Published in final edited form as:
J Mach Learn Res. 2014 January 1; 15(1): 445–488.

Node-Based Learning of Multiple Gaussian Graphical Models

Karthik Mohan,
Department of Electrical Engineering, University of Washington, Seattle WA, 98195

Palma London,
Department of Electrical Engineering, University of Washington, Seattle WA, 98195

Maryam Fazel,
Department of Electrical Engineering, University of Washington, Seattle WA, 98195

Daniela Witten, and
Department of Biostatistics, University of Washingtonm, Seattle WA, 98195

Su-In Lee
Departments of Computer Science and Engineering, Genome Sciences, University of Washington, Seattle WA, 98195

Karthik Mohan: karna@uw.edu; Palma London: palondon@uw.edu; Maryam Fazel: mfazel@uw.edu; Daniela Witten: dwitten@uw.edu; Su-In Lee: suinlee@cs.washington.edu

Abstract

We consider the problem of estimating high-dimensional Gaussian graphical models corresponding to a single set of variables under several distinct conditions. This problem is motivated by the task of recovering transcriptional regulatory networks on the basis of gene expression data containing heterogeneous samples, such as different disease states, multiple species, or different developmental stages. We assume that most aspects of the conditional dependence networks are shared, but that there are some structured differences between them. Rather than assuming that similarities and differences between networks are driven by individual edges, we take a *node-based* approach, which in many cases provides a more intuitive interpretation of the network differences. We consider estimation under two distinct assumptions: (1) differences between the K networks are due to individual nodes that are *perturbed* across conditions, or (2) similarities among the K networks are due to the presence of *common hub nodes* that are shared across all K networks. Using a *row-column overlap norm* penalty function, we formulate two convex optimization problems that correspond to these two assumptions. We solve these problems using an alternating direction method of multipliers algorithm, and we derive a set of necessary and sufficient conditions that allows us to decompose the problem into independent subproblems so that our algorithm can be scaled to high-dimensional settings. Our proposal is illustrated on synthetic data, a webpage data set, and a brain cancer gene expression data set.

Keywords

graphical model; structured sparsity; alternating direction method of multipliers; gene regulatory network; lasso; multivariate normal

1. Introduction

Graphical models encode the conditional dependence relationships among a set of p variables (Lauritzen, 1996). They are a tool of growing importance in a number of fields, including finance, biology, and computer vision. A graphical model is often referred to as a conditional dependence *network*, or simply as a *network*. Motivated by network terminology, we can refer to the p variables in a graphical model as *nodes*. If a pair of variables are conditionally dependent, then there is an *edge* between the corresponding pair of nodes; otherwise, no edge is present.

Suppose that we have n observations that are independently drawn from a multivariate normal distribution with covariance matrix Σ. Then the corresponding *Gaussian graphical model* (GGM) that describes the conditional dependence relationships among the variables is encoded by the sparsity pattern of the inverse covariance matrix, Σ^{-1} (see, e.g., Mardia et al., 1979; Lauritzen, 1996). That is, the jth and j' th variables are conditionally independent if and only if $(\Sigma^{-1})_{jj'} = 0$. Unfortunately, when $p > n$, obtaining an accurate estimate of Σ^{-1} is challenging. In such a scenario, we can use prior information—such as the knowledge that many of the pairs of variables are conditionally independent—in order to more accurately estimate Σ^{-1} (see, e.g., Yuan and Lin, 2007a; Friedman et al., 2007; Banerjee et al., 2008).

In this paper, we consider the task of estimating K GGMs on a single set of p variables under the assumption that the GGMs are similar, with certain structured differences. As a motivating example, suppose that we have access to gene expression measurements for n_1 lung cancer samples and n_2 normal lung samples, and that we would like to estimate the gene regulatory networks underlying the normal and cancer lung tissue. We can model each of these regulatory networks using a GGM. We have two obvious options.

1. We can estimate a single network on the basis of all $n_1 + n_2$ tissue samples. But this approach overlooks fundamental differences between the true lung cancer and normal gene regulatory networks.
2. We can estimate separate networks based on then_1 cancer and n_2 normal samples. However, this approach fails to exploit substantial commonality of the two networks, such as lung-specific pathways.

In order to effectively make use of the available data, we need a principled approach for jointly estimating the two networks in such a way that the two estimates are encouraged to be quite similar to each other, while allowing for certain structured differences. In fact, these differences may be of scientific interest.

Another example of estimating multiple GGMs arises in the analysis of the conditional dependence relationships among p stocks at two distinct points in time. We might be interested in detecting stocks that have differential connectivity with all other stocks across the two time points, as these likely correspond to companies that have undergone significant changes. Yet another example occurs in the field of neuroscience, in which it is of interest to learn how the connectivity of neurons changes over time.

Past work on joint estimation of multiple GGMs has assumed that individual *edges* are shared or differ across conditions (see, e.g., Kolar et al., 2010; Zhang and Wang, 2010; Guo et al., 2011; Danaher et al., 2013); here we refer to such approaches as *edge-based*. In this paper, we instead take a *node-based* approach: we seek to estimate K GGMs under the assumption that similarities and differences between networks are driven by individual *nodes* whose patterns of connectivity to other nodes are shared across networks, or differ between networks. As we will see, node-based learning is more powerful than edge-based learning, since it more fully exploits our prior assumptions about the similarities and differences between networks.

More specifically, in this paper we consider two types of shared network structure.

1. Certain nodes serve as highly-connected *hub* nodes. We assume that the same nodes serve as hubs in each of the K networks. Figure 1 illustrates a toy example of this setting, with $p = 5$ nodes and $K = 2$ networks. In this example, the second variable, X_2, serves as a hub node in each network. In the context of transcriptional regulatory networks, X_2 might represent a gene that encodes a *transcription factor* that regulates a large number of downstream genes in all K contexts. We propose the *common hub* (*co-hub*) *node joint graphical lasso* (CNJGL), a convex optimization problem for estimating GGMs in this setting.
2. The networks differ due to particular nodes that are*perturbed* across conditions, and therefore have a completely different connectivity pattern to other nodes in the K networks. Figure 2 displays a toy example, with $p = 5$ nodes and $K = 2$ networks; here we see that all of the network differences are driven by perturbation in the second variable, X_2. In the context of transcriptional regulatory networks, X_2 might represent a gene that is mutated in a particular condition, effectively disrupting its conditional dependence relationships with other genes. We propose the *perturbed-node joint graphical lasso* (PNJGL), a convex optimization problem for estimating GGMs in this context.

Node-based learning of multiple GGMs is challenging, due to complications resulting from symmetry of the precision matrices. In this paper, we overcome this problem through the use of a new convex regularizer.

The rest of this paper is organized as follows. We introduce some relevant background material in Section 2. In Section 3, we present the *row-column overlap norm* (RCON), a regularizer that encourages a matrix to have a support that is the *union of a set of rows and columns*. We apply the RCON penalty to a pair of inverse covariance matrices, or to the difference between a pair of inverse covariance matrices, in order to obtain the CNJGL and PNJGL formulations just described. In Section 4, we propose an *alternating direction method of multipliers* (ADMM) algorithm in order to solve these two convex formulations. In order to scale this algorithm to problems with many variables, in Section 5 we introduce a set of simple conditions on the regularization parameters that indicate that the problem can be broken down into many independent subproblems, leading to substantial algorithm speed-ups. In Section 6, we apply CNJGL and PNJGL to synthetic data, and in Section 7 we apply them to gene expression data and to webpage data. The Discussion is in Section 8. Proofs are in the Appendix.

A preliminary version of some of the ideas in this paper appear in Mohan et al. (2012). There the PNJGL formulation was proposed, along with an ADMM algorithm. Here we expand upon that formulation and present the CNJGL formulation, an ADMM algorithm for solving it, as well as comprehensive results on both real and simulated data. Furthermore, in this paper we discuss theoretical conditions for computational speed-ups, which are critical to application of both PNJGL and CNJGL to data sets with many variables.

2. Background on High-Dimensional GGM Estimation

In this section, we review the literature on learning Gaussian graphical models.

2.1 The Graphical Lasso for Estimating a Single GGM

As was mentioned in Section 1, estimating a single GGM on the basis of n independent and identically distributed observations from a $N_p(\mathbf{0}, \Sigma)$ distribution amounts to learning the sparsity structure of Σ^{-1} (Mardia et al., 1979; Lauritzen, 1996). When $n > p$, one can estimate Σ^{-1} by maximum likelihood. But in high dimensions when p is large relative to n, this is not possible because the empirical covariance matrix is singular. Consequently, a number of authors (among others, Yuan and Lin, 2007a; Friedman et al., 2007; Ravikumar et al., 2008; Banerjee et al., 2008; Scheinberg et al., 2010; Hsieh et al., 2011) have considered maximizing the penalized log likelihood

$$\underset{\Theta \in \mathbb{S}_{++}^{p}}{\text{maximize}} \{\log\det\Theta - \text{trace}(S\Theta) - \lambda \|\Theta\|_1\}, \quad (1)$$

where S is the empirical covariance matrix, λ is a nonnegative tuning parameter, $\mathbb{S}_{++}^{p}$ denotes the set of positive definite matrices of size p, and $\|\Theta\|_1 = \Sigma_{i,j}|\Theta_{ij}|$. The solution to (1) serves as an estimate of Σ^{-1}, and a zero element in the solution corresponds to a pair of variables that are estimated to be conditionally independent. Due to the ℓ_1 penalty (Tibshirani, 1996) in (1), this estimate will be positive definite for any $\lambda > 0$, and sparse when λ is sufficiently large. We refer to (1) as the *graphical lasso*. Problem (1) is convex, and efficient algorithms for solving it are available (among others, Friedman et al., 2007; Banerjee et al., 2008; Rothman et al., 2008; D'Aspremont et al., 2008; Scheinberg et al., 2010; Witten et al., 2011).

2.2 The Joint Graphical Lasso for Estimating Multiple GGMs

Several formulations have recently been proposed for extending the graphical lasso (1) to the setting in which one has access to a number of observations from K distinct conditions, each with measurements on the same set of p variables. The goal is to estimate a graphical model for each condition under the assumption that the K networks share certain characteristics but are allowed to differ in certain structured ways. Guo et al. (2011) take a non-convex approach to solving this problem. Zhang and Wang (2010) take a convex approach, but use a least squares loss function rather than the negative Gaussian log likelihood. Here we review the convex formulation of Danaher et al. (2013), which forms the starting point for the proposal in this paper.

Suppose that $X_1^k, \ldots, X_{n_k}^k \in \mathbb{R}^p$ are independent and identically distributed from a $N_p(\mathbf{0}, \Sigma^k)$ distribution, for $k = 1, \ldots, K$. Here n_k is the number of observations in the kth condition, or class. Letting S^k denote the empirical covariance matrix for the kth class, we can maximize the penalized log likelihood

$$\underset{\Theta^1 \in \mathbb{S}_{++}^p, \ldots, \Theta^K \in \mathbb{S}_{++}^p}{\text{maximize}} \left\{ L(\Theta^1, \ldots, \Theta^K) - \lambda_1 \sum_{k=1}^{K} \|\Theta^k\|_1 - \lambda_2 \sum_{i \neq j} P(\Theta_{ij}^1, \ldots, \Theta_{ij}^K) \right\}, \quad (2)$$

where $L(\Theta^1, \ldots, \Theta^K) = \sum_{k=1}^{K} n_k \left(\log\det \Theta^k - \mathrm{trace}(S^k \Theta^k) \right)$, λ_1 and λ_2 are nonnegative tuning parameters, and $P(\Theta_{ij}^1, \ldots, \Theta_{ij}^K)$ is a convex penalty function applied to each off diagonal element of $\Theta^1, \ldots, \Theta^K$ in order to encourage similarity among them. Then the $\hat{\Theta^1}, \ldots, \hat{\Theta^K}$ that solve (2) serve as estimates for $(\Sigma^1)^{-1}, \ldots, (\Sigma^K)^{-1}$. Danaher et al. (2013) refer to (2) as the *joint graphical lasso* (JGL). In particular, they consider the use of a *fused lasso penalty* (Tibshirani et al., 2005),

$$P(\Theta_{ij}^1, \ldots, \Theta_{ij}^K) = \sum_{k < k'} |\Theta_{ij}^k - \Theta_{ij}^{k'}|, \quad (3)$$

on the differences between pairs of network edges, as well as a *group lasso penalty* (Yuan and Lin, 2007b),

$$P(\Theta_{ij}^1, \Theta_{ij}^2, \ldots, \Theta_{ij}^K) = \sqrt{\sum_{k=1}^{K} (\Theta_{ij}^k)^2}, \quad (4)$$

on the edges themselves. Danaher et al. (2013) refer to problem (2) combined with (3) as the *fused graphical lasso* (FGL), and to (2) combined with (4) as the *group graphical lasso* (GGL).

FGL encourages the K network estimates to have identical edge values, whereas GGL encourages the K network estimates to have a shared pattern of sparsity. Both the FGL and GGL optimization problems are convex. An approach related to FGL and GGL is proposed in Hara and Washio (2013).

Because FGL and GGL borrow strength across all available observations in estimating each network, they can lead to much more accurate inference than simply learning each of the K networks separately.

But both FGL and GGL take an *edge-based* approach: they assume that differences between and similarities among the networks arise from individual edges. In this paper, we propose a *node-based* formulation that allows for more powerful estimation of multiple GGMs, under the assumption that network similarities and differences arise from *nodes* whose connectivity patterns to other nodes are shared or disrupted across conditions.

3. Node-Based Joint Graphical Lasso

In this section, we first discuss the failure of a naive approach for node-based learning of multiple GGMs. We then present a norm that will play a critical role in our formulations for this task. Finally, we discuss two approaches for node-based learning of multiple GGMs.

3.1 Why is Node-Based Learning Challenging?

At first glance, node-based learning of multiple GGMs seems straightforward. For instance, consider the task of estimating $K = 2$ networks under the assumption that the connectivity patterns of individual nodes differ across the networks. It seems that we could simply modify (2) combined with (3) as follows,

$$\underset{\Theta^1 \in \mathbb{S}^p_{++}, \Theta^2 \in \mathbb{S}^p_{++}}{\text{maximize}} \left\{ L(\Theta^1, \Theta^2) - \lambda_1 \|\Theta^1\|_1 - \lambda_1 \|\Theta^2\|_1 - \lambda_2 \sum_{j=1}^{P} \|\Theta^1_j - \Theta^2_j\|_2 \right\}, \quad (5)$$

where Θ^k_j is the *j*th column of the matrix Θ^k. This amounts to applying a *group asso* (Yuan and Lin, 2007b) penalty to the columns of $\Theta^1 - \Theta^2$. Equation (5) seems to accomplish our goal of encouraging $\Theta^1_j = \Theta^2_j$. We will refer to this as the *naive group lasso* approach.

In (5), we have applied the group lasso using *p* groups; the *j*th group is the *j*th column of $\Theta^1 - \Theta^2$. Due to the symmetry of Θ^1 and Θ^2, there is substantial overlap among the *p* groups: the (i, j)th element of $\Theta^1 - \Theta^2$ is contained in both the *i*th and *j*th groups. In the presence of overlapping groups, the group lasso penalty yields estimates whose *support is the complement of the union of groups* (Jacob et al., 2009; Obozinski et al., 2011). Figure 3(a) displays a simple example of the results obtained if we attempt to estimate $(\Sigma^1)^{-1} - (\Sigma^2)^{-1}$ using (5). The figure reveals that (5) cannot be used to detect node perturbation.

A naive approach to co-hub detection is challenging for a similar reason. Recall that the *j*th node is a co-hub if the *j*th columns of both Θ^1 and Θ^2 contain predominantly non-zero elements, and let diag(Θ) denote a matrix consisting of the diagonal elements of Θ. It is tempting to formulate the optimization problem

$$\underset{\Theta^1 \in \mathbb{S}^p_{++}, \Theta^2 \in \mathbb{S}^p_{++}}{\text{maximize}} \left\{ L(\Theta^1, \Theta^2) - \lambda_1 \|\Theta^1\|_1 - \lambda_1 \|\Theta^2\|_1 - \lambda_2 \sum_{j=1}^{p} \left\| \begin{bmatrix} \Theta^1 - \text{diag}(\Theta^1) \\ \Theta^2 - \text{diag}(\Theta^2) \end{bmatrix}_j \right\|_2 \right\},$$

where the group lasso penalty encourages the off-diagonal elements of many of the columns to be simultaneously zero in Θ^1 and Θ^2. Unfortunately, once again, the presence of overlapping groups encourages the support of the matrices Θ^1 and Θ^2 to be the intersection of a set of rows and columns, as in Figure 3(a), rather than the union of a set of rows and columns.

3.2 Row-Column Overlap Norm

Detection of perturbed nodes or co-hub nodes requires a penalty function that, when applied to a matrix, yields a support given by the union of a set of rows and columns. We now propose the *row-column overlap norm* (RCON) for this task.

Definition 1: The row-column overlap norm (RCON) induced by a matrix norm ||.|| is defined as

$$\Omega(\Theta^1, \Theta^2, \ldots, \Theta^K) = \min_{V^1, V^2, \ldots, V^K} \left\| \begin{bmatrix} V^1 \\ V^2 \\ \vdots \\ V^K \end{bmatrix} \right\| \quad \text{subject to} \quad \Theta^k = V^k + (V^k)^T \text{ for } k = 1, \ldots, K.$$

It is easy to check that Ω is indeed a norm for all matrix norms ||.||. Also, when ||.|| is symmetric in its argument, that is, $\|V\| = \|V^T\|$, then

$$\Omega(\Theta^1, \Theta^2, \ldots, \Theta^K) = \frac{1}{2} \left\| \begin{bmatrix} \Theta^1 \\ \Theta^2 \\ \vdots \\ \Theta^K \end{bmatrix} \right\|.$$

Thus if $\|\cdot\|$ is an ℓ_1/ℓ_1 norm, then $\Omega(\Theta^1, \Theta^2, \ldots, \Theta^K) = \frac{1}{2}\sum_{k=1}^{K}\sum_{i,j}|\Theta_{ij}^k|$.

We now discuss the motivation behind Definition 1. Any symmetric matrix Θ^k can be (non-uniquely) decomposed as $V^k + (V^k)^T$; note that V^k need not be symmetric. This amounts to interpreting Θ^k as a set of columns (the columns of V^k) *plus* a set of rows (the columns of V^k, transposed). In this paper, we are interested in the particular case of RCON penalties where ||.|| is an ℓ_1/ℓ_q norm, given by $\|V\| = \sum_{j=1}^{p} \|V_j\|_q$, where $1 \leq q \leq \infty$. With a little abuse of notation, we will let Ω_q denote Ω when ||.|| is given by the ℓ_1/ℓ_q norm. Then Ω_q encourages $\Theta^1, \Theta^2, \ldots, \Theta^K$ to decompose into V^k and $(V^k)^T$ such that the summed ℓ_q norms of all of the columns (concatenated over $V^1, \ldots, V^K$) is small. This encourages structures of interest on the columns *and* rows of $\Theta^1, \Theta^2, \ldots, \Theta^K$.

To illustrate this point, in Figure 3 we display schematic results obtained from estimating a 5 × 5 matrix subject to the RCON penalty Ω_q, for $q = 1, 2$, and ∞. We see from Figure 3(b) that when $q = 1$, the RCON penalty yields a matrix estimate with unstructured sparsity; recall that Ω_1 amounts to an ℓ_1 penalty applied to the matrix entries. When $q = 2$ or $q = \infty$, we see from Figures 3(c)–(d) that the RCON penalty yields a sparse matrix estimate for which the non-zero elements are a set of rows *plus* a set of columns|that is, the union of a set of rows and columns.

We note that Ω_2 can be derived from the *overlap norm* (Obozinski et al., 2011; Jacob et al., 2009) applied to groups given by rows and columns of $\Theta^1, \ldots, \Theta^K$. Details are described in Appendix E. Additional properties of RCON are discussed in Appendix A.

3.3 Node-Based Approaches for Learning GGMs

We discuss two approaches for node-based learning of GGMs. The first promotes networks whose differences are attributable to perturbed nodes. The second encourages the networks to share co-hub nodes.

3.3.1 Perturbed-node Joint Graphical Lasso—Consider the task of jointly estimating K precision matrices by solving

$$\underset{\Theta^1,\Theta^2,\ldots,\Theta^K \in \mathbb{S}^p_{++}}{\text{maximize}} \left\{ L(\Theta^1, \Theta^2, \ldots, \Theta^K) - \lambda_1 \sum_{k=1}^{K} \|\Theta^k\|_1 - \lambda_2 \sum_{k<k'} \Omega_q(\Theta^k - \Theta^{k'}) \right\}. \quad (6)$$

We refer to the convex optimization problem (6) as the *perturbed-node joint graphical lasso* (PNJGL). Let $\hat{\Theta}^1, \hat{\Theta}^2, \ldots, \hat{\Theta}^K$ denote the solution to (6); these serve as estimates for $(\Sigma^1)^{-1}, \ldots, (\Sigma^K)^{-1}$. In (6), λ_1 and λ_2 are nonnegative tuning parameters, and $q \geq 1$. When $\lambda_2 = 0$, (6) amounts simply to applying the graphical lasso optimization problem (1) to each condition separately in order to separately estimate K networks. When $\lambda_2 > 0$, we are encouraging similarity among the K network estimates. When $q = 1$, we have the following observation.

Remark 2: The FGL formulation (Equations 2 and 3) is a special case of PNJGL (6) with q = 1.

In other words, when $q = 1$, (6) amounts to the *edge-based* approach of Danaher et al. (2013) that encourages many entries of $\hat{\Theta}^k \times \hat{\Theta}^{k'}$ to equal zero.

However, when $q = 2$ or $q = \infty$, then (6) amounts to a *node-based approach*: the support of $\hat{\Theta}^k - \hat{\Theta}^{k'}$ is encouraged to be a union of a few rows and the corresponding columns. These can be interpreted as a set of nodes that are perturbed across the conditions. An example of the sparsity structure detected by PNJGL with $q = 2$ or $q = \infty$ is shown in Figure 2.

3.3.2 Co-hub Node Joint Graphical Lasso—We now consider jointly estimating K precision matrices by solving the convex optimization problem

$$\underset{\Theta^1,\Theta^2,\ldots,\Theta^K \in \mathbb{S}^p_{++}}{\text{maximize}} \left\{ L(\Theta^1, \Theta^2, \ldots, \Theta^K) - \lambda_1 \sum_{k=1}^{K} \|\Theta^k\|_1 - \lambda_2 \Omega_q(\Theta^1 - \text{diag}(\Theta^1), \ldots, \Theta^K - \text{diag}(\Theta^K)) \right\}. \quad (7)$$

We refer to (7) as the *co-hub node joint graphical lasso* (CNJGL) formulation. In (7), λ_1 and λ_2 are nonnegative tuning parameters, and $q \geq 1$. When $\lambda_2 = 0$ then this amounts to a graphical lasso optimization problem applied to each network separately; however, when $\lambda_2 > 0$, a shared structure is encouraged among the K networks. In particular, (7) encourages network estimates that have a common set of hub nodes—that is, it encourages the supports of $\Theta^1, \Theta^2, \ldots, \Theta^K$ to be the same, and the union of a set of rows and columns.

CNJGL can be interpreted as a node-based extension of the GGL proposal (given in Equations 2 and 4, and originally proposed by Danaher et al., 2013). While GGL encourages the K networks to share a common edge support, CNJGL instead encourages the networks to share a common node support.

We now remark on an additional connection between CNJGL and the graphical lasso.

Remark 3: If q = 1, then CNJGL amounts to a modified graphical lasso on each network separately, with a penalty of λ_1 applied to the diagonal elements, and a penalty of $\lambda_1 + \lambda_2/2$ applied to the off-diagonal elements.

4. Algorithms

The PNJGL and CNJGL optimization problems (6, 7) are convex, and so can be directly solved in the modeling environment cvx (Grant and Boyd, 2010), which calls conic interior-point solvers such as SeDuMi or SDPT3. However, when applied to solve semi-definite programs, second-order methods such as the interior-point algorithm do not scale well with the problem size.

We next examine the use of existing first-order methods to solve (6) and (7). Several first-order algorithms have been proposed for minimizing a least squares objective with a group lasso penalty (as in Yuan and Lin, 2007b) in the presence of overlapping groups (Argyriou et al., 2011; Chen et al., 2011; Mosci et al., 2010). Unfortunately, those algorithms cannot be applied to the PNJGL and CNJGL formulations, which involve the RCON penalty rather than simply a standard group lasso with overlapping groups. The RCON penalty is a variant of the overlap norm proposed in Obozinski et al. (2011), and indeed those authors propose an algorithm for minimizing a least squares objective subject to the overlap norm. However, in the context of CNJGL and PNJGL, the objective of interest is a Gaussian log likelihood, and the algorithm of Obozinski et al. (2011) cannot be easily applied.

Another possible approach for solving (6) and (7) involves the use of a standard first-order method, such as a projected subgradient approach. Unfortunately, such an approach is not straightforward, since computing the subgradients of the RCON penalty involves solving a non-trivial optimization problem (to be discussed in detail in Appendix A). Similarly, a proximal gradient approach for solving (6) and (7) is challenging because the proximal operator of the combination of the overlap norm and the ℓ_1 norm has no closed form.

To overcome the challenges outlined above, we propose to solve the PNJGL and CNJGL problems using an *alternating direction method of multipliers* algorithm (ADMM; see, e.g., Boyd et al., 2010).

4.1 The ADMM Approach

Here we briefly outline the standard ADMM approach for a general optimization problem,

$$\begin{array}{ll} \underset{X}{\text{minimize}} & g(X)+h(X) \\ \text{subject to} & X \in \mathscr{X}. \end{array} \quad (8)$$

ADMM is attractive in cases where the proximal operator of *g*(*X*) + *h*(*X*) cannot be easily computed, but where the proximal operator of *g*(*X*) and the proximal operator of *h*(*X*) are easily obtained. The approach is as follows (Boyd et al., 2010; Eckstein and Bertsekas, 1992; Gabay and Mercier, 1976):

1. Rewrite the optimization problem (8) as

   $$\begin{array}{ll} \underset{X,Y}{\text{minimize}} & g(X)+h(Y) \\ \text{subject to} & X \in \mathscr{X},\ X=Y, \end{array} \quad (9)$$

   where here we have decoupled *g* and *h* by introducing a new optimization variable, *Y*.

2. Form the augmented Lagrangian to (9) by first forming the Lagrangian,

   $$L(X,Y,\Lambda)=g(X)+h(Y)+\langle \Lambda, X-Y\rangle,$$

   and then augmenting it by a quadratic function of $X - Y$,

   $$L_\rho(X,Y,\Lambda)=L(X,Y,\Lambda)+\frac{\rho}{2}\|X-Y\|_F^2,$$

   where ρ is a positive constant.

3. Iterate until convergence:

   a. Update each primal variable in turn by minimizing the augmented Lagrangian with respect to that variable, while keeping all other variables fixed. The updates in the *k*th iteration are as follows:

   $$\begin{array}{l} X^{k+1} \leftarrow \arg\underset{X\in\mathscr{X}}{\min} L_\rho(X,Y^k,\Lambda^k), \\ Y^{k+1} \leftarrow \arg\underset{Y}{\min} L_\rho(X^{k+1},Y,\Lambda^k). \end{array}$$

   b. Update the dual variable using a dual-ascent update,

   $$\Lambda^{k+1} \leftarrow \Lambda^k+\rho(X^{k+1}-Y^{k+1}).$$

The standard ADMM presented here involves minimization over two primal variables, *X* and *Y*. For our problems, we will use a similar algorithm but with *more than two primal variables*. More details about the algorithm and its convergence are discussed in Section 4.2.4.

4.2 ADMM Algorithms for PNJGL and CNJGL

Here we outline the ADMM algorithms for the PNJGL and CNJGL optimization problems; we refer the reader to Appendix F for detailed derivations of the update rules.

4.2.1 ADMM Algorithm for PNJGL—Here we consider solving PNJGL with $K = 2$; the extension for $K > 2$ is slightly more complicated. To begin, we note that (6) can be rewritten as

$$\begin{array}{cc}\underset{\Theta^1,\Theta^2\in\mathbb{S}^p_{++},V\in\mathbb{R}^{p\times p}}{\text{maximize}} & \left\{L(\Theta^1,\Theta^2)-\lambda_1\|\Theta^1\|_1-\lambda_1\|\Theta^2\|_1-\lambda_2\sum_{j=1}^p\|V_j\|_q\right\} \\ \text{subject to} & \Theta^1-\Theta^2=V+V^T.\end{array} \quad (10)$$

We now reformulate (10) by introducing new variables, so as to decouple some of the terms in the objective function that are difficult to optimize jointly:

$$\begin{array}{cc}\underset{\Theta^1\in S^p_{++},\Theta^2\in S^p_{++},Z^1 Z^2,V,W}{\text{minimize}} & \left\{-L(\Theta^1,\Theta^2)+\lambda_1\|Z^1\|_1+\lambda_1\|Z^2\|_1+\lambda_2\sum_{j=1}^p\|V_j\|_q\right\} \\ \text{subject to} & \Theta^1-\Theta^2=V+W, V=W^T, \Theta^1=Z^1, \Theta^2=Z^2.\end{array} \quad (11)$$

The augmented Lagrangian to (11) is given by

$$\begin{array}{c}-L(\Theta^1,\Theta^2)+\lambda_1\|Z^1\|_1+\lambda_1\|Z^2\|_1+\lambda_2\sum_{j=1}^p\|V_j\|_q+\langle F,\Theta^1-\Theta^2-(V+W)\rangle \\ +\langle G,V-W^T\rangle+\langle Q^1,\Theta^1-Z^1\rangle+\langle Q^2,\Theta^2-Z^2\rangle+\frac{\rho}{2}\|\Theta^1-\Theta^2-(V+W)\|_F^2 \\ +\frac{\rho}{2}\|V-W^T\|_F^2+\frac{\rho}{2}\|\Theta^1-Z^1\|_F^2+\frac{\rho}{2}\|\Theta^2-Z^2\|_F^2.\end{array} \quad (12)$$

In (12) there are six primal variables and four dual variables. Based on this augmented Lagrangian, the complete ADMM algorithm for (6) is given in Algorithm 1, in which the operator Expand is given by

$$\text{Expand}(A,\rho,n_k)=\underset{\Theta\in\mathbb{S}^p_{++}}{\text{argmin}}\{-n_k\text{logdet}(\Theta)+\rho\|\Theta-A\|_F^2\}=\frac{1}{2}U\left(D+\sqrt{D^2+\frac{2n_k}{\rho}I}\right)U^T,$$

where UDU^T is the eigenvalue decomposition of a symmetric matrix A, and as mentioned earlier, n_k is the number of observations in the kth class. The operator $\mathcal{T}_q$ is given by

$$\mathcal{T}_q(A,\lambda)=\underset{X}{\text{argmin}}\left\{\frac{1}{2}\|X-A\|_F^2+\lambda\sum_{j=1}^p\|X_j\|_q\right\},$$

and is also known as the proximal operator corresponding to the ℓ_1/ℓ_q norm. For $q = 1, 2, \infty$, $\mathcal{T}_q$ takes a simple form (see, e.g., Section 5 of Duchi and Singer, 2009).

Algorithm 1

ADMM algorithm for the PNJGL optimization problem (6)

```
input: $\rho > 0, \mu > 1, t_{\max} > 0$;
Initialize: Primal variables to the identity matrix and dual variables to the zero matrix;
for $t = 1{:}t_{\max}$ do
    $\rho \leftarrow \mu\rho$;
    while Not converged do
        $\Theta^1 \leftarrow \text{Expand}\left(\frac{1}{2}(\Theta^2 + V + W + Z^1) - \frac{1}{2\rho}(Q^1 + n_1 S^1 + F), \rho, n_1\right)$;
        $\Theta^2 \leftarrow \text{Expand}\left(\frac{1}{2}(\Theta^1 - (V + W) + Z^2) - \frac{1}{2\rho}(Q^2 + n_2 S^2 - F), \rho, n_2\right)$;
        $Z^i \leftarrow \mathcal{T}_1\left(\Theta^i + \frac{Q^i}{\rho}, \frac{\lambda_1}{\rho}\right)$ for $i = 1, 2$;
        $V \leftarrow \mathcal{T}_q\left(\frac{1}{2}(W^T - W + (\Theta^1 - \Theta^2)) + \frac{1}{2\rho}(F - G), \frac{\lambda_2}{2\rho}\right)$;
        $W \leftarrow \frac{1}{2}(V^T - V + (\Theta^1 - \Theta^2)) + \frac{1}{2\rho}(F + G^T)$;
        $F \leftarrow F + \rho(\Theta^1 - \Theta^2 - (V + W))$;
        $G \leftarrow G + \rho(V - W^T)$;
        $Q^i \leftarrow Q^i + \rho(\Theta^i - Z^i)$ for $i = 1, 2$
```

4.2.2 ADMM Algorithm for CNJGL—The CNJGL formulation in (7) is equivalent to

$$\begin{array}{ll} \underset{\Theta^i \in \mathbb{S}^p_{++}, V^i \in \mathbb{R}^{p\times p}, i=1\ldots K}{\text{minimize}} & -L(\Theta^1, \Theta^2, \ldots, \Theta^K) + \lambda_1 \sum_{i=1}^{K} \|\Theta^i\|_1 + \lambda_2 \sum_{j=1}^{p} \left\| \begin{bmatrix} V^1 \\ V^2 \\ \vdots \\ V^K \end{bmatrix}_j \right\|_q \\ \text{subject to} & \Theta^i - \text{diag}(\Theta^i) = V^i + (V^i)^T \text{ for } i = 1, \ldots, K. \end{array} \quad (13)$$

One can easily see that the problem (13) is equivalent to the problem

$$\begin{array}{ll} \underset{\Theta^i \in \mathbb{S}^p_{++}, \tilde{V}^i \in \mathbb{R}^{p\times p}, i=1\ldots K}{\text{minimize}} & -L(\Theta^1, \Theta^2, \ldots, \Theta^K) + \lambda_1 \sum_{i=1}^{K} \|\Theta^i\|_1 + \lambda_2 \sum_{j=1}^{p} \left\| \begin{bmatrix} \tilde{V}^1 - \text{diag}(\tilde{V}^1) \\ \tilde{V}^2 - \text{diag}(\tilde{V}^2) \\ \vdots \\ \tilde{V}^K - \text{diag}(\tilde{V}^K) \end{bmatrix}_j \right\|_q \\ \text{subject to} & \Theta^i = \tilde{V}^i + (\tilde{V}^i)^T \text{ for } i = 1, 2, \ldots, K, \end{array} \quad (14)$$

in the sense that the optimal solution $\{V^i\}$ to (13) and the optimal solution $\{\tilde{V}^i\}$ to (14) have the following relationship: $V^i = \tilde{V}^i - \text{diag}(\tilde{V}^i)$ for $i = 1, 2, \ldots, K$. We now present an ADMM algorithm for solving (14). We reformulate (14) by introducing additional variables in order to decouple some terms of the objective that are difficult to optimize jointly:

$$\begin{array}{ll}\underset{\Theta^i\in\mathbb{S}^p_{++},Z^i,\tilde{V}^i,W^i\in\mathbb{R}^{p\times p}}{\text{minimize}} & -L(\Theta^1,\Theta^2,\ldots,\Theta^K)+\lambda_1\sum_{i=1}^K\|Z^i\|_1+\lambda_2\sum_{j=1}^p\left\|\begin{bmatrix}\tilde{V}^1-\text{diag}(\tilde{V}^1)\\ \tilde{V}^2-\text{diag}(\tilde{V}^2)\\ \vdots\\ \tilde{V}^K-\text{diag}(\tilde{V}^K)\end{bmatrix}_j\right\|_q \\ \text{subject to} & \Theta^i=\tilde{V}^i+W^i,\ \tilde{V}^i=(W^i)^T,\ \Theta^i=Z^i\ \text{for}\ i=1,2,\ldots,K.\end{array} \tag{15}$$

The augmented Lagrangian to (15) is given by

$$\begin{aligned}&\sum_{i=1}^K n_i(-\text{logdet}(\Theta^i)+\text{trace}(S^i\Theta^i))+\lambda_1\sum_{i=1}^K\|Z^i\|_1+\lambda_2\sum_{j=1}^p\left\|\begin{bmatrix}\tilde{V}^1-\text{diag}(\tilde{V}^1)\\ \tilde{V}^2-\text{diag}(\tilde{V}^2)\\ \vdots\\ \tilde{V}^K-\text{diag}(\tilde{V}^K)\end{bmatrix}_j\right\|_q+\\ &\sum_{i=1}^K\left\{\langle F^i,\Theta^i-(\tilde{V}^i+W^i)\rangle+\langle G^i,\tilde{V}^i-(W^i)^T\rangle+\langle Q^i,\Theta^i-Z^i\rangle\right\}+\\ &\frac{\rho}{2}\sum_{i=1}^K\left\{\|\Theta^i-(\tilde{V}^i+W^i)\|_F^2+\|\tilde{V}^i-(W^i)^T\|_F^2+\|\Theta^i-Z^i\|_F^2\right\}.\end{aligned} \tag{16}$$

The corresponding ADMM algorithm is given in Algorithm 2.

Algorithm 2

ADMM algorithm for the CNJGL optimization problem (7)

```
input: $\rho > 0, \mu > 1, t_{\max} > 0$;
Initialize: Primal variables to the identity matrix and dual variables to the zero matrix;
for $t = 1{:}t_{\max}$ do
    $\rho \leftarrow \mu\rho$;
    while Not converged do
        $\Theta^i \leftarrow \text{Expand}\left(\frac{1}{2}(\tilde{V}^i + W^i + Z^i) - \frac{1}{2\rho}(Q^i + n_i S^i + F^i), \rho, n_i\right)$ for $i = 1,\ldots,K$;
        $Z^i \leftarrow \mathcal{T}_1\left(\Theta^i + \frac{Q^i}{\rho}, \frac{\lambda_1}{\rho}\right)$ for $i = 1,\ldots,K$;
        Let $C^i = \frac{1}{2}((W^i)^T - W^i + \Theta^i) + \frac{1}{2\rho}(F^i - G^i)$ for $i = 1,\ldots,K$;
        $\begin{bmatrix}\tilde{V}^1\\ \tilde{V}^2\\ \vdots\\ \tilde{V}^K\end{bmatrix} \leftarrow \mathcal{T}_q\left(\begin{bmatrix}C^1-\text{diag}(C^1)\\ C^2-\text{diag}(C^2)\\ \vdots\\ C^K-\text{diag}(C^K)\end{bmatrix}, \frac{\lambda_2}{2\rho}\right) + \begin{bmatrix}\text{diag}(C^1)\\ \text{diag}(C^2)\\ \vdots\\ \text{diag}(C^K)\end{bmatrix}$;
        $W^i \leftarrow \frac{1}{2}((\tilde{V}^i)^T - \tilde{V}^i + \Theta^i) + \frac{1}{2\rho}(F^i + (G^i)^T)$ for $i = 1,\ldots,K$;
        $F^i \leftarrow F^i + \rho(\Theta^i - (\tilde{V}^i + W^i))$ for $i = 1,\ldots,K$;
        $G^i \leftarrow G^i + \rho(\tilde{V}^i - (W^i)^T)$ for $i = 1,\ldots,K$;
        $Q^i \leftarrow Q^i + \rho(\Theta^i - Z^i)$ for $i = 1,\ldots,K$
```

4.2.3 Numerical Issues and Run-Time of the ADMM Algorithms—We set $\mu = 5$, $\rho = 0.5$ and $t_{\max} = 1000$ in the PNJGL and CNJGL algorithms. In our implementation of these algorithms, the stopping criterion for the inner loop (corresponding to a fixed ρ) is

$$\max_{i \in \{1,2,\ldots,K\}} \left\{ \frac{\|(\Theta^i)^{(k+1)} - (\Theta^i)^{(k)}\|_F}{\|(\Theta^i)^{(k)}\|_F} \right\} \leq \varepsilon,$$

where $(\Theta^i)^{(k)}$ denotes the estimate of Θ^i in the kth iteration of the ADMM algorithm, and ε is a tolerance that is chosen in our experiments to equal 10^{-4}.

The periteration complexity of the ADMM algorithms for CNJGL and PNJGL (with $K = 2$) is $O(p^3)$; this is the complexity of computing the SVD. On the other hand, the complexity of a general interior point method is $O(p^6)$. In a small example with $p = 30$, run on an Intel Xeon X3430 2.4Ghz CPU, the interior point method (using cvx, which calls Sedumi) takes 7 minutes to run, while the ADMM algorithm for PNJGL, coded in Matlab, takes only 0.58 seconds. When $p = 50$, the times are 3.5 hours and 2.1 seconds, respectively. Let $\hat{\Theta^1}$, $\hat{\Theta^2}$ and $\bar{\Theta^1}$, $\bar{\Theta^2}$ denote the solutions obtained by ADMM and cvx, respectively. We observe that on average, the error $\max_{i \in \{1,2\}} \left\{ \|\hat{\Theta}^i - \overline{\Theta}^i\|_F / \|\overline{\Theta}^i\|_F \right\}$ is on the order of 10^{-4}. Thus, the algorithm has good empirical accuracy in recovering the optimal solution.

We now present a more extensive runtime study for the ADMM algorithms for PNJGL and CNJGL. We ran experiments with $p = 100$, 200, 500 and with $n_1 = n_2 = p/2$. We generated synthetic data as described in Section 6. Results are displayed in Figures 4(a)–(d), where the panels depict the run-time and number of iterations required for the algorithm to terminate, as a function of λ_1, and with λ_2 fixed. The number of iterations required for the algorithm to terminate is computed as the total number of inner loop iterations performed in Algorithms 1 and 2. From Figures 4(b) and (d), we observe that as p increases from 100 to 500, the run-times increase substantially, but never exceed several minutes.

Figure 4(a) indicates that for CNJGL, the total number of iterations required for algorithm termination is small when λ_1 is small. In contrast, for PNJGL, Figure 4(c) indicates that the total number of iterations is large when λ_1 is small. This phenomenon results from the use of the identity matrix to initialize the network estimates in the ADMM algorithms: when λ_1 is small, the identity is a poor initialization for PNJGL, but a good initialization for CNJGL (since for CNJGL, λ_2 induces sparsity even when $\lambda_1 = 0$).

4.2.4 Convergence of the ADMM Algorithm—Problem (9) involves two (groups of) primal variables, X and Y; in this setting, convergence of ADMM has been established (see, e.g., Boyd et al., 2010; Mota et al., 2011). However, the PNJGL and CNJGL optimization problems involve more than two groups of primal variables, and convergence of ADMM in this setting is an ongoing area of research. Indeed, as mentioned in Eckstein (2012), the standard analysis for ADMM with two groups does not extend in a straightforward way to ADMM with more than two groups of variables. Han and Yuan (2012) and Hong and Luo

(2012) show convergence of ADMM with more than two groups of variables under assumptions that do not hold for CNJGL and PNJGL. Under very minimal assumptions, He et al. (2012) proved that a modified ADMM algorithm (with Gauss-Seidel updates) converges to the optimal solution for problems with any number of groups. More general conditions for convergence of the ADMM algorithm with more than two groups is left as a topic for future work. We also leave for future work a reformulation of the CNJGL and PNJGL problems as consensus problems, for which an ADMM algorithm involving two groups of primal variables can be obtained, and for which convergence would be guaranteed. Finally, note that despite the lack of convergence theory, ADMM with more than two groups has been used in practice and often observed to converge faster than other variants. As an example see Tao and Yuan (2011), where their ASALM algorithm (which is the same as ADMM with more than two groups) is reported to be significantly faster than a variant with theoretical convergence.

5. Algorithm-Independent Computational Speed-Ups

The ADMM algorithms presented in the previous section work well on problems of moderate size. In order to solve the PNJGL or CNJGL optimization problems when the number of variables is large, a faster approach is needed. We now describe conditions under which any algorithm for solving the PNJGL or CNJGL problems can be sped up substantially, for an appropriate range of tuning parameter values. Our approach mirrors previous results for the graphical lasso (Witten et al., 2011; Mazumder and Hastie, 2012), and FGL and GGL (Danaher et al., 2013). The idea is simple: if the solutions to the PNJGL or CNJGL optimization problem are block-diagonal (up to some permutation of the variables) with shared support, then we can obtain the global solution to the PNJGL or CNJGL optimization problem by solving the PNJGL or CNJGL problem separately on the variables within each block. This can lead to massive speed-ups. For instance, if the solutions are block-diagonal with L blocks of equal size, then the complexity of our ADMM algorithm reduces from $O(p^3)$ per iteration, to $O((p/L)^3)$ per iteration in each of L independent subproblems. Of course, this hinges upon knowing that the PNJGL or CNJGL solutions are block-diagonal, and knowing the partition of the variables into blocks.

In Sections 5.1–5.3 we derive necessary and sufficient conditions for the solutions to the PNJGL and CNJGL problems to be block-diagonal. Our conditions depend only on the sample covariance matrices $S^1, \ldots, S^k$ and regularization parameters λ_1, λ_2. These conditions can be applied in at most $O(p^2)$ operations. In Section 5.4, we demonstrate the speed-ups that can result from applying these sufficient conditions.

Related results for the graphical lasso (Witten et al., 2011; Mazumder and Hastie, 2012) and FGL and GGL (Danaher et al., 2013) involve a single condition that is both necessary and sufficient for the solution to be block diagonal. In contrast, in the results derived below, there is a gap between the necessary and sufficient conditions. Though only the sufficient conditions are required in order to obtain the computational speed-ups discussed in Section 5.4, knowing the necessary conditions allows us to get a handle on the tightness (and, consequently, the practical utility) of the sufficient conditions, for a particular value of the tuning parameters.

We now introduce some notation that will be used throughout this section. Let $(I_1, I_2, \ldots, I_L)$ be a partition of the index set $\{1, 2, \ldots, p\}$, and let $T = \cup_{i=1}^{L} \{I_i \times I_i\}$. Define the *support* of a matrix Θ, denoted by supp(Θ), as the set of indices of the non-zero entries in Θ. We say Θ is supported on T if $\text{supp}(\Theta) \subseteq T$. Note that any matrix supported on T is block-diagonal subject to some permutation of its rows and columns. Let $|T|$ denote the cardinality of the set T, and let T^c denote the complement of T. The scheme is displayed in Figure 5. In what follows we use an ℓ_1/ℓ_q norm in the RCON penalty, with $q \geq 1$, and let $\frac{1}{s} + \frac{1}{q} = 1$.

5.1 Conditions for PNJGL Formulation to Have Block-Diagonal Solutions

In this section, we give necessary conditions and sufficient conditions on the regularization parameters λ_1, λ_2 in the PNJGL problem (6) so that the resulting precision matrix estimates $\hat{\Theta^1}, \ldots, \hat{\Theta^K}$ have a shared block-diagonal structure (up to a permutation of the variables).

We first present a necessary condition for $\hat{\Theta^1}$ and $\hat{\Theta^2}$ that minimize (6) with $K = 2$ to be block-diagonal.

Theorem 4—Suppose that the matrices $\hat{\Theta^1}$ and $\hat{\Theta^2}$ that minimize (6) with $K = 2$ have support T. Then, if $q \geq 1$, it must hold that

$$n_k |S_{ij}^k| \leq \lambda_1 + \lambda_2/2 \quad \forall (i,j) \in T^c, \quad \text{for } k = 1, 2, \text{ and} \quad (17)$$

$$|n_1 S_{ij}^1 + n_2 S_{ij}^2| \leq 2\lambda_1 \quad \forall (i,j) \in T^c. \quad (18)$$

Furthermore, if $q > 1$, then it must additionally hold that

$$\frac{n_k}{|T^c|} \sum_{(i,j) \in T^c} |S_{ij}^k| \leq \lambda_1 + \frac{\lambda_2}{2} \left(\frac{p}{|T^c|} \right)^{1/s}, \quad \text{for } k = 1, 2. \quad (19)$$

Remark 5—If $|T^c| = O(p^r)$ with $r > 1$, then as $p \to \infty$, (19) simplifies to $\frac{n_k}{|T^c|} \sum_{(i,j) \in T^c} |S_{ij}^k| \leq \lambda_1$.

We now present a sufficient condition for $\hat{\Theta^1}, \ldots, \hat{\Theta^K}$ that minimize (6) to be block-diagonal.

Theorem 6—For $q \geq 1$, a sufficient condition for the matrices $\hat{\Theta^1}, \ldots, \hat{\Theta^K}$ that minimize (6) to each have support T is that

$$n_k |S_{ij}^k| \leq \lambda_1 \quad \forall (i,j) \in T^c, \quad \text{for } k = 1, \ldots, K.$$

Furthermore, if $q = 1$ and $K = 2$, then the necessary conditions (17) and (18) are also sufficient.

When $q = 1$ and $K = 2$, then the necessary and sufficient conditions in Theorems 4 and 6 are identical, as was previously reported in Danaher et al. (2013). In contrast, there is a gap between the necessary and sufficient conditions in Theorems 4 and 6 when $q > 1$ and $\lambda_2 > 0$. When $\lambda_2 = 0$, the necessary and sufficient conditions in Theorems 4 and 6 reduce to the results laid out in Witten et al. (2011) for the graphical lasso.

5.2 Conditions for CNJGL Formulation to Have Block-Diagonal Solutions

In this section, we give necessary and sufficient conditions on the regularization parameters λ_1, λ_2 in the CNJGL optimization problem (7) so that the resulting precision matrix estimates $\hat{\Theta^1}, \dots, \hat{\Theta^K}$ have a shared block-diagonal structure (up to a permutation of the variables).

Theorem 7—Suppose that the matrices $\hat{\Theta^1}, \hat{\Theta^2}, \dots, \hat{\Theta^K}$ that minimize (7) have support T. Then, if $q \geq 1$, it must hold that

$$n_k|S_{ij}^k| \leq \lambda_1 + \lambda_2/2 \quad \forall (i,j) \in T^c, \quad \text{for } k=1,\dots,K.$$

Furthermore, if $q > 1$, then it must additionally hold that

$$\frac{n_k}{|T^c|}\sum_{(i,j)\in T^c}|S_{ij}^k| \leq \lambda_1 + \frac{\lambda_2}{2}\left(\frac{p}{|T^c|}\right)^{1/s}, \quad \text{for } k=1,\dots,K. \tag{20}$$

Remark 8—If $|T^c| = O(p^r)$ with $r > 1$, then as $p \to \infty$, (20) simplifies to $\frac{n_k}{|T^c|}\sum_{(i,j)\in T^c}|S_{ij}^k| \leq \lambda_1$.

We now present a sufficient condition for $\hat{\Theta^1}, \hat{\Theta^2}, \dots, \hat{\Theta^K}$ that minimize (7) to be block-diagonal.

Theorem 9—A sufficient condition for $\hat{\Theta^1}, \hat{\Theta^2}, \dots, \hat{\Theta^K}$ that minimize (7) to have support T is that

$$n_k|S_{ij}^k| \leq \lambda_1 \quad \forall (i,j) \in T^c, \quad \text{for } k=1,\dots,K.$$

As was the case for the PNJGL formulation, there is a gap between the necessary and sufficient conditions for the estimated precision matrices from the CNJGL formulation to have a common block-diagonal support.

5.3 General Sufficient Conditions

In this section, we give sufficient conditions for the solution to a general class of optimization problems that include FGL, PNJGL, and CNJGL as special cases to be block-diagonal. Consider the optimization problem

$$\underset{\Theta^1,\ldots,\Theta^K \in \mathbb{S}^p_{++}}{\text{minimize}} \left\{ \sum_{k=1}^{K} n_k(-\text{logdet}(\Theta^k)+\langle\Theta^k, S^k\rangle)+\sum_{k=1}^{K}\lambda_1\|\Theta^k\|_1+\lambda_2 h(\Theta^1,\ldots,\Theta^K)\right\}. \quad (21)$$

Once again, let T be the support of a $p \times p$ block-diagonal matrix. Let Θ_T denote the restriction of any $p \times p$ matrix Θ to T; that is, $(\Theta_T)_{ij}=\begin{cases}\Theta_{ij} & \text{if } (i,j)\in T\\ 0 & \text{else}\end{cases}$. Assume that the function h satisfies

$$h(\Theta^1,\ldots,\Theta^K)>h(\Theta^1_U,\ldots,\Theta^K_U)$$

for any matrices $\Theta^1, \ldots, \Theta^K$ whose support strictly contains U.

Theorem 10—A sufficient condition for the matrices $\Theta^{\hat{1}}, \ldots, \Theta^{\hat{K}}$ that solve (21) to have support T is that

$$n_k|S^k_{ij}| \le \lambda_1 \quad \forall (i,j)\in T^c, \quad \text{for } k=1,\ldots,K.$$

Note that this sufficient condition applies to a broad class of regularizers h; indeed, the sufficient conditions for PNJGL and CNJGL given in Theorems 6 and 9 are special cases of Theorem 10. In contrast, the necessary conditions for PNJGL and CNJGL in Theorems 4 and 7 exploit the specific structure of the RCON penalty.

5.4 Evaluation of Speed-Ups on Synthetic Data

Theorems 6 and 9 provide sufficient conditions for the precision matrix estimates from PNJGL or CNJGL to be block-diagonal with a given support. How can these be used in order to obtain computational speed-ups? We construct a $p \times p$ matrix A with elements

$$A_{ij}=\begin{cases}1 & \text{if } i=j\\ 1 & \text{if } n_k|S^k_{ij}|>\lambda_1 \text{ for any } k=1,\ldots,K\\ 0 & \text{else}\end{cases}.$$

We can then check, in $O(p^2)$ operations, whether A is (subject to some permutation of the rows and columns) block-diagonal, and can also determine the partition of the rows and columns corresponding to the blocks (see, e.g., Tarjan, 1972). Then, by Theorems 6 and 9, we can conclude that the PNJGL or CNJGL estimates are block-diagonal, with the same partition of the variables into blocks. Inspection of the PNJGL and CNJGL optimization problems reveals that we can then solve the problems on the variables within each block separately, in order to obtain the global solution to the original PNJGL or CNJGL optimization problems.

We now investigate the speed-ups that result from applying this approach. We consider the problem of estimating two networks of size $p = 500$. We create two inverse covariance matrices that are block diagonal with two equally-sized blocks, and sparse within each block. We then generate $n_1 = 250$ observations from a multivariate normal distribution with the first covariance matrix, and $n_2 = 250$ observations from a multivariate normal distribution with the second covariance matrix. These observations are used to generate sample covariance matrices S^1 and S^2. We then performed CNJGL and PNJGL with $\lambda_2 = 1$ and a range of λ_1 values, with and without the computational speed-ups just described.

Figure 6 displays the performance of the CNJGL and PNJGL formulations, averaged over 20 data sets generated in this way. In each panel, the *x*-axis shows the number of blocks into which the optimization problems were decomposed using the sufficient conditions; note that this is a surrogate for the value of λ_1 in the CNJGL or PNJGL optimization problems. Figure 6(a) displays the ratio of the run-time taken by the ADMM algorithm when exploiting the sufficient conditions to the run-time when not using the sufficient conditions. Figure 6(b) displays the true-positive ratio|that is, the ratio of the number of true positive edges in the precision matrix estimates to the total number of edges in the precision matrix estimates. Figure 6(c) displays the total number of true positives for the CNJGL and PNJGL estimates. Figure 6 indicates that the sufficient conditions detailed in this section lead to substantial computational improvements.

6. Simulation Study

In this section, we present the results of a simulation study demonstrating the empirical performance of PNJGL and CNJGL.

6.1 Data Generation

In the simulation study, we generated two synthetic networks (either Erdos-Renyi, scale-free, or community), each of which contains a common set of *p* nodes. Four of the *p* nodes were then modified in order to create two perturbed nodes and two co-hub nodes. Details are provided in Sections 6.1.1–6.1.3.

6.1.1 Data Generation for Erdos-Renyi Network—We generated the data as follows, for $p = 100$, and $n \in \{25, 50, 100, 200\}$:

Step 1: To generate an Erdos-Renyi network, we created a $p \times p$ symmetric matrix A with elements

$$A_{ij} \sim_{\text{i.i.d.}} \begin{cases} 0 & \text{with probability } 0.98, \\ \text{Unif}([-0.6, -0.3] \cup [0.3, 0.6]) & \text{otherwise.} \end{cases}$$

Step 2: We duplicated A into two matrices, A^1 and A^2. We selected two nodes at random, and for each node, we set the elements of the corresponding row and column of either A^1 or A^2 (chosen at random) to be i.i.d. draws from a Unif($[-0.6, -0.3] \cup [0.3, 0.6]$) distribution. This results in two perturbed nodes.

Step 3: We randomly selected two nodes to serve as co-hub nodes, and set each element of the corresponding rows and columns in each network to be i.i.d. draws from a Unif($[-0.6, -0.3] \cup [0.3, 0.6]$) distribution. In other words, these co-hub nodes are *identical* across the two networks.

Step 4: In order to make the matrices positive definite, we let $c = \min(\lambda_{\min}(A^1), \lambda_{\min}(A^2))$, where $\lambda_{\min}(\cdot)$ indicates the smallest eigenvalue of the matrix. We then set $(\Sigma^1)^{-1}$ equal to $A^1 + (0.1+|c|)I$ and set $(\Sigma^2)^{-1}$ equal to $A^2 + (0.1+|c|)I$, where I is the $p \times p$ identity matrix.

Step 5: We generated n independent observations each from a $N(0, \Sigma^1)$ and a $N(0, \Sigma^2)$ distribution, and used them to compute the sample covariance matrices S^1 and S^2.

6.1.2 Data Generation for Scale-free Network—The data generation proceeded as in Section 6.1.1, except that Step 1 was modified:

Step 1: We used the SFNG functions in Matlab (George, 2007) with parameters mlinks=2 and seed=1 to generate a scale-free network with p nodes. We then created a $p \times p$ symmetric matrix A that has non-zero elements only for the edges in the scale-free network. These non-zero elements were generated i.i.d. from a Unif($[-0.6, -0.3] \cup [0.3, 0.6]$) distribution.

Steps 2–5 proceeded as in Section 6.1.1.

6.1.3 Data Generation for Community Network—We generated data as in Section 6.1.1, except for one modification: at the end of Step 3, we set the [1:40, 61:100] and [61:100, 1:40] submatrices of A^1 and A^2 equal to zero.

Then A^1 and A^2 have non-zero entries concentrated in the top and bottom 60×60 principal submatrices. These two submatrices correspond to two communities. Twenty nodes overlap between the two communities.

6.2 Results

We now define several metrics used to measure algorithm performance. We wish to quantify each algorithm's (1) recovery of the support of the true inverse covariance matrices, (2) successful detection of co-hub and perturbed nodes, and (3) error in estimation of $\Theta^1 = (\Sigma^1)^{-1}$ and $\Theta^2 = (\Sigma_2)^{-1}$. Details are given in Table 1. These metrics are discussed further in Appendix G.

We compared the performance of PNJGL to its edge-based counterpart FGL, as well as to graphical lasso (GL). We compared the performance of CNJGL to GGL and GL. We expect CNJGL to be able to detect co-hub nodes (and, to a lesser extent, perturbed nodes), and we expect PNJGL to be able to detect perturbed nodes. (The co-hub nodes will not be detected by PNJGL, since they are identical across the networks.)

The simulation results for the set-up of Section 6.1.1 are displayed in Figures 7 and 8. Each row corresponds to a sample size while each column corresponds to a performance metric. In Figure 7, PNJGL, FGL, and GL are compared, and in Figure 8, CNJGL, GGL, and GL are compared. Within each plot, each colored line corresponds to the results obtained using a fixed value of λ_2 (for either PNJGL, FGL, CNJGL, or GGL), as λ_1 is varied. Recall that GL corresponds to any of these four approaches with $\lambda_2 = 0$. Note that the number of positive edges (defined in Table 1) decreases approximately monotonically with the regularization parameter λ_1, and so on the *x*-axis we plot the number of positive edges, rather than λ_1, for ease of interpretation.

In Figure 7, we observe that PNJGL outperforms FGL and GL for a suitable range of the regularization parameter λ_2, in the sense that for a fixed number of edges estimated, PNJGL identifies more true positives, correctly identifies a greater ratio of perturbed nodes, and yields a lower Frobenius error in the estimates of Θ^1 and Θ^2. In particular, PNJGL performs best relative to FGL and GL when the number of samples is the smallest, that is, in the high-dimensional data setting. Unlike FGL, PNJGL fully exploits the fact that differences between Θ^1 and Θ^2 are due to node perturbation. Not surprisingly, GL performs worst among the three algorithms, since it does not borrow strength across the conditions in estimating Θ^1 and Θ^2.

In Figure 8, we note that CNJGL outperforms GGL and GL for a suitable range of the regularization parameter λ_2. In particular, CNJGL outperforms GGL and GL by a larger margin when the number of samples is the smallest. Once again, GL performs the worst since it does not borrow strength across the two networks; CNJGL performs the best since it fully exploits the presence of hub nodes in the data.

We note one interesting feature of Figure 8: the colored lines corresponding to CNJGL with very large values of λ_2 do not extend beyond around 400 positive edges. This is because for CNJGL, a large value of λ_2 induces sparsity in the network estimates, even if λ_1 is small or zero. Consequently, it is not possible to obtain a dense estimate of Θ^1 and Θ^2 if CNJGL is performed with a large value of λ_2. In contrast, in the case of PNJGL, sparsity is induced only by λ_1, and not at all by λ_2. We note that a similar situation occurs for the edge-based counterparts of CNJGL and PNJGL: when GGL is performed with a large value of λ_2 then the network estimates are necessarily sparse, regardless of the value of λ_1. But the same is not true for FGL.

The simulation results for the set-ups of Sections 6.1.2 and 6.1.3 are displayed in Figures 9 and 10, respectively, for the case $n = 50$. The results show that once again, PNJGL and CNJGL substantially outperform the edge-based approaches on the three metrics defined earlier.

7. Real Data Analysis

In this section, we present the results of PNJGL and CNJGL applied to two real data sets: gene expression data set and university webpage data set.

7.1 Gene Expression Data

In this experiment, we aim to reconstruct the gene regulatory networks of two subtypes of glioblastoma multiforme (GBM), as well as to identify genes that can improve our understanding of the disease. Cancer is caused by somatic (cancer-specific) mutations in the genes involved in various cellular processes including cell cycle, cell growth, and DNA repair; such mutations can lead to uncontrolled cell growth. We will show that PNJGL and CNJGL can be used to identify genes that play central roles in the development and progression of cancer. PNJGL tries to identify genes whose interactions with other genes vary significantly between the subtypes. Such genes are likely to have deleterious somatic mutations. CNJGL tries to identify genes that have interactions with many other genes in all subtypes. Such genes are likely to play an important role in controlling other genes' expression, and are typically called *regulators*.

We applied the proposed methods to a publicly available gene expression data set that measures mRNA expression levels of 11,861 genes in 220 tissue samples from patients with GBM (Verhaak et al., 2010). The raw gene expression data were generated using the Affymetrix GeneChips technology. We downloaded the raw data in .CEL format from the The Caner Genome Atlas (TCGA) website. The raw data were normalized by using the Affymetrix MAS5 algorithm, which has been shown to perform well in many studies (Lim et al., 2007). The data were then log2 transformed and batch-effected corrected using the software ComBat (Johnson and Li, 2006). Each patient has one of four subtypes of GBM| Proneural, Neural, Classical, or Mesenchymal. We selected two subtypes, Proneural (53 tissue samples) and Mesenchymal (56 tissue samples), that have the largest sample sizes. All analyses were restricted to the corresponding set of 109 tissue samples.

To evaluate PNJGL's ability to identify genes with somatic mutations, we focused on the following 10 genes that have been suggested to be frequently mutated across the four GBM subtypes (Verhaak et al., 2010): TP53, PTEN, NF1, EGFR, IDH1, PIK3R1, RB1, ERBB2, PIK3CA, PDGFRA. We then considered inferring the regulatory network of a set of genes that is known to be involved in a single biological process, based on the Reactome database (Matthews et al., 2008). In particular, we focused our analysis on the "TCR signaling" gene set, which contains the largest number of mutated genes. This gene set contains 34 genes, of which three (PTEN, PIK3R1, and PIK3CA) are in the list of 10 genes suggested to be mutated in GBM. We applied PNJGL with $q = 2$ to the resulting 53×34 and 56×34 gene expression data sets, after standardizing each gene to have variance one. As can be seen in Figure 11, the pattern of network differences indicates that one of the three highly-mutated genes is in fact perturbed across the two GBM subtypes. The perturbed gene is PTEN, a tumor suppressor gene, and it is known that mutations in this gene are associated with the development and progression of many cancers (see, e.g., Chalhoub and Baker, 2009).

To evaluate the performance of CNJGL in identifying genes known to be regulators, we used a manually curated list of genes that have been identified as regulators in a previous study (Gentles et al., 2009); this list includes genes annotated as transcription factors, chromatin modifiers, or translation initiation genes. We then selected a gene set from Reactome, called "G2/M checkpoints," which is relevant to cancer and contains a large number of regulators. This gene set contains 38 genes of which 15 are regulators. We applied CNJGL to the resulting 53×38 and 56×38 gene expression data sets, to see if the 15 regulators tend to be identified as co-hub genes. Figure 12 indicates that all four co-hub genes (CDC6, MCM6, CCNB1 and CCNB2) detected by CNJGL are known to be regulators.

7.2 University Webpage Data

We applied PNJGL and CNJGL to the university webpages data set from the “World Wide Knowledge Base” project at Carnegie Mellon University. This data set was pre-processed by Cardoso-Cachopo (2009). The data set describes the number of appearances of various terms, or words, on webpages from the computer science departments of Cornell, Texas, Washington and Wisconsin. We consider the 544 student webpages, and the 374 faculty webpages. We standardize the student webpage data so that each term has mean zero and standard deviation one, and we also standardize the faculty webpage data so that each term has mean zero and standard deviation one. Our goal is to identify terms that are perturbed or co-hub between the student and faculty webpage networks. We restrict our analysis to the 100 terms with the largest entropy.

We performed 5-fold cross-validation of the log-likelihood, computed as

$$\log\det\Theta^1 - \operatorname{trace}(S^1\Theta^1) + \log\det\Theta^2 - \operatorname{trace}(S^2\Theta^2),$$

for PNJGL, FGL, CNJGL, GGL, and GL, using a range of tuning parameters. The results for PNJGL, FGL and GL are found in Figure 13(a). PNJGL and FGL achieve comparable log-likelihood values. However, for a fixed number of non-zero edges, PNJGL outperforms FGL, suggesting that PNJGL can achieve a comparable model fit for a more interpretable model. Figure 13(b) displays the results for CNJGL, GGL and GL. It appears that PNJGL and FGL provide the best fit to the data.

Given that PNJGL fits the data well, we highlight a particular solution, found in Figure 14. PNJGL is performed with $\lambda_1 = 27$, $\lambda_2 = 381$; these values were chosen because they result in a high log-likelihood in Figure 13(a), and yield an interpretable pair of network estimates. Several perturbed nodes are identified: *advisor*, *high*, *construct*, *email*, *applic*, *fax*, *and receiv*. The student and faculty webpage precision matrices, $\Theta^{\hat{S}}$ and $\Theta^{\hat{F}}$, are overlaid in Figure 14.

For example, the perturbed node *receiv* is connected to the terms *advis*, *inform*, and *student* among the student webpages. In contrast, among faculty webpages, the phrase *receiv* is connected to *associate* and *faculty*.

8. Discussion

We have proposed node-based learning of multiple Gaussian graphical models through the use of two convex formulations, perturbed-node joint graphical lasso and cohub node joint graphical lasso. These techniques are well-motivated by many real-world applications, such as learning transcriptional regulatory networks in multiple contexts from gene expression data. Both of these formulations rely on the use of the row-column overlap norm penalty, which when applied to a matrix encourages a support that can be expressed as the union of a few rows and columns. We solve the convex optimization problems that correspond to PNJGL and CNJGL using the ADMM algorithm, which is more efficient and scalable than standard interior point methods and also first-order methods such as projected subgradient. We also provide necessary and sufficient conditions on the regularization parameters in CNJGL and PNJGL so that the optimal solutions to these formulations are block diagonal, up to a permutation of the rows and columns. When the sufficient conditions are met, any algorithm that is applicable to these two formulations can be sped up by breaking down the optimization problem into smaller subproblems. Our proposed approaches lead to better performance than two alternative approaches: learning Gaussian graphical models under the assumption of edge perturbation or shared edges, or simply learning each model separately.

We next discuss possible directions for future work.

- We have focused on promoting a row-column structure in either the difference of the networks or in the networks themselves. However, the RCON penalty can be generalized to other forms of structured sparsity. For instance, we might believe that particular sets of genes in the same pathway tend to be simultaneously activated or perturbed across multiple distinct conditions; a modification of the RCON penalty can be used in this setting.
- Convergence of the ADMM algorithm in the presence of more than two sets of variable updates has only been addressed partially in the literature. However, the PNJGL and CNJGL formulations can be rewritten along the lines of an approach given in Ma et al. (2013), so that only two sets of primal variables are involved, so that convergence is guaranteed. We leave for future study an investigation of whether this alternative approach leads to better performance in practice.
- Transcriptional regulatory networks involve tens of thousands of genes. Hence it is imperative that our algorithms scale up to large problem sizes. In future work, speedups of our ADMM algorithm as well as adaptations of other fast algorithms such as the accelerated proximal gradient method or second-order methods can be considered.
- In Section 5, we presented a set of conditions that allow us to break up the CNJGL and PNJGL optimization problems into many independent subproblems. However, there is a gap between the necessary and sufficient conditions that we presented. Making this gap tighter could potentially lead to greater computational improvements.
- Tuning parameter selection in high-dimensional unsupervised settings remains an open problem. An existing approach such as stability selection (Meinshausen and Buhlmann, 2010) could be applied in order to select the tuning parameters λ_1 and λ_2 for CNJGL and PNJGL.
- The CNJGL and PNJGL formulations are aimed at jointly learning several high-dimensional Gaussian graphical models. These approaches could be modified in order to learn other types of probabilistic graphical models (see, e.g., Ravikumar et al., 2010; Yang et al., 2012).

- It is well-known that adaptive weights can improve the performance of penalized estimation approaches in other contexts (e.g., the adaptive lasso of Zou, 2006 improves over the lasso of Tibshirani, 1996). In a similar manner, the use of adaptive weights may provide improvement over the PNJGL and CNJGL proposals in this paper. Other options include *reweighted* ℓ_1 *norm* approaches that adjust the weights iteratively: one example is the algorithm proposed in Lobo et al. (2007) and further studied in Candes et al. (2007). This algorithm uses a weight for each variable that is proportional to the inverse of its value in the previous iteration, yielding improvements over the use of an ℓ_1 norm. This method can be seen as locally minimizing the sum of the logarithms of the entries, solved by iterative linearization. In general, any of these approaches can be explored for the problems in this paper.

Matlab code implementing CNJGL and PNJGL is available at http://faculty.washington.edu/mfazel/, http://www.biostat.washington.edu/~dwitten/software.html, and http://suinlee.cs.washington.edu/software.

Acknowledgments

The authors acknowledge funding from the following sources: NIH DP5OD009145 and NSF CAREER DMS-1252624 to DW, NSF CAREER ECCS-0847077 to MF, and Univ. Washington Royalty Research Fund to DW, MF, and SL.

References

Argyriou A, Micchelli CA, Pontil M. Efficient first order methods for linear composite regularizers. 2011 arXiv:1104.1436 [cs.LG].

Banerjee O, El Ghaoui LE, d'Aspremont A. Model selection through sparse maximum likelihood estimation for multivariate Gaussian or binary data. JMLR. 2008; 9:485–516.

Boyd SP, Parikh N, Chu E, Peleato B, Eckstein J. Distributed optimization and statistical learning via the alternating direction method of multipliers. Foundations and Trends in ML. 2010; 3(1):1–122.

Candes EJ, Wakin M, Boyd S. Enhancing sparsity by reweighted l1 minimization. Journal of Fourier Analysis and Applications. 2007; 14:877–905.

Cardoso-Cachopo, A. 2009. URL http://web.ist.utl.pt/acardoso/datasets/

Chalhoub N, Baker SJ. PTEN and the PI3-kinase pathway in cancer. Annual Review of Pathology. 2009; 4:127–150.

Chen, X.; Lin, Q.; Kim, S.; Carbonell, JG.; Xing, EP. Smoothing proximal gradient method for general structured sparse learning. Proceedings of the Conference on Uncertainty in Artificial Intelligence; 2011.

Danaher P, Wang P, Witten D. The joint graphical lasso for inverse covariance estimation across multiple classes. Journal of the Royal Statistical Society, Series B. 2013

D'Aspremont A, Banerjee O, El Ghaoui L. First-order methods for sparse covariance selection. SIAM Journal on Matrix Analysis and Applications. 2008; 30(1):56–66.

Duchi J, Singer Y. Efficient online and batch learning using forward backward splitting. Journal of Machine Learning Research. 2009:2899–2934.

Eckstein, J. Technical Report RUTCOR Research Report RRR 32-2012. Rutgers University; 2012. Augmented lagrangian and alternating direction methods for convex optimization: A tutorial and some illustrative computational results.

Eckstein J, Bertsekas DP. On the douglas-rachford splitting method and the proximal point algorithm for maximal monotone operators. Mathematical Programing: Series A and B. 1992; 55(3):293–318.

Friedman J, Hastie T, Tibshirani R. Sparse inverse covariance estimation with the graphical lasso. Biostatistics. 2007; 9:432–441. [PubMed: 18079126]

Gabay D, Mercier B. A dual algorithm for the solution of nonlinear variational problems via finite element approximation. Computers and Mathematics with Applications. 1976; 2(1):17–40.

Gentles AJ, Alizadeh AA, Lee SI, Myklebust JH, Shachaf CM, Levy R, Koller D, Plevritis SK. A pluripotency signature predicts histologic transformation and influences survival in follicular lymphoma patients. Blood. 2009; 114(15):3158–66. [PubMed: 19636063]

George, M. Matlab code. 2007. B-a scale-free network generation and visualization.

Grant, M.; Boyd, S. [October 2010] cvx version 1.21. “http://cvxr.com/cvx”

Guo J, Levina E, Michailidis G, Zhu J. Joint estimation of multiple graphical models. Biometrika. 2011; 98(1):1–15. [PubMed: 23049124]

Han D, Yuan Z. A note on the alternating direction method of multipliers. Journal of Optimization Theory and Applications. 2012; 155(1):227–238.

Hara S, Washio T. Learning a common substructure of multiple graphical gaussian models. Neural Networks. 2013; 38:23–38. [PubMed: 23220164]

He B, Tao M, Yuan X. Alternating direction method with gaussian back substitution for separable convex programming. SIAM Journal of Optimization. 2012:313–340.

Hong, M.; Luo, Z. On the linear convergence of the alternating direction method of multipliers. 2012. arXiv:1208:3922 [math.OC]

Hsieh CJ, Sustik M, Dhillon I, Ravikumar P. Sparse inverse covariance estimation using quadratic approximation. Advances in Neural Information Processing Systems. 2011

Jacob, L.; Obozinski, G.; Vert, JP. Group lasso with overlap and graph lasso. Proceedings of the 26th International Conference on Machine Learning; 2009.

van Johnson WE, Li C. Adjusting batch effects in microarray expression data using empirical bayes methods. Biostatistics. 2006; 8(1):118–27. [PubMed: 16632515]

Kolar M, Song L, Ahmed A, Xing EP. Estimating time-varying networks. Annals of Applied Statistics. 2010; 4 (1):94–123.

Lauritzen, SL. Graphical Models. Oxford Science Publications; 1996.

Lim WK, Wang J, Lefebvre C, Clifano A. Comparative analysis of microarray normalization procedures: effects on reverse engineering gene networks. Bioinformatics. 2007; 23(13):282–288.

Lobo M, Fazel M, Boyd S. Portfolio optimization with linear and fixed transaction costs. Annals of Operations Research. 2007; 152(1):376–394.

Ma S, Xue L, Zou H. Alternating direction methods for latent variable Gaussian graphical model selection. Neural Computation. 2013

Mardia, KV.; Kent, J.; Bibby, JM. Multivariate Analysis. Academic Press; 1979.

Matthews L, Gopinath G, Gillespie M, Caudy M, Croft D, de Bono B, Garapati P, Hemish J, Hermjakob H, Jassal B, Kanapin A, Lewis S, Mahajan S, May B, Schmidt E, Vastrik I, Wu G, Birney E, Stein L, D’Eustachio P. Reactome knowledgebase of biological pathways and processes. Nucleic Acids Research. 2008; 37:D619–22. [PubMed: 18981052]

Mazumder R, Hastie T. Exact covariance-thresholding into connected components for large-scale graphical lasso. Journal of Machine learning Research. 2012; 13:723–736.

Meinshausen M, Buhlmann P. Stability selection (with discussion). Journal of the Royal Statistical Society, Series B. 2010; 72:417–473.

Mohan K, Chung M, Han S, Witten D, Lee S, Fazel M. Structured learning of gaussian graphical models. Advances in Neural Information Processing Systems. 2012

Mosci S, Villa S, Verri A, Rosasco L. A primal-dual algorithm for group sparse regularization with overlapping groups. Advances in Neural Information Processing Systems. 2010:2604–2612.

Mota, JFC.; Xavier, JMF.; Aguiar, PMQ.; Puschel, M. A proof of convergence for the alternating direction method of multipliers applied to polyhedral-constrained functions. 2011. arXiv: 1112.2295 [math.OC]

Obozinski, G.; Jacob, L.; Vert, JP. Group lasso with overlaps: the latent group lasso approach. 2011. arXiv preprint arXiv:1110.0413

Ravikumar P, Wainwright MJ, Raskutti G, Yu B. Model selection in gaussian graphical models: high-dimensional consistency of l1-regularized MLE. Advances in Neural Information Processing Systems. 2008

Ravikumar P, Wainwright MJ, Lafferty JD. High-dimensional Ising model selection using l1-regularized logistic regression. Annals of Statisitcs. 2010; 38(3):1287–1319.

Rothman A, Levina E, Zhu J. Sparse permutation invariant covariance estimation. Electronic Journal of Statistics. 2008; 2:494–515.

Scheinberg K, Ma S, Goldfarb D. Sparse inverse covariance selection via alternating linearization methods. Advances in Neural Information Processing Systems. 2010

Tao M, Yuan X. Recovