- Section 7 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 8 presents numerical simulations to highlight the advantages of using junction trees for UGMS in practice.
- Section 9 summarizes the paper and outlines some future work.

## 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 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 $\{i, s_1, \ldots, s_d, j\}$: A sequence of nodes such that $(i, s_1), (s_d, j), (s_k, s_{k+1}) \in E$ for $k = 1, \ldots, d-1$.

- 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 $G_2 = (V_2, E(G_2))$, we define the following standard operations:

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

### 2.2 Undirected Graphical Models

**Definition 1 (Undirected Graphical Model, Lauritzen, 1996)** *An undirected graphical model is a probability distribution $P_X$ defined on a graph $G^* = (V, E(G^*))$, where $V = \{1, \ldots, p\}$ indexes the random vector $X = (X_1, \ldots, X_p)$ 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, that is, $P_X > 0$, the Markov property in Definition 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 Section (UGMS)

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

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

We give examples of $\Psi$ in Appendix B. Assumption 1 also takes into account the high-dimensional case where $p$ depends on $n$ in such a way that $p, n \to \infty$.

### 2.4 Junction Trees

Junction trees (Robertson and Seymour, 1986) are used extensively for efficiently solving various graph related problems, see Arnborg and Proskurowski (1989) for some examples. Reference Lauritzen and Spiegelhalter (1988) 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 3 (Junction tree)** *For an undirected graph $G = (V, E(G))$, a junction tree $\mathcal{J} = (\mathcal{C}, E(\mathcal{J}))$ is a tree-structured graph over clusters of nodes in $V$ such that*

*(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 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 be formed. For example, 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 (Wainwright, 2002). 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 4 (Robertson and Seymour, 1986)** *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 $C_u \backslash S_{uv}$ and $C_v \backslash S_{uv}$.*

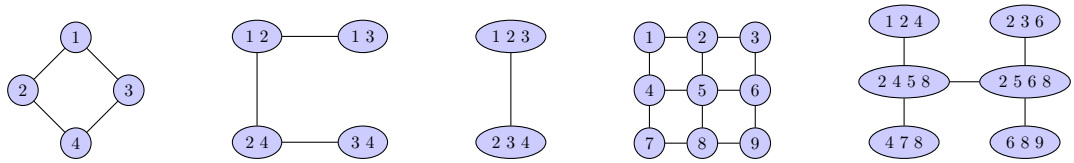



Figure 3: (a) An undirected graph, (b) Invalid junction tree since $\{1, 2\}$ separates $\{1, 3\}$ and $\{3, 4\}$,but $3 \notin \{1, 2\}$. (c) Valid junction tree for the graph in (a). (d) A grid graph. (e) Junction tree representation of (d).

The set of nodes $S_{uv}$ on the edges are called the *separators* of the junction tree. Proposition 4 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 4 allow the decomposition of UGMS into multiple subproblems over clusters and subsets of the separators in a junction tree.

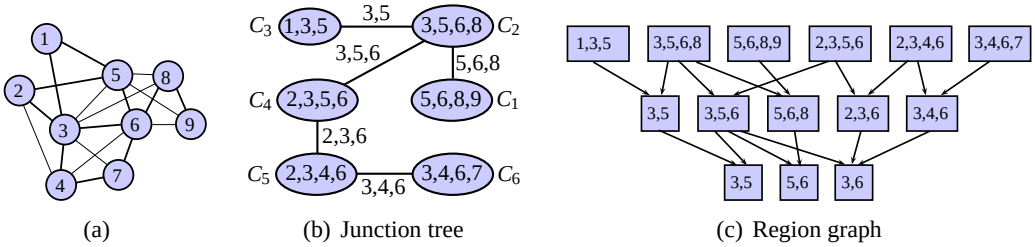
## 3. Overview of Region Graphs

In this section, we show how junction trees can be represented as region graphs. As we will see in Section 5, 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 Yedidia et al. (2005) 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 standard region graphs defined in the literature are used for computations over graphical models.

A *region* is a collection of nodes, which in this paper 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 $(R, S) \in \vec{E}(\mathcal{G})$, then 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 a graph $H$. We associate a label $l$ with every region in $\mathcal{R}$ and group




(a) (b) Junction tree (c) Region graph

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.

**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 $\mathcal{J}$, that is, $\mathcal{R}^2 = \{S_{uv} = C_u \cap C_v : (C_u, C_v) \in E(\mathcal{J})\}$.

3 To construct $\mathcal{R}^3$, 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 $\mathcal{R}^4, \ldots, \mathcal{R}^L$ until there are no more intersecting regions of cardinality greater than one.

5 For $R \in \mathcal{R}^\ell$ and $S \in \mathcal{R}^{\ell+1}$, add the edge $(R, S)$ to $\vec{E}(\mathcal{G})$ if $S \subseteq R$.

6 Let $\mathcal{R} = \{\mathcal{R}^1, \ldots, \mathcal{R}^L\}$.

regions with the same label to partition $\mathcal{R}$ into $L$ groups $\mathcal{R}^1, \ldots, \mathcal{R}^L$. 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 $\mathcal{R}^3, \ldots, \mathcal{R}^L$ 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).

**Remark 5** Note that the construction of the region graph depends on the junction tree. Using methods in Vats and Moura (2012), we can always construct junction trees such that the region graph only has two sets of regions, namely the clusters of the junction tree and the separators of the junction tree. 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 5, depends on the size of the regions.

**Remark 6 (Region graph vs. Junction tree)** 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 $\mathcal{R}^3, \ldots, \mathcal{R}^L$.

**Remark 7** 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.

## 4. 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:

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

We say $R$ connects to $S$ if $(R,S) \in \vec{E}(\mathcal{G})$. Thus, the children in (1) consist of 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

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

Thus, $\overline{R}$ *contains the union of all clusters in the junction tree that contain $R$.* 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[2]

$$H'_R = H[R] \backslash \left\{ \cup_{S \in ch(R)} K_S \right\}, \tag{3}$$

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 $\overline{R}$.

2. For graphs $G_1$ and $G_2$, $E(G_1 \backslash G_2) = E(G_1) \backslash E(G_2)$ 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 $\Psi$.
2: Compute $H'_R$ using (3) and $\overline{R}$ using (2).
3: Apply $\Psi$ to $\mathfrak{X}^n_{\overline{R}}$ to estimate edges in $H'_R$. See Appendix B for examples.
4: **Return** the estimated edges $\widehat{E}_R$.

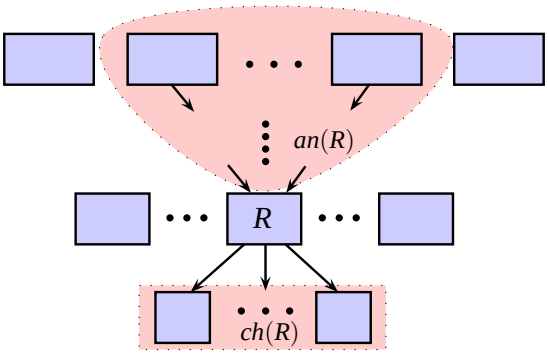



Figure 5: Notations defined on region graphs. The children $ch(R)$ are the set of regions that $R$ connects to. The ancestors $an(R)$ are all the regions that have a directed path to the region $R$. The set $\overline{R}$ takes the union of all regions in $an(R)$ and $R$.

**Proposition 8** *Suppose $E(G^*) \subseteq E(H)$. All edges in $H'_R$ can be estimated by solving a UGMS problem over the vertices $\overline{R}$.*

**Proof** See Appendix C. ■

Proposition 8 says that all edges in $H'_R$ can be estimated by applying a UGMS algorithm to the set of vertices $\overline{R}$. 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 $\overline{R}$. 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 the edges in $H'_R$ using the UGMS algorithm $\Psi$ 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 $\Psi$ to $R$.

2. Let $R = \{3, 5, 6\}$. The children of this region are $\{3, 5\}$, $\{5, 6\}$, and $\{3, 6\}$. This means that $H'_R = \emptyset$, that is, no edge over $H[R]$ can be estimated by applying $\Psi$ to $\{3, 5, 6\}$.

| 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 $H$ constructed using Algorithm 1. |
| $\mathcal{R} = (\mathcal{R}^1, \ldots, \mathcal{R}^L)$ | Partitioning of the regions in $\mathcal{R}$ into $L$ labels. |
| $\overline{R}$ | The set of vertices used when applying $\Psi$ to estimate edges over $R$. See (2) for definition. |
| $H'_R$ | Edges in $H[R]$ that can be estimated using Algorithm 2. See (3) for definition. |

Table 1: A summary of some notations.

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 $\Psi$ needs to be applied to $\overline{R} = \{2, 3, 4, 6, 7\}$.

## 5. UGMS Using Junction Trees: A General Framework

In this section, we present the junction tree framework for UGMS using the results from Sections 3-4. Section 5.1 presents the junction tree framework. Section 5.2 discusses the computational complexity of the framework. Section 5.3 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.

### 5.1 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 $\Psi$ in Assumption 1 needs to be applied to the set $\overline{R}$ 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 $\overline{R} = \{2, 3, 4, 5, 6, 7, 8\}$, 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 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 $\widehat{G}$ and all edges that are estimated are removed from $H$. Thus, $H$ now represents all the edges that are left to be estimated and $\widehat{G} \cup H$ contains

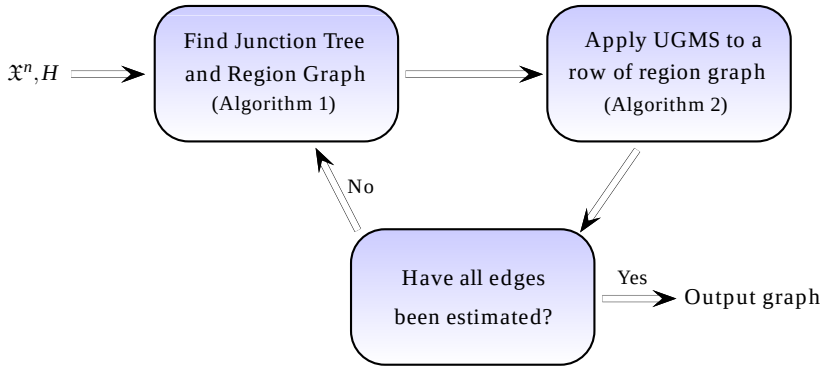



Figure 6: A high level overview of the junction tree framework for UGMS in Algorithm 3.

**Algorithm 3:** Junction Tree Framework for UGMS

See Table 1 for notations.

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

Step 2. Find the smallest $\ell$ 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 $\mathcal{R}^\ell$.

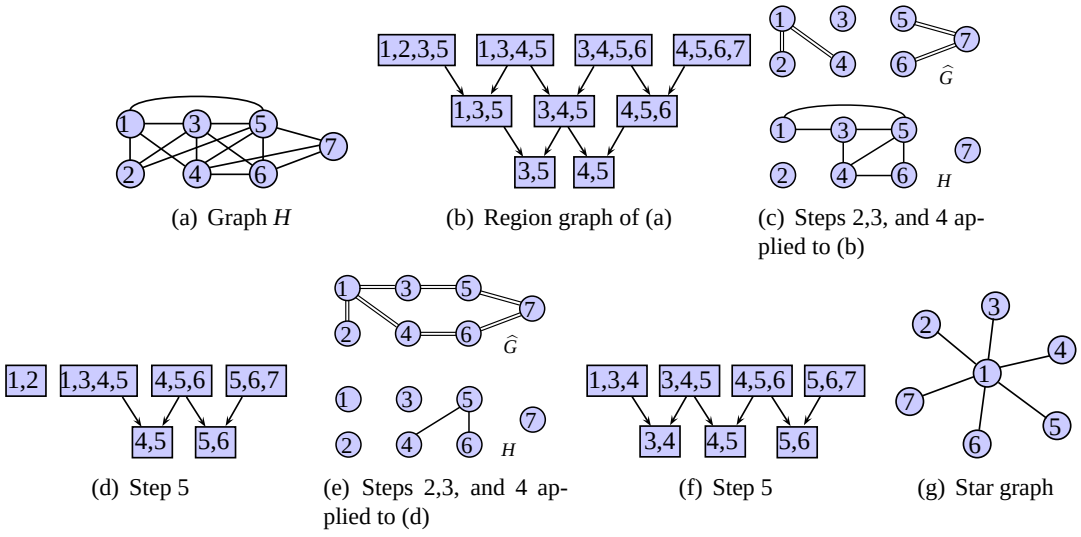
Step 4. Add all estimated edges to $\widehat{G}$ and *remove edges* from $H$ that have been estimated. Now $H \cup \widehat{G}$ contains all the edges in $G^*$.

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

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

all the edges in $G^*$. We repeat the above steps on a new region graph computed using $\widehat{G} \cup H$ 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 $\widehat{G}$ and $H$, respectively, after estimating all the edges in $\mathcal{R}^1$ of Figure 7(b). The edges in $\widehat{G}$ are represented by double lines to distinguish them from the edges in $H$. Figure 7(d) shows the region graph of $\widehat{G} \cup H$. Figure 7(e) shows the updated $\widehat{G}$ 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 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.




(a) Graph $H$ (b) Region graph of (a) (c) Steps 2,3, and 4 applied to (b)

(d) Step 5 (e) Steps 2,3, and 4 applied to (d) (f) Step 5 (g) Star graph

Figure 7: Example to illustrate the junction tree framework in Algorithm 3.

### 5.2 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 Fan and Lv (2008) and Vats (to appear), 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 $\overline{R}$ 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:* For a given graph, there exists several junction tree representations. The computational complexity of applying UGMS algorithms to a junction tree depends on the size of the clusters, the size of the separators, and the degree of the junction tree. In theory, it is useful to select a junction tree so that the overall computational complexity of the framework is as small as possible. However, this is hard

since there can be an exponential number of possible junction tree representations. Alternatively, we can select a junction tree so that the maximum size of the clusters is as small as possible. Such junction trees are often referred to as *optimal junction trees* in the literature. Although finding optimal junction trees is also hard (Arnborg et al., 1987), there exists several computationally tractable heuristics for finding close to optimal junction trees (Kjaerulff, 1990; Berry et al., 2003). The complexity of such algorithms range from $O(p^2)$ to $O(p^3)$, depending on the degree of approximation. We note that 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.

### 5.3 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 $\widehat{G}$ of $G^*$ (Meinshausen and Bühlmann, 2006; Bresler et al., 2008; Netrapalli et al., 2010; Ravikumar et al., 2010). 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). A junction tree representation of the star graph can be computed so that all clusters will have size two. Subsequently, the junction tree framework will only require 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 5.1 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 7 that the smaller subproblems lead to weaker conditions on the number of observations required for consistent graph estimation. Moreover, our numerical simulations in Section 8 empirically show the advantages of using junction tree over neighborhood selection based algorithms.

## 6. 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 7. The PC-Algorithm was originally proposed in the literature for learning directed graphical models (Spirtes and Glymour, 1991). 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 Anandkumar et al. (2012b,a). 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* $\kappa$: PC iteratively searches for separators for an edge $(i, j)$ by searching for separators of size $0, 1, \ldots, \kappa$. 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 $\kappa$ 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 Hoeffding (1948), Rasch et al. (2012) and Zhang et al. (2012) 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

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

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

**Inputs:**

$\kappa$: An integer that controls the computational complexity of $\mathsf{PC}$.

$\mathfrak{X}^n$: $n$ 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 $\widehat{G}$ that contains edges in $L$ that are estimated to be in $G^*$.

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

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

$$\Sigma_{A,B|S} = \Sigma_{A,B} - \Sigma_{A,S}\Sigma_{S,S}^{-1}\Sigma_{B,S}\,.$$

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

$$\text{Conditional Independence Test: } |\widehat{\rho}_{ij|S}| < \lambda_n \Longrightarrow X_i \perp\!\!\!\perp X_j | X_S\,, \tag{4}$$

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

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:

$$\widehat{G}_R = \mathsf{PC}\left(\eta, \mathfrak{X}^n, H, H'_R\right)\,, \tag{5}$$

where $H'_R$ is defined in (3). Notice that we do not use $\overline{R}$ in (5). This is because Line 4 of Algorithm 4 automatically finds the set of vertices to apply the $\mathsf{PC}$ algorithm to. Alternatively, we can apply Algorithm 4 using $\overline{R}$ as follows:

$$\widehat{G}_R = \mathsf{PC}\left(\eta, \mathfrak{X}^n_{\overline{R}}, K_{\overline{R}}, H'_R\right)\,, \tag{6}$$

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 $\mathsf{PC}$ and differentiates $\mathsf{PC}$ from the first stage of the SGS-Algorithm (Spirtes et al., 1990), which specifies $\mathcal{S}_{ij} = V \backslash \{i,j\}$.

## 7. 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 9, 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 9, the particular gain in performance depends on the structure of the graph.

Section 7.1 discusses the assumptions we place on the graphical model. Section 7.2 presents the main theoretical result highlighting the advantages of using junction trees. Throughout this section, 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 $\leq$.

### 7.1 Assumptions

(A1) *Gaussian graphical model:* We assume $X = (X_1, \ldots, X_p) \sim P_X$, where $P_X$ is a multivariate normal distribution with mean zero and covariance $\Sigma$. Further, $P_X$ is Markov on $G^*$ and not Markov on any subgraph of $G^*$. It is well known that this is assumption translates into the fact that $\Sigma^{-1}_{ij} = 0$ if and only if $(i,j) \notin G^*$ (Speed and Kiiveri, 1986).

(A2) *Faithfulness:* If $X_i \perp\!\!\!\perp X_j | X_S$, then $i$ and $j$ are separated by[3] $S$. 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 $\eta$:* For all $(i,j) \notin G^*$, there exists a subset of nodes $S \subset V \backslash \{i,j\}$, 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 $\Sigma$:* Under (A3), we assume that $\rho_{ij|S}$ satisfies

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

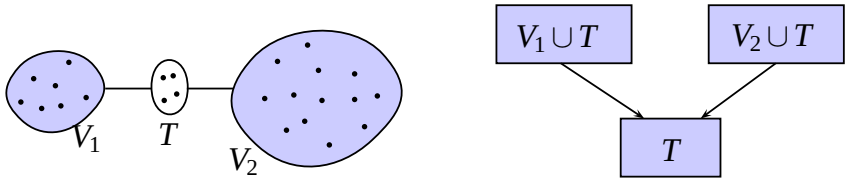
where $M$ is a constant. Further, we assume that $\max_{i,S,|S|\leq\eta} \Sigma_{i,i|S} \leq L < \infty$.

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

(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.

Assumptions (A1)-(A5) are standard conditions for proving high-dimensional consistency of the PC-Algorithm for Gaussian graphical models. The structural constraints on

3. If $S$ is the empty set, then there is no path between $i$ and $j$.




(a) Structure of the graph in (A5) (b) Region graph of (a)

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

the graph in Assumption (A6) are required for showing the advantages of the junction tree framework. We note that although (A6) appears to be a strong assumption, there are several graph families that satisfy this assumption. For example, the graph in Figure 1(a) satisfies (A6) with $V_1 = \{1, 2\}$, $V_2 = \{1, 3, 4, 5, 6, 7\}$, and $T = \{1\}$. In general, if there exists a separator in the graph of size less than $\eta$, then (A6) is clearly satisfied. Further, we remark that we only assume the *existence* of the sets $V_1$, $V_2$, and $T$ and do *not* assume that these sets are known *a priori*. We refer to Remark 17 for more discussions about (A6) and some extensions of this assumption.

### 7.2 Theoretical Result and Analysis

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

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

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

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

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

$$\widehat{G}_{V_k} = \mathsf{PC}\left(\eta, \mathfrak{X}^n, H[V_k \cup T] \cup K_T, H'_{V_k \cup T}\right), \text{for } k = 1, 2,$$

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

Step 3. Estimate edges over $T$ using a regularization parameter $\lambda_n^T$:

$$\widehat{G}_T = \mathsf{PC}\left(\eta, \mathfrak{X}^n, H[T \cup ne_{\widehat{G}_{V_1} \cup \widehat{G}_{V_2}}(T)], H[T]\right)\,.$$

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

Step 1 is the screening algorithm used to eliminate some edges from the complete graph. For the region graph in Figure 8(b), Step 2 corresponds to applying PC to the regions $V_1 \cup T$ and $V_2 \cup T$. Step 3 corresponds to applying PC to the region $T$ and all neighbors of $T$ estimated so far. Step 4 merges all the estimated edges. 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:

$$
\begin{aligned}
&p_1 = |V_1| + |T|\,,\ p_2 = |V_2| + |T|\,,\ p_T = |T \cup ne_{G^*}(T)|\,,\ \eta_T = |T|\,,\\
&\rho_0 = \inf\{|\rho_{ij|S}| : i,j \ s.t.\ |S| \leq \eta_T \ \&\ |\rho_{ij|S}| > 0\}\,,\\
&\rho_1 = \inf\{|\rho_{ij|S}| : i \in V_1, j \in V_1 \cup T \ s.t.\ (i,j) \in E(G^*), S \subseteq V_1 \cup T, |S| \leq \eta\}\,,\\
&\rho_2 = \inf\{|\rho_{ij|S}| : i \in V_2, j \in V_2 \cup T \ s.t.\ (i,j) \in E(G^*), S \subseteq V_2 \cup T, |S| \leq \eta\}\,,\\
&\rho_T = \inf\{|\rho_{ij|S}| : i,j \in T \ s.t.\ (i,j) \in E, S \subseteq T \cup ne_{G^*}(T), \eta_T < |S| \leq \eta\}\,,
\end{aligned}
$$

The term $\rho_0$ 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 $\rho_1$ and $\rho_2$ are measures of how hard it is learn the edges in $G^*[V_1 \cup T]\backslash K_T$ and $G^*[V_2 \cup T]\backslash K_T$ (Step 2), respectively, given that $E(G^*) \subseteq E(H)$. The term $\rho_T$ is a measure of how hard it is 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 9** *Under Assumptions (A1)-(A6), there exists a conditional independence test such that if*

$$
n = \Omega\left(\max\left\{\rho_0^{-2}\eta_T \log(p), \rho_1^{-2}\eta \log(p_1), \rho_2^{-2}\eta \log(p_2), \rho_T^{-2}\eta \log(p_T)\right\}\right)\,, \tag{7}
$$

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

**Proof** See Appendix E. ■

We now make several remarks regarding Theorem 9 and its consequences.

**Remark 10 (Comparison to Necessary Conditions)** Using results from Wang et al. (2010), it follows that a necessary condition for any algorithm to recover the graph $G^*$ that satisfies Assumptions (A1) and (A6) is that $n = \Omega(\max\{\theta_1^{-2}\log(p_1 - d), \theta_2^{-2}\log(p_2 - d)\}$, where $d$ is the maximum degree of the graph and $\theta_1$ and $\theta_2$ are defined as follows:

$$
\theta_k = \min_{(i,j) \in G^*[V_k \cup T]\backslash G^*[T]} \frac{|\Sigma^{-1}_{ij}|}{\sqrt{|\Sigma^{-1}_{ii}\Sigma^{-1}_{jj}|}}\,, k = 1, 2\,.
$$

If $\eta$ is a constant and $\rho_1$ and $\rho_2$ are chosen so that the corresponding expressions dominate all other expressions, then (7) reduces to $n = \Omega(\max\{\rho_1^{-2}\log(p_1), \rho_2^{-2}\log(p_2)\})$. Furthermore, for certain classes of Gaussian graphical models, namely walk summable graphical models (Malioutov et al., 2006), the results in Anandkumar et al. (2012a) show that there exists conditions under which $\rho_1 = \Omega(\theta_1)$ and $\rho_2 = \Omega(\theta_2)$. In this case, (7) is equivalent to $n = \Omega(\max\{\theta_1^{-2}\log(p_1), \theta_2^{-2}\log(p_2)\})$. Thus, as long as $p_1, p_2 \gg d$, there exists a family

of graphical models for which the sufficient conditions in Theorem 9 nearly match the necessary conditions for asymptotically reliable estimation of the graph. We note that the particular family of graphical models is quite broad, and includes forests, scale-free graphs, and some random graphs. We refer to Anandkumar et al. (2012a) for a characterization of such graphical models.

**Remark 11 (Choice of Regularization Parameters)** We use the conditional independence test in (4) 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:

$$\begin{aligned}
&\lambda_n^0 = O(\rho_0) \text{ and } \rho_0 = \Omega\left(\sqrt{\eta_T \log(p)/n}\right), \\
&\lambda_n^k = O(\rho_k) \text{ and } \rho_k = \Omega\left(\sqrt{\eta \log(p_k)/n}\right), k = 1, 2, \\
&\lambda_n^T = O(\rho_T) \text{ and } \rho_T = \Omega\left(\sqrt{\eta \log(p_T)/n}\right).
\end{aligned}$$

We clearly see that different regularization parameters are used to estimate different parts of the graph. Furthermore, just like in the traditional analysis of UGMS algorithms, the optimal choice of the regularization parameter depends on unknown parameters of the graphical model. In practice, we use model selection algorithms to select regularization parameters. We refer to Section 8 for more details.

**Remark 12 (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

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

This result is proved in Appendix D using results from Kalisch and Bühlmann (2007) and Anandkumar et al. (2012a). Since $\rho_{min} \leq \min\{\rho_0, \rho_1, \rho_2, \rho_T\}$, it is clear that (7) 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 13 (Reduced Sample Complexity)** Suppose $\eta$, $\rho_0$, and $\rho_T$ are constants and $\rho_1 < \rho_2$. In this case, (7) reduces to

$$n = \Omega\left(\max\left\{\log(p), \rho_1^{-2}\log(p_1), \rho_2^{-2}\log(p_2)\right\}\right). \tag{8}$$

If $\rho_1^{-2} = \Omega\left(\max\left\{\rho_2^{-2}\log(p_2)/\log(p_1), \log(p)\right\}\right)$, then (8) reduces to

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

On the other hand, if we do not use junction trees, $n = \Omega\left(\rho_{min}^{-2}\log(p)\right)$, 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

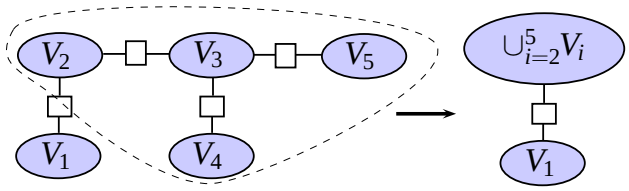



Figure 9: Junction tree representation with clusters $V_1, \ldots, V_5$ and separators denotes by rectangular boxes. We can cluster vertices in the junction tree to get a two cluster representation as in Figure 8.

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 may accurately learn the graph using significantly less number of observations.

**Remark 14 (Learning Weak Edges)** We now analyze Theorem 9 to see how the conditional correlation coefficients scale for high-dimensional consistency. Under the assumption in Remark 13, it is easy to see that the minimal conditional correlation coefficient scales as $\Omega(\sqrt{\log(p_1)/n})$ when using junction trees and as $\Omega(\sqrt{\log(p)/n})$ when not using junction trees. This suggests that when $p_1 \ll p$, it may be possible to learn edges with weaker conditional correlation coefficients when using junction trees. Our numerical simulations in Section 8 empirically show this property of the junction tree framework.

**Remark 15 (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 edge requires a search over at most $O(p^\eta)$ separators. The worst case computational complexity of Steps 1-4 is roughly $O\left(p^{|T|+2} + p_1^{\eta+2} + p_2^{\eta+2} + p_T^{\eta+2}\right)$. 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 16 (Using other UGMS Algorithms)** Although our analysis used the PC-Algorithm to derive sufficient conditions for accurately estimating the graph, we can easily use other algorithms, such as the graphical Lasso or the neighborhood selection based Lasso, for analysis. The main difference will be in the assumptions imposed on the graphical model.

**Remark 17 (Extensions)** We have analyzed the junction tree framework assuming that the junction tree of $H$ only has two clusters. One way to generalize our analysis to junction trees with multiple clusters is to merge clusters so that the resulting junction tree admits

the structure in Figure 8. For example, suppose the graph $G^*$ has a junction tree representation as in Figure 9 with five clusters. If $|V_1 \cap V_2| < \eta$, then we can merge the clusters $V_2, V_3, \ldots, V_5$ so that the resulting junction tree admits the two cluster representation in Figure 8. Furthermore, we can also generalize Theorem 9 to cases when $|T| = \eta$. The main change in the analysis will be in the definition of $\rho_0$. For example, if the graph is a chain so that the first $p_1$ vertices are associated with "weak edges", we can get similar results as in Theorem 9. Finally, we note that a full analysis of the junction tree framework, that also incorporates the step of updating the junction tree in Algorithm 3, is challenging and will be addressed in future work.

## 8. 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 $(i,j) \notin G^* \iff \Theta_{ij} = 0$ (Speed and Kiiveri, 1986). Some algorithmic details used in the simulations are described as follows.

*Computing $H$:* We apply Algorithm 4 with a suitable value of $\kappa$ in such a way that the separator search space $\mathcal{S}_{ij}$ (see Line 4) is restricted to be small. In other words, we do not test for all possible conditional independence tests so as to restrict the computational complexity of the screening algorithm. 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 7.2 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 $\widehat{G}$ 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 trees.[4] Once the junction tree is computed, we merge clusters so that the maximum size of the separator is at most $\kappa + 1$, where $\kappa$ is the parameter used when computing the graph $H$. For example, in Figure 9, if the separator associated with $V_2$ and $V_3$ has cardinality greater than $\kappa + 1$, then we merge $V_2$ and $V_3$ and resulting junction tree is such that $V_1$, $V_4$, and $V_5$ all connect to the cluster $V_2 \cup V_3$.

*UGMS Algorithms:* We apply the junction tree framework in conjunction with graphical Lasso (gL) (Banerjee et al., 2008), neighborhood selection using Lasso (nL) (Meinshausen and Bühlmann, 2006), and the PC-Algorithm (PC) (Spirtes and Glymour, 1991). 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 (Zou, 2006) for finding neighbors of a vertex since this is known to give superior results for variable selection (van de Geer et al., 2011).

4. We use the GreedyFillin heuristic. This is known to give good results with reasonable computational time (Kjaerulff, 1990).

*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 (Meinshausen and Bühlmann, 2010; Liu et al., 2010). Several alternative methods have been proposed in the literature; see for example stability selection (Meinshausen and Bühlmann, 2010; Liu et al., 2010) and extended Bayesian information (EBIC) criterion (Chen and Chen, 2008; Foygel and Drton, 2010). In all our simulations, we use EBIC since it is much faster than stability based methods when the distribution is Gaussian. EBIC selects a regularization parameter $\widehat{\lambda}_n$ as follows:

$$\widehat{\lambda}_n = \max_{\lambda_n > 0} \left\{ n \left[ \log\det \widehat{\Theta}_{\lambda_n} - \mathrm{trace}(\widehat{S}\Theta) \right] + |E(\widehat{G}_{\lambda_n})| \log n + 4\gamma |E(\widehat{G}_{\lambda_n})| \log p \right\},$$

where $\widehat{S}$ is the empirical covariance matrix, $\widehat{\Theta}_{\lambda_n}$ is the estimate of the inverse covariance matrix and $|E(\widehat{G}_{\lambda_n})|$ is the number of edges in the estimated graph. The estimate $\widehat{\lambda}_n$ depends on a parameter $\gamma \in [0, 1]$ such that $\gamma = 0$ results in the BIC estimate and increasing $\gamma$ produces sparser graphs. The authors in reference Foygel and Drton (2010) 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 $\log |\overline{R}|$, $\widehat{\Theta}_{\lambda_n}$ is replaced by the inverse covariance over the vertices $\overline{R}$, and $|\widehat{G}_{\lambda_n}|$ is replaced by the number of edges estimated from the graph $H'_R$.
*Small subproblems:* Whenever $|\overline{R}|$ 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.

### 8.1 Results on Synthetic Graphs

We assume that $\Theta_{ii} = 1$ for all $i = 1, \ldots, p$. 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 ($\mathsf{CH}_1$ and $\mathsf{CH}_2$): $\Theta_{i,i+1} = \rho_1$ for $i = 1, \ldots, p_1 - 1$ (weak edges) and $\Theta_{i,i+1} = \rho_2$ for $i = p_1, p - 1$ (strong edges). For $\mathsf{CH}_1$, $\rho_1 = 0.15$ and $\rho_2 = 0.245$. For $\mathsf{CH}_2$, $\rho_1 = 0.075$ and $\rho_2 = 0.245$. Let $\Theta_{ij} = \Theta_{ji}$.

- Cycle ($\mathsf{CY}_1$ and $\mathsf{CY}_2$): $\Theta_{i,i+1} = \rho_1$ for $i = 1, \ldots, p_1 - 1$ (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 $i = 1, \ldots, p_1 - 3$ and $\Theta_{i,i+3} = \rho_2$ for $i = p_1, p_1 + 1, \ldots, p - 3$. This introduces multiple cycles in the graph. For $\mathsf{CY}_1$, $\rho_1 = 0.15$ and $\rho_2 = 0.245$. For $\mathsf{CY}_2$, $\rho_1 = 0.075$ and $\rho_2 = 0.245$.

- Hub ($\mathsf{HB}_1$ and $\mathsf{HB}_2$): For the first $p_1$ vertices, construct as many star[5] 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 $\mathsf{HB}_1$, $d_1 = 8$ and $d_2 = 5$. For $\mathsf{HB}_2$, $d_1 = 12$ and $d_2 = 5$.

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

- Neighborhood graph ($\mathsf{NB_1}$ and $\mathsf{NB_2}$): Randomly place vertices on the unit square at coordinates $y_1, \ldots, y_p$. 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 $i, j \in \{1, \ldots, p_1\}$ such that $i > j$. For all $i, j \in \{p_1+1, \ldots, p\}$ 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 $p_1 + 1, \ldots, p$, delete edges such that the neighborhood of each vertex is at most $d_2$. Finally, randomly add four edges from a vertex in $\{1, \ldots, p_1\}$ to a vertex in $\{p_1, p_1 + 1, \ldots, p\}$ such that for each such edge, $\Theta_{ij} = \rho_1$. We let $\rho_2 = 0.245$, $d_1 = 6$, and $d_2 = 4$. For $\mathsf{NB_1}$, $\rho_1 = 0.15$ and for $\mathsf{NB_2}$, $\rho_2 = 0.075$.

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).

- The nL and JnL estimates are computed differently since it is difficult to control the number of edges estimated using both these algorithms.[6] We apply both nL and JnL for multiple different values of $\gamma$ (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 $\kappa$ as 1, 2, 1, and 3 for Chain, Cycle, Hub, and Neighborhood graphs, respectively. When computing $H$, we choose $\kappa$ 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 $\widehat{G}$ of $G^*$, we evaluate $\widehat{G}$ using the weak edge discovery rate (WEDR), false discovery rate (FDR), true positive rate (TPR), and the edit distance (ED).

$$\begin{aligned}
\text{WEDR} &= \frac{\#\text{ weak edges in } \widehat{G}}{\#\text{ of weak edges in } G^*}, \\
\text{FDR} &= \frac{\#\text{ of edges in } \widehat{G} \backslash G^*}{\#\text{ of edges in } \widehat{G}}, \\
\text{TPR} &= \frac{\#\text{ of edges in } \widehat{G} \cap G^*}{\#\text{ of edges in } G^*}, \\
\text{ED} &= \{\#\text{ edges in } \widehat{G} \backslash G^*\} + \{\#\text{ edges in } G^* \backslash \widehat{G}\},
\end{aligned}$$

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

| Model | $n$ | Alg | WEDR | FDR | TPR | ED | $\lvert\widehat{G}\rvert$ |
|---|---|---|---|---|---|---|---|
| $\mathsf{CH}_1$ | 300 | JgL | **0.305** (0.005) | 0.048 (0.001) | 0.767 (0.002) | 27.0 (0.176) | 79.8 |
| $p = 100$ | | gL | 0.180 (0.004) | 0.061 (0.001) | 0.757 (0.001) | 29.0 (0.153) | 79.8 |
| | | JPC | **0.312** (0.004) | 0.047 (0.001) | 0.775 (0.001) | 26.0 (0.162) | 80.5 |
| | | PC | 0.264 (0.005) | 0.047 (0.001) | 0.781 (0.001) | 25.6 (0.169) | 81.2 |
| | | JnL | **0.306** (0.005) | 0.072 (0.001) | 0.769 (0.002) | 28.8 (0.188) | 82.1 |
| | | nL | 0.271 (0.005) | 0.073 (0.001) | 0.757 (0.001) | 30.0 (0.197) | 80.9 |
| $\mathsf{CH}_2$ | 300 | JgL | **0.052** (0.002) | 0.067 (0.001) | 0.727 (0.001) | 32.2 (0.173) | 77.3 |
| $p = 100$ | | gL | 0.009 (0.001) | 0.062 (0.001) | 0.733 (0.002) | 31.3 (0.162) | 77.4 |
| | | JPC | **0.048** (0.002) | 0.064 (0.001) | 0.735 (0.001) | 31.2 (0.169) | 77.8 |
| | | PC | 0.0337 (0.002) | 0.055 (0.001) | 0.748 (0.001) | 29.3 (0.144) | 78.4 |
| | | JnL | **0.052** (0.002) | 0.077 (0.001) | 0.733 (0.001) | 32.5 (0.186) | 78.7 |
| | | nL | 0.039 (0.002) | 0.086 (0.001) | 0.723 (0.001) | 34.2 (0.216) | 78.4 |
| $\mathsf{CH}_1$ | 500 | JgL | **0.596** (0.006) | 0.021 (0.001) | 0.916 (0.001) | 10.2 (0.133) | 92.6 |
| $p = 100$ | | gL | 0.44 (0.005) | 0.050 (0.001) | 0.889 (0.001) | 15.6 (0.132) | 92.7 |
| | | JPC | **0.612** (0.005) | 0.022 (0.001) | 0.921 (0.001) | 9.86 (0.128) | 93.2 |
| | | PC | 0.577 (0.005) | 0.032 (0.001) | 0.916 (0.001) | 11.4 (0.124) | 93.7 |
| | | JnL | **0.623** (0.005) | 0.059 (0.001) | 0.922 (0.001) | 13.5 (0.133) | 97.0 |
| | | nL | 0.596 (0.005) | 0.069 (0.001) | 0.918 (0.001) | 14.9 (0.164) | 97.6 |
| $\mathsf{CH}_2$ | 500 | JgL | **0.077** (0.002) | 0.044 (0.001) | 0.816 (0.001) | 22.0 (0.107) | 84.5 |
| $p = 100$ | | gL | 0.0211 (0.001) | 0.053 (0.001) | 0.808 (0.000) | 23.5 (0.082) | 84.6 |
| | | JPC | **0.073** (0.002) | 0.042 (0.001) | 0.817 (0.001) | 21.7 (0.082) | 84.5 |
| | | PC | 0.0516 (0.002) | 0.049 (0.001) | 0.815 (0.001) | 22.5 (0.092) | 84.9 |
| | | JnL | **0.076** (0.002) | 0.070 (0.001) | 0.818 (0.001) | 24.2 (0.102) | 87.2 |
| | | nL | 0.066 (0.002) | 0.077 (0.001) | 0.815 (0.001) | 25.1 (0.126) | 87.5 |

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

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 18 (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 19 (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 20 (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 Anandkumar et al. (2012b,a), can be used to weaken the assumption in (A6).

| Model | $n$ | Alg | WEDR | FDR | TPR | ED | $\lvert\widehat{G}\rvert$ |
|---|---|---|---|---|---|---|---|
| $\mathsf{CY}_1$ | 300 | JgL | **0.314** (0.003) | 0.036 (0.001) | 0.814 (0.001) | 28.5 (0.142) | 111 |
| $p = 100$ | | gL | 0.105 (0.003) | 0.057 (0.001) | 0.798 (0.001) | 32.9 (0.16) | 112 |
| | | JPC | **0.326** (0.004) | 0.030 (0.001) | 0.819 (0.001) | 27.2 (0.18) | 112 |
| | | PC | 0.307 (0.004) | 0.027 (0.001) | 0.826 (0.001) | 26 (0.169) | 112 |
| | | JnL | **0.342** (0.004) | 0.043 (0.001) | 0.813 (0.001) | 29.5 (0.175) | 112 |
| | | nL | 0.299 (0.004) | 0.044 (0.001) | 0.793 (0.001) | 32.3 (0.192) | 110 |
| $\mathsf{CY}_2$ | 300 | JgL | **0.047** (0.002) | 0.045 (0.001) | 0.762 (0.001) | 36.2 (0.163) | 105 |
| $p = 100$ | | gL | 0.001 (0.001) | 0.049 (0.001) | 0.759 (0.001) | 37.0 (0.172) | 105 |
| | | JPC | **0.043** (0.002) | 0.042 (0.001) | 0.764 (0.001) | 35.6 (0.174) | 105 |
| | | PC | 0.027 (0.002) | 0.036 (0.001) | 0.773 (0.001) | 33.7 (0.137) | 106 |
| | | JnL | **0.042** (0.002) | 0.058 (0.002) | 0.754 (0.001) | 38.6 (0.210) | 106 |
| | | nL | 0.035 (0.002) | 0.057 (0.002) | 0.743 (0.001) | 39.9 (0.228) | 104 |
| $\mathsf{CY}_1$ | 500 | JgL | **0.532** (0.005) | 0.022 (0.001) | 0.907 (0.001) | 15.1 (0.139) | 122 |
| $p = 100$ | | gL | 0.278 (0.001) | 0.071 (0.001) | 0.862 (0.001) | 26.9 (0.178) | 122 |
| | | JPC | **0.61** (0.004) | 0.012 (0.001) | 0.925 (0.001) | 11.9 (0.150) | 124 |
| | | PC | 0.609 (0.004) | 0.020 (0.001) | 0.925 (0.001) | 12.5 (0.134) | 125 |
| | | JnL | **0.612** (0.005) | 0.028 (0.001) | 0.924 (0.001) | 13.6 (0.151) | 125 |
| | | nL | 0.584 (0.005) | 0.041 (0.001) | 0.919 (0.001) | 15.9 (0.171) | 126 |
| $\mathsf{CY}_2$ | 500 | JgL | **0.086** (0.003) | 0.039 (0.001) | 0.821 (0.001) | 28.1 (0.116) | 113 |
| $p = 100$ | | gL | 0.004 (0.001) | 0.058 (0.001) | 0.805 (0.000) | 32.3 (0.088) | 113 |
| | | JPC | **0.087** (0.002) | 0.034 (0.001) | 0.825 (0.001) | 27.0 (0.099) | 113 |
| | | PC | 0.074 (0.002) | 0.040 (0.001) | 0.823 (0.001) | 27.9 (0.010) | 113 |
| | | JnL | **0.085** (0.003) | 0.045 (0.001) | 0.824 (0.001) | 28.4 (0.147) | 114 |
| | | nL | 0.069 (0.003) | 0.053 (0.001) | 0.821 (0.001) | 29.8 (0.158) | 114 |

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

| Model | $n$ | Alg | WEDR | FDR | TPR | ED | $\lvert\widehat{G}\rvert$ |
|---|---|---|---|---|---|---|---|
| $\mathsf{HB}_1$ | 300 | JgL | **0.204** (0.004) | 0.039 (0.001) | 0.755 (0.002) | 22.3 (0.151) | 63.7 |
| $p = 100$ | | gL | 0.154 (0.004) | 0.038 (0.001) | 0.758 (0.002) | 22.1 (0.130) | 63.8 |
| | | JPC | **0.204** (0.004) | 0.038 (0.001) | 0.753 (0.002) | 22.4 (0.160) | 63.4 |
| | | PC | 0.193 (0.004) | 0.038 (0.001) | 0.762 (0.002) | 21.7 (0.143) | 64.2 |
| | | JnL | 0.245 (0.005) | 0.089 (0.001) | 0.750 (0.002) | 26.2 (0.174) | 66.7 |
| | | nL | **0.247** (0.005) | 0.098 (0.002) | 0.752 (0.002) | 26.8 (0.198) | 67.6 |
| $\mathsf{HB}_2$ | 300 | JgL | **0.044** (0.002) | 0.047 (0.001) | 0.710 (0.001) | 26.7 (0.116) | 61.2 |
| $p = 100$ | | gL | 0.013 (0.002) | 0.043 (0.001) | 0.716 (0.001) | 26.0 (0.121) | 61.4 |
| | | JPC | **0.048** (0.002) | 0.043 (0.001) | 0.709 (0.001) | 26.5 (0.108) | 60.8 |
| | | PC | 0.029 (0.002) | 0.038 (0.001) | 0.718 (0.001) | 25.5 (0.121) | 61.3 |
| | | JnL | **0.054** (0.003) | 0.083 (0.001) | 0.704 (0.001) | 29.6 (0.146) | 63.0 |
| | | nL | 0.0467 (0.002) | 0.096 (0.001) | 0.700 (0.001) | 30.7 (0.138) | 63.5 |
| $\mathsf{HB}_1$ | 500 | JgL | **0.413** (0.007) | 0.026 (0.001) | 0.870 (0.002) | 12.4 (0.156) | 72.4 |
| $p = 100$ | | gL | 0.364 (0.007) | 0.035 (0.001) | 0.863 (0.002) | 13.7 (0.144) | 72.5 |
| | | JPC | 0.438 (0.007) | 0.027 (0.001) | 0.878 (0.002) | 11.9 (0.148) | 73.1 |
| | | PC | **0.448** (0.007) | 0.027 (0.001) | 0.882 (0.001) | 11.6 (0.141) | 73.4 |
| | | JnL | 0.507 (0.006) | 0.076 (0.001) | 0.890 (0.001) | 14.9 (0.152) | 78.2 |
| | | nL | **0.52** (0.007) | 0.091 (0.001) | 0.893 (0.002) | 15.9 (0.191) | 79.6 |
| $\mathsf{HB}_2$ | 500 | JgL | **0.086** (0.003) | 0.042 (0.001) | 0.794 (0.001) | 19.8 (0.086) | 68.0 |
| $p = 100$ | | gL | 0.050 (0.002) | 0.047 (0.001) | 0.789 (0.001) | 20.6 (0.098) | 68.0 |
| | | JPC | **0.097** (0.003) | 0.040 (0.001) | 0.798 (0.001) | 19.3 (0.109) | 68.2 |
| | | PC | 0.087 (0.003) | 0.044 (0.001) | 0.797 (0.001) | 19.7 (0.111) | 68.4 |
| | | JnL | **0.123** (0.004) | 0.084 (0.002) | 0.804 (0.001) | 22.2 (0.15) | 72.1 |
| | | nL | 0.106 (0.003) | 0.105 (0.002) | 0.801 (0.001) | 24.1 (0.143) | 73.4 |

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

| Model | $n$ | Alg | WEDR | FDR | TPR | ED | $\lvert\widehat{G}\rvert$ |
|---|---|---|---|---|---|---|---|
| $\mathsf{NB}_1$ | 300 | JgL | **0.251** (0.002) | 0.030 (0.000) | 0.813 (0.000) | 126 (0.329) | 498 |
| $p = 100$ | | gL | 0.102 (0.0015) | 0.039 (0.000) | 0.806 (0.001) | 135 (0.345) | 498 |
| | | JPC | **0.259** (0.002) | 0.031 (0.000) | 0.814 (0.000) | 126 (0.260) | 499 |
| | | PC | 0.255 (0.002) | 0.036 (0.000) | 0.813 (0.000) | 129 (0.330) | 501 |
| | | JnL | **0.254** (0.002) | 0.035 (0.000) | 0.812 (0.001) | 129 (0.461) | 500 |
| | | nL | 0.226 (0.002) | 0.039 (0.000) | 0.804 (0.001) | 136 (0.458) | 497 |
| $\mathsf{NB}_1$ | 300 | JgL | **0.005** (0.000) | 0.043 (0.000) | 0.784 (0.001) | 149 (0.385) | 486 |
| $p = 100$ | | gL | 0.000 (0.000) | 0.036 (0.000) | 0.790 (0.000) | 142 (0.259) | 487 |
| | | JPC | **0.004** (0.000) | 0.042 (0.000) | 0.784 (0.001) | 148 (0.376) | 486 |
| | | PC | 0.003 (0.000) | 0.048 (0.000) | 0.782 (0.000) | 153 (0.239) | 488 |
| | | JnL | **0.005** (0.000) | 0.046 (0.000) | 0.783 (0.000) | 151 (0.356) | 488 |
| | | nL | 0.003 (0.000) | 0.050 (0.000) | 0.775 (0.000) | 158 (0.374) | 485 |
| $\mathsf{NB}_1$ | 500 | JgL | **0.449** (0.001) | 0.018 (0.000) | 0.921 (0.000) | 57.1 (0.199) | 557 |
| $p = 100$ | | gL | 0.319 (0.002) | 0.035 (0.000) | 0.905 (0.000) | 75.8 (0.242) | 557 |
| | | JPC | 0.489 (0.002) | 0.019 (0.000) | 0.925 (0.000) | 52.8 (0.189) | 558 |
| | | PC | **0.496** (0.002) | 0.023 (0.000) | 0.920 (0.000) | 60.2 (0.214) | 559 |
| | | JnL | **0.508** (0.003) | 0.027 (0.000) | 0.929 (0.000) | 57.9 (0.348) | 567 |
| | | nL | 0.494 (0.003) | 0.033 (0.000) | 0.927 (0.000) | 62.3 (0.400) | 570 |
| $\mathsf{NB}_2$ | 500 | JgL | **0.008** (0.000) | 0.033 (0.000) | 0.870 (0.000) | 95.0 (0.206) | 534 |
| $p = 100$ | | gL | 0.000 (0.000) | 0.034 (0.000) | 0.869 (0.000) | 96.0 (0.214) | 534 |
| | | JPC | **0.009** (0.000) | 0.032 (0.000) | 0.870 (0.000) | 94.2 (0.215) | 534 |
| | | PC | 0.005 (0.000) | 0.040 (0.000) | 0.865 (0.000) | 102 (0.207) | 536 |
| | | JnL | **0.001** (0.000) | 0.038 (0.000) | 0.871 (0.000) | 97.3 (0.220) | 538 |
| | | nL | 0.005 (0.000) | 0.043 (0.000) | 0.870 (0.000) | 101 (0.234) | 540 |

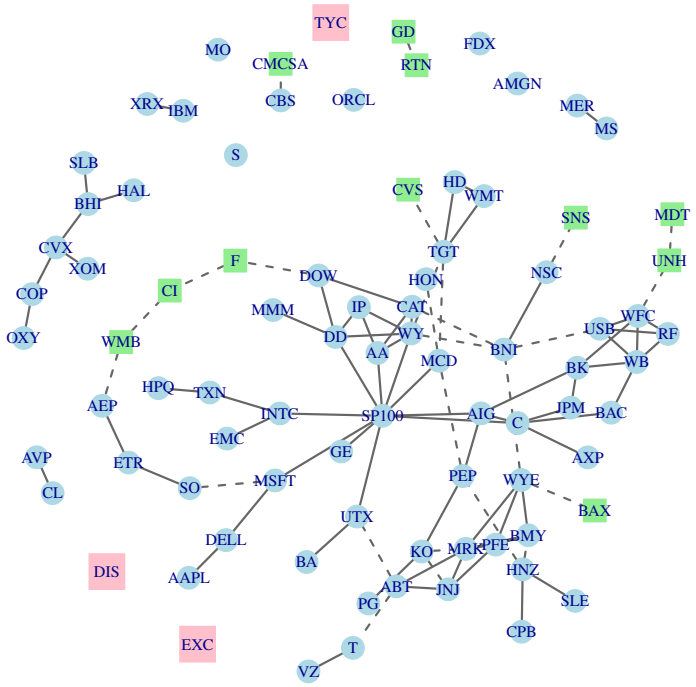
Table 5: Results for Neighborhood graph, $p = 300$ and $p_1 = 30$

**Remark 21 (Hub Graph)** For the hub graph $\mathsf{HB}_1$, 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 $\mathsf{HB}_2$, the junction tree estimate, in general, performs slightly better. This is because the parameters associated with the weak edges in $\mathsf{HB}_2$ are smaller than that of $\mathsf{HB}_1$.

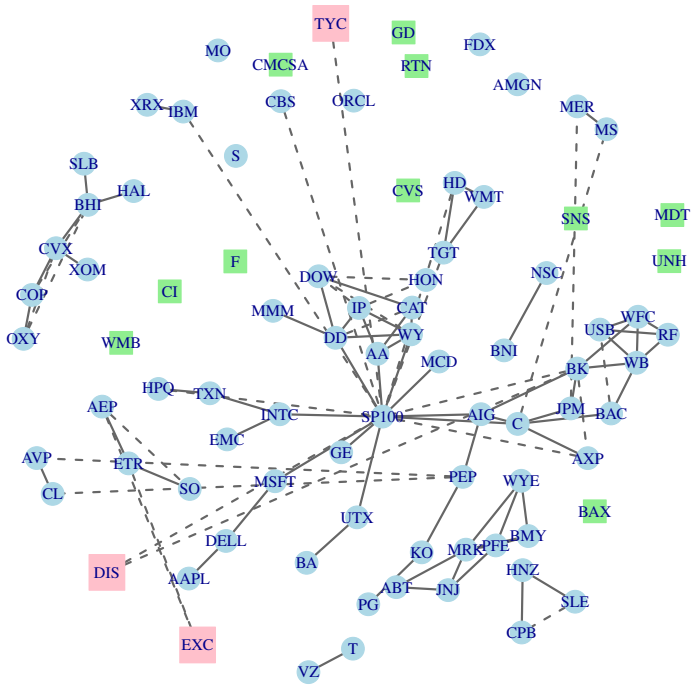
**Remark 22 (General Conclusion)** We see that, in general, the WEDR and TPR are higher, while the FDR and ED are lower, for junction tree based algorithms. This clearly suggests that using junction trees results in more accurate graph estimation. Moreover, the higher WEDR suggest that the main differences between the two algorithms are over the weak edges, that is, junction tree based algorithms are estimating more weak edges when compared to a non junction tree based algorithm.

### 8.2 Analysis of Stock Returns Data

We applied our methods to the data set in Choi et al. (2011) 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 10 shows the two estimated graphs in such a way that the vertices are positioned so that the JgL graph looks aesthetically pleasing. In Figure 11, the vertices are positioned so that gL looks aesthetically pleasing. In each graph, we mark the common edges by bold lines and the

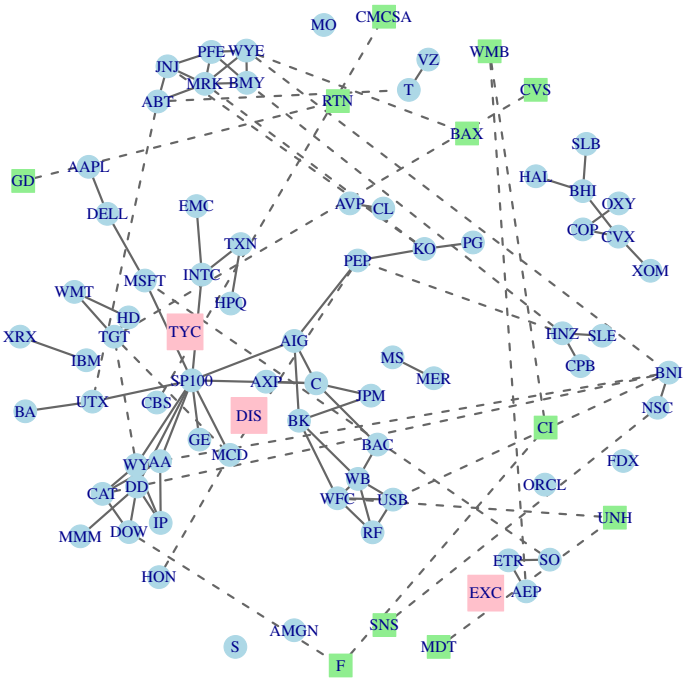



(a) Junction tree based graphical Lasso

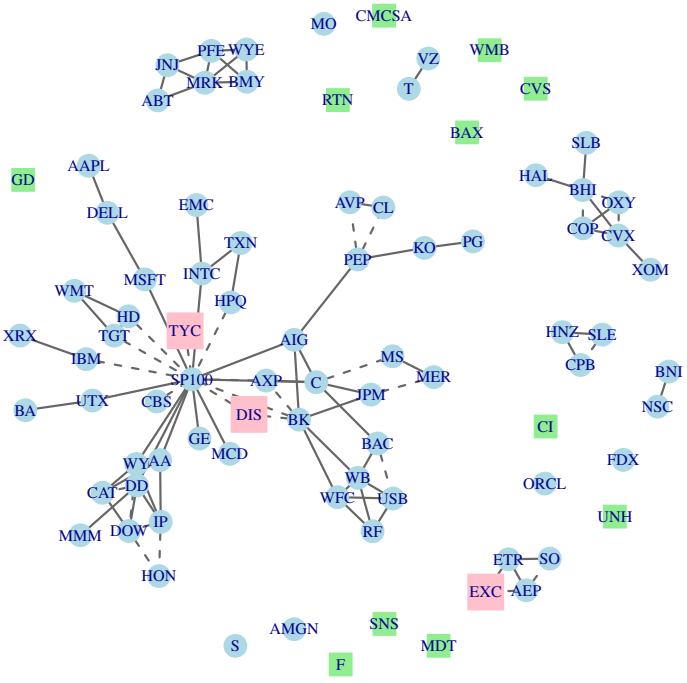



(b) Graphical Lasso

Figure 10: Graph over a subset of companies in the S&P 100. The positioning of the vertices is chosen so that the *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 11: 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.

remaining edges by dashed lines. Some conclusions that we draw from the estimated graphs are summarized as follows:

- The gL graph in Figure 11(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 10(a) with the exception that there are now connections between the clusters. As observed in Choi et al. (2011) and Chandrasekaran et al. (2012), 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.

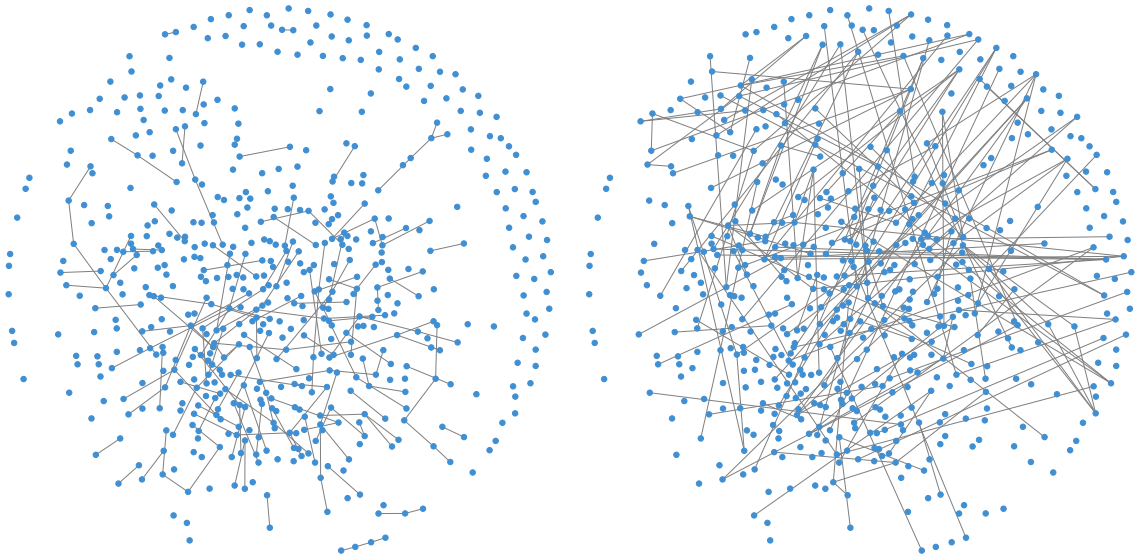
- We now focus on some specific edges and nodes in the graphs. The 11 vertices represented by smaller squares and shaded in green are not connected to any other vertex in gL. On the other hand, all these 11 vertices are connected to at least one other vertex in JgL (see Figure 10). Moreover, several 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.

### 8.3 Analysis of Gene Expression Data

Graphical models have been used extensively for studying gene interactions using gene expression data (Nevins et al., 2004; Wille et al., 2004). The gene expression data we study is the Lymph node status data which contains $n = 148$ expression values from $p = 587$ genes (Li and Toh, 2010). 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 12 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.

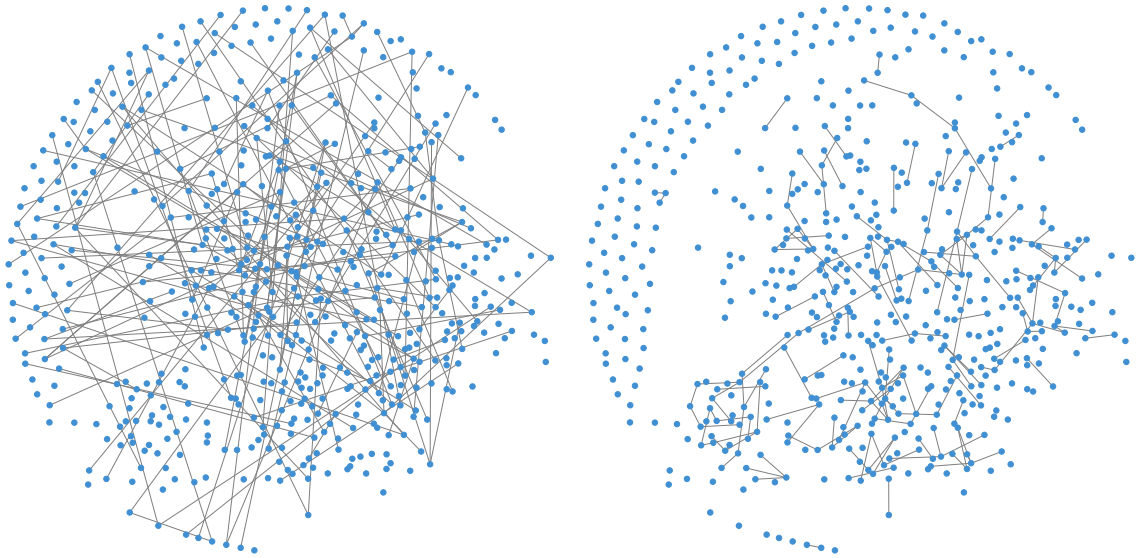
## 9. Summary and Future Work

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 subprolems 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



(a) Junction tree based graphical Lasso

(b) Graphical Lasso



(c) Junction tree based graphical Lasso

(d) Graphical Lasso

Figure 12: 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 (c), we only show edges that are estimated by graphical Lasso, but not by the junction tree estimate.

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.

## Acknowledgments

The first author thanks the Institute for Mathematics and its Applications (IMA) for financial support in the form of a postdoctoral fellowship. The authors thank Vincent Tan for discussions and comments on an earlier version of the paper. The authors thank the anonymous reviewers for comments which significantly improved this manuscript.

## Appendix A. Marginal Graph

**Definition 23** *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 $(i,j) \in E(K_A) \backslash E(G^*[A])$, 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 $(i,j) \in E(K_A) \backslash E(G^*[A])$, if there exists a path from $i$ to $j$ in $G^*$ such that all nodes in the path, except $i$ and $j$, are in $V \backslash A$, then $(i,j) \in G^{*,m}[A]$.*

The graph $K_A$ is the complete graph over the vertices $A$. The first condition in Definition 23 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 13(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 $A \backslash \{4,5\}$. Similarly, the marginal graph over $A = \{4,5,6,7,8\}$ can be constructed as in Figure 13(c). The importance of marginal graphs is highlighted in the following proposition.

**Proposition 24** *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 $G^{*,m}[A]$.*

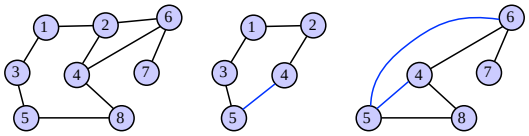



Figure 13: (a) A graph over eight nodes. (b) The marginal graph over $\{1, 2, 3, 4, 5\}$. (c) The marginal graph over $\{4, 5, 6, 7, 8\}$.

**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 $S \subseteq V \backslash \{i, j\}$. Thus, $G[A] \subset \check{G}_A$.
- For any edge $(i, j) \in K_A \backslash G[A]$, suppose that for every path from $i$ to $j$ contains at least one node from $A \backslash \{i, j\}$. Then, there exists a set of nodes $S \subseteq A \backslash \{i, j\}$ such that $X_i \perp\!\!\!\perp X_j | X_S$ and $(i, j) \notin \check{G}_A$.
- For any edge $(i, j) \in K_A \backslash G[A]$, suppose that there exists a path from $i$ to $j$ such that all nodes in the path, except $i$ and $j$, are in $V \backslash A$. 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 23, it is clear that $\check{G}_A = G^m[A]$. ■

Using Proposition 24, it is clear that if the UGMS algorithm $\Psi$ 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 23, 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 $\Sigma^*$. Such distributions are referred to as Gaussian graphical models. It is well known (Speed and Kiiveri, 1986) that 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 $\widehat{S}_A$ over a set of vertices $A \subset V$ as follows:

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

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

$$\widehat{\Theta} = \arg \max_{\Theta \succ 0, \Theta_{ij}=0 \; \forall \; (i,j) \notin H^m\left[\overline{R}\right]} \left\{ \log\det(\Theta) - \text{trace}\left(\widehat{S}_{\overline{R}}\,\Theta\right) - \lambda \sum_{(i,j)\in H'_R} \Theta_{ij} \right\}, \quad (9)$$

$$\widehat{E}_R = \left\{(i,j) \in H'_A : \widehat{\Theta}_{ij} \neq 0\right\}.$$

The graph $H^m[\overline{R}]$ is the marginal graph over $\overline{R}$ (see Appendix A). When $\overline{R} = V$, $H = K_V$, and $H'_A = K_V$, the above equations recover the standard gLasso estimator, which was first proposed in Banerjee et al. (2008). Equation (9) can be solved using algorithms in Yuan and Lin (2007), Banerjee et al. (2008), Scheinberg et al. (2010) and Hsieh et al. (2011). Theoretical properties of the estimates $\widehat{\Theta}$ and $\widehat{E}_R$ have been studied in Ravikumar et al. (2011). Note that the regularization parameter in (9) controls the sparsity of $\widehat{E}_R$. A larger $\lambda$ corresponds to a sparser solution. Further, we only regularize the terms in $\Theta_{ij}$ corresponding to the edges that need to be estimated, that is, the edges in $H'_R$. Finally, Equation (9) also accounts for the edges $H$ by computing the marginal graph over $\overline{R}$. In general, $H^m\left[\overline{R}\right]$ can be replaced by any graph that is superset of $H^m\left[\overline{R}\right]$.

### B.2 Neighborhood Selection (nLasso)

Using the local Markov property of undirected graphical models (see Definition 1), we know that if $P_X$ is Markov on $G^*$, then $P\left(X_i \,|\, X_{V\setminus i}\right) = P\left(X_i \,|\, X_{ne_{G^*}(i)}\right)$. 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 (Tibshirani, 1996) at each node (Meinshausen and Bühlmann, 2006). Recall that we are interested in estimating all edges in $H'_R$ by applying a UGMS algorithm to $\overline{R}$. The neighborhood selection using Lasso (nLasso) algorithm is given as follows:

$$H'' = K_{\overline{R}} \backslash H^m\left[\overline{R}\right],$$

$$\widehat{\beta}^k = \arg \min_{\beta_i = 0, i \in ne_{H''}(k) \cup k \cup V\setminus A} \left\{ \|\mathfrak{X}_k^n - \mathfrak{X}^n \beta\|_2^2 + \lambda \sum_{i \in ne_{H'_R}(k)} |\beta_i| \right\}, \quad (10)$$

$$\widehat{ne}^k = \left\{ i : \widehat{\beta}_i^k \neq 0 \right\},$$

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

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, that is, $(i,j)$ 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 Meinshausen and Bühlmann (2006) and Wainwright (2009). 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 (10). References Bresler et al. (2008), Netrapalli et al. (2010) and Ravikumar et al. (2010) extend the neighborhood selection based method to discrete valued graphical models.

## Appendix C. Proof of Proposition 8

We first prove the following result.

**Lemma 25** *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 set of vertices $\bar{S}$ not in $\overline{R}$. Then, there will exist a separator $S$ in the junction tree such that $S$ separates $\{i,j\}$ and $\bar{S}$. Thus, all paths in $H$ from $i$ and $j$ to $\bar{S}$ 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 about the existence of a non-direct path from $i$ to $j$ that passes through the set $\bar{S}$ not in $\overline{R}$.

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. ■

We now prove Proposition 8.

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 Lemma 25, 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 Lemma 25 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 the analysis of Algorithm 4 using results from Anandkumar et al. (2012a) and Kalisch and Bühlmann (2007). The analysis presented here is for the non-junction tree based algorithm. Throughout this section, assume

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

where $K_V$ is the complete graph over the vertices $V$. Further, let the threshold for the conditional independence test in (4) be $\lambda_n$. We are interested in finding conditions under which $\widehat{G} = G^*$ with high probability.

**Theorem 26** *Under Assumptions (A1)-(A5), there exists a conditional independence test such that if*

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

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

We now prove Theorem 26. Define the set $B_\eta$ as follows:

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

The following concentration inequality follows from Anandkumar et al. (2012a).

**Lemma 27** *Under Assumption (A4), there exists constants* $c_1$ *and* $c_2$ *such that for* $\epsilon < M$,

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

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

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, that is, $\widehat{\rho}_{ij|S} \leq \lambda_n \implies X_i \perp\!\!\!\perp X_j | X_S$. 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

$$\begin{aligned}
P_e &\leq P(\mathcal{E}_1) + P(\mathcal{E}_2)\,,\\
P(\mathcal{E}_1) &= P\left(\bigcup_{(i,j)\notin G} \{\exists\ S \text{ s.t. } |\widehat{\rho}_{ij|S}| > \lambda_n\}\right)\,,\\
P(\mathcal{E}_2) &= P\left(\bigcup_{(i,j)\in G} \{\exists\ S \text{ s.t. } |\widehat{\rho}_{ij|S}| \leq \lambda_n\}\right)\,.
\end{aligned}$$

We will find conditions under which $P(\mathcal{E}_1) \to 0$ and $P(\mathcal{E}_2) \to 0$ which will imply that $P_e \to 0$. The term $P(\mathcal{E}_1)$, the probability of including an edge in $\widehat{G}$ that does not belong to

the true graph, can be upper bounded as follows:

$$\begin{aligned}
P(\mathcal{E}_1) &\leq P\left(\bigcup_{(i,j)\notin G}\{\exists\ S \text{ s.t. } |\widehat{\rho}_{ij|S}| > \lambda_n\}\right) \leq P\left(\bigcup_{(i,j)\notin G, S\subset V\setminus\{i,j\}}\{|\widehat{\rho}_{ij|S}| > \lambda_n\}\right), \\
&\leq p^{\eta+2}\sup_{(i,j,S)\in B_\eta} P\left(|\widehat{\rho}_{ij|S}| > \lambda_n\right), \\
&\leq c_1 p^{\eta+2}\exp\left(-c_2(n-\eta)\lambda_n^2\right) = c_1\exp\left((\eta+2)\log(p) - c_2(n-\eta)\lambda_n^2\right).
\end{aligned}$$

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 $p^\eta$ number of separators for each edge. Choosing $\lambda_n$ such that

$$\lim_{n,p\to\infty}\frac{(n-\eta)\lambda_n^2}{(\eta+2)\log(p)} = \infty, \tag{11}$$

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

$$\lambda_n < c_3\rho_{min}. \tag{12}$$

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

$$\begin{aligned}
P(\mathcal{E}_2) &\leq P\left(\bigcup_{(i,j)\in G}\{\exists\ S \text{ s.t. } |\widehat{\rho}_{ij|S}| \leq \lambda_n\}\right), \\
&\leq P\left(\bigcup_{(i,j)\in G, S\subset V\setminus\{i,j\}} |\rho_{ij|S}| - |\widehat{\rho}_{ij|S}| > |\rho_{ij|S}| - \lambda_n\right), \\
&\leq p^{\eta+2}\sup_{(i,j,S)\in B_\eta} P\left(|\rho_{ij|S}| - |\widehat{\rho}_{ij|S}| > |\rho_{ij|S}| - \lambda_n\right), \\
&\leq p^{\eta+2}\sup_{(i,j,S)\in B_\eta} P\left(||\rho_{ij|S}| - |\widehat{\rho}_{ij|S}|| > \rho_{min} - \lambda_n\right), \\
&\leq c_1 p^{\eta+2}\exp\left(-c_2(n-\eta)(\rho_{min}-\lambda_n)^2\right) = c_1\exp\left((\eta+2)\log(p) - c_4(n-\eta)\rho_{min}^2\right).
\end{aligned} \tag{13}$$

To get (13), we use (12) so that $(\rho_{min}-\lambda_n) > (1-c_3)\rho_{min}$. For some constant $c_5 > 0$, suppose that for all $n > n'$ and $p > p'$,

$$c_4(n-\eta)\rho_{min}^2 > (\eta+2+c_5)\log(p). \tag{14}$$

Given (14), $P(\mathcal{E}_2)\to 0$ as $n,p\to\infty$. In asymptotic notation, we can write (14) as

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

which proves the Theorem. The conditional independence test is such that $\lambda_n$ is chosen to satisfy (11) and (12). In asymptotic notation, we can show that $\lambda_n = O(\rho_{min})$ and $\lambda_n^2 = \Omega\left(\eta\log(p)/n\right)$ satisfies (11) and (12).

## Appendix E. Proof of Theorem 9

To prove the theorem, it is sufficient to establish that

$$\rho_0 = \Omega\left(\sqrt{\eta_T \log(p)/n}\right), \tag{15}$$

$$\rho_1 = \Omega\left(\sqrt{\eta \log(p_1)/n}\right), \tag{16}$$

$$\rho_2 = \Omega\left(\sqrt{\eta \log(p_2)/n}\right), \tag{17}$$

$$\rho_T = \Omega\left(\sqrt{\eta \log(p_T)/n}\right). \tag{18}$$

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 $|\widehat{\rho}_{ij|S}| \leq \lambda_n^0$. Using the proof of Theorem 26 (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 $P(E(G^*) \not\subset E(H)) \to 0$ as $n \to 0$. Further, the threshold is chosen such that $\lambda_n^0 = O(\rho_0)$ and $(\lambda_n^0)^2 = \Omega\left(\eta_T \log(p)/n\right)$. This proves (15).

In Step 2, we estimate the graphs $\widehat{G}_1$ and $\widehat{G}_2$ by applying the PC-Algorithm to the vertices $V_1 \cup T$ and $V_2 \cup T$, respectively. For $\widehat{G}_1$, given that all edges that have a separator of size $\eta_T$ have been removed, we can again use the analysis in the proof of Theorem 26 to show that for $\lambda_n^1 = O(\rho_1)$ and $(\lambda_n^1)^2 = \Omega\left(\eta \log(p_1)/n\right)$, $n = \Omega(\rho_1^{-2}\eta \log(p_1))$ is sufficient for $P(\widehat{G}_1 \neq G^*[V_1 \cup T] \backslash K_T) | G^* \subset H) \to 0$ as $n \to \infty$. This proves (16). Using similar analysis, we can prove (17) and (18).

The probability of error can be written as

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

Given (15)-(18), each term on the right goes to 0 as $n \to \infty$, so $P_e \to 0$ as $n \to \infty$.

## References

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, 2012a.

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, 2012b.

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

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.

F. R. Bach and M. I. Jordan. Thin junction trees. In *Advances in Neural Information Processing Systems (NIPS)*, pages 569–576. MIT Press, 2001.

O. Banerjee, L. E. Ghaoui, and A. d'Aspremont. Model selection through sparse maximum likelihood estimation for multivariate Gaussian or binary data. *Journal of Machine Learning Research*, 9:485–516, June 2008.

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.

G. Bresler, E. Mossel, and A. Sly. Reconstruction of Markov random fields from samples: Some observations and algorithms. In Ashish Goel, Klaus Jansen, Jos Rolim, and Ronitt Rubinfeld, editors, *Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques*, volume 5171 of *Lecture Notes in Computer Science*, pages 343–356. Springer Berlin, 2008.

F. Bromberg, D. Margaritis, and V. Honavar. Efficient Markov network structure discovery using independence tests. *Journal of Artificial Intelligence Research (JAIR)*, 35:449–484, 2009.

T. Cai, W. Liu, and X. Luo. A constrained $\ell_1$ minimization approach to sparse precision matrix estimation. *Journal of the American Statistical Association*, 106(494):594–607, 2011.

V. Chandrasekaran, P.A. Parrilo, and A.S. Willsky. Latent variable graphical model selection via convex optimization. *Annals of Statistics*, 40(4):1935–1967, 2012.

A. Chechetka and C. Guestrin. Efficient principled learning of thin junction trees. In *Advances in Neural Information Processing Systems (NIPS)*, pages 273–280, December 2007.

J. Chen and Z. Chen. Extended Bayesian information criteria for model selection with large model spaces. *Biometrika*, 95(3):759–771, 2008.

M. J. Choi, V. Y. F. Tan, A. Anandkumar, and A. S. Willsky. Learning latent tree graphical models. *Journal of Machine Learning Research*, 12:1771–1812, May 2011.

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

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.

R. Foygel and M. Drton. Extended Bayesian information criteria for Gaussian graphical models. In *Advances in Neural Information Processing Systems (NIPS)*, pages 604–612, 2010.

J. Friedman, T. Hastie, and R. Tibshirani. Sparse inverse covariance estimation with the graphical Lasso. *Biostatistics*, 9(3):432–441, July 2008.

P. Giudici and P. J. Green. Decomposable graphical gaussian model determination. *Biometrika*, 86(4):785–801, 1999.

W. Hoeffding. A non-parametric test of independence. *The Annals of Mathematical Statistics*, 19(4):546–557, 1948.

C. Hsieh, M. A. Sustik, I. S. Dhillon, and P. Ravikumar. Sparse inverse covariance matrix estimation using quadratic approximation. In *Advances in Neural Information Processing Systems 24*, pages 2330–2338, 2011.

A. Jalali, C. Johnson, and P. Ravikumar. On learning discrete graphical models using greedy methods. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1935–1943, 2011.

C. C. Johnson, A. Jalali, and P. Ravikumar. High-dimensional sparse inverse covariance estimation using greedy methods. *Journal of Machine Learning Research - Proceedings Track*, 22:574–582, 2012.

M. Kalisch and P. Bühlmann. Estimating high-dimensional directed acyclic graphs with the PC algorithm. *Journal of Machine Learning Research*, 8:613–636, 2007.

D. Karger and N. Srebro. Learning Markov networks: Maximum bounded tree-width graphs. In *Proceedings of the twelfth annual ACM-SIAM symposium on Discrete algorithms*, pages 392–401, 2001.

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.

D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. The MIT Press, 2009.

K. S. S. Kumar and F. Bach. Convex relaxations for learning bounded-treewidth decomposable graphs. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2013.

J. Lafferty, H. Liu, and L. Wasserman. Sparse nonparametric graphical models. *Statistical Science*, 27(4):519–537, 2012.

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.

S. L. Lauritzen. *Graphical Models*. Oxford University Press, USA, 1996.

L. Li and K. C. Toh. An inexact interior point method for $\ell_1$-regularized sparse covariance selection. *Mathematical Programming Computation*, 2(3):291–315, 2010.

H. Liu, K. Roeder, and L. Wasserman. Stability approach to regularization selection (stars) for high dimensional graphical models. In *Advances in Neural Information Processing Systems (NIPS)*, 2010.

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, 2012a.

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.

H. Liu, F. Han, and C. Zhang. Transelliptical graphical models. In *Advances in Neural Information Processing Systems (NIPS)*, pages 809–817, 2012b.

P. Loh and M. J. Wainwright. Structure estimation for discrete graphical models: Generalized covariance matrices and their inverses. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2096–2104, 2012.

Z. Ma, X. Xie, and Z. Geng. Structural learning of chain graphs via decomposition. *Journal of Machine Learning Research*, 9:2847–2880, December 2008.

D. M. Malioutov, J. K. Johnson, and A. S. Willsky. Walk-sums and belief propagation in gaussian graphical models. *The Journal of Machine Learning Research*, 7:2031–2064, 2006.

F. M. Malvestuto. Approximating discrete probability distributions with decomposable models. *IEEE Transactions on Systems, Man and Cybernetics*, 21(5):1287–1294, 1991.

R. Mazumder and T. Hastie. Exact covariance thresholding into connected components for large-scale graphical lasso. *Journal of Machine Learning Research*, 13:781–794, March 2012. ISSN 1532-4435.

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

N. Meinshausen and P. Bühlmann. High-dimensional graphs and variable selection with the lasso. *Annals of Statistics*, 34(3):1436–1462, 2006.

P. Netrapalli, S. Banerjee, S. Sanghavi, and S. Shakkottai. Greedy learning of Markov network structure. In *2010 48th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 1295–1302, 2010.

A. Dobra, C. Hans, B. Jones, J. R. Nevins, and G. Yao, and M. West. Sparse graphical models for exploring gene expression data. *Journal of Multivariate Analysis*, 90:196–212, 2004.

M. J Rasch, A. Gretton, Y. Murayama, W. Maass, N. K 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.

P. Ravikumar, M. J. Wainwright, and J. Lafferty. High-dimensional Ising model selection using $\ell_1$-regularized logistic regression. *Annals of Statistics*, 38(3):1287–1319, 2010.

P. Ravikumar, M. J. Wainwright, G. Raskutti, and B. Yu. High-dimensional covariance estimation by minimizing $\ell_1$-penalized log-determinant divergence. *Electronic Journal of Statistics*, 5:935–980, 2011.

N. Robertson and P. D. Seymour. Graph minors. II. Algorithmic aspects of tree-width. *Journal of Algorithms*, 7(3):309 – 322, 1986.

K. Scheinberg, S. Ma, and D. Goldfarb. Sparse inverse covariance selection via alternating linearization methods. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2101–2109, 2010.

T. P. Speed and H. T. Kiiveri. Gaussian Markov distributions over finite graphs. *The Annals of Statistics*, 14(1):138–150, 1986.

P. Spirtes and C. Glymour. An algorithm for fast recovery of sparse causal graphs. *Social Science Computer Review*, 9:62–72, 1991.

P. Spirtes, C. Glymour, and R. Scheines. Causality from probability. In *Advanced Computing for the Social Sciences*, 1990.

R. Tibshirani. Regression shrinkage and selection via the Lasso. *Journal of the Royal Statistical Society, Series B*, 58(1):267–288, 1996.

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.

D. Vats. High-dimensional screening using multiple grouping of variables. *IEEE Transactions On Signal Processing*, to appear.

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.

M. J. Wainwright. *Stochastic Processes on Graphs: Geometric and Variational Approaches*. PhD thesis, Department of EECS, Massachusetts Institute of Technology, 2002.

M. J. Wainwright. Sharp thresholds for high-dimensional and noisy sparsity recovery using $\ell_1$-constrained quadratic programming (Lasso). *IEEE Transactions on Information Theory*, 55(5):2183–2202, 2009. ISSN 0018-9448.

W. Wang, M. J. Wainwright, and K. Ramchandran. Information-theoretic bounds on model selection for Gaussian Markov random fields. In *IEEE International Symposium on Information Theory (ISIT)*, 2010.

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.

D. M. Witten, J. H. Friedman, and N. Simon. New insights and faster computations for the graphical lasso. *Journal of Computational and Graphical Statistics*, 20(4):892–900, 2011.

X. Xie and Z. Geng. A recursive method for structural learning of directed acyclic graphs. *Journal of Machine Learning Research*, 9:459–483, 2008.

L. Xue and H. Zou. Regularized rank-based estimation of high-dimensional nonparanormal graphical models. *The Annals of Statistics*, 40(5):2541–2571, 2012.

E. Yang, G. Allen, Z. Liu, and P. Ravikumar. Graphical models via generalized linear models. In *Advances in Neural Information Processing Systems*, pages 1367–1375, 2012.

J. S. Yedidia, W. T. Freeman, and Y. Weiss. Constructing free-energy approximations and generalized belief propagation algorithms. *IEEE Transactions on Information Theory*, 51(7):2282–2312, 2005.

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

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.

H. Zou. The adaptive lasso and its oracle properties. *Journal of the American Statistical Association*, 101(476):1418–1429, 2006.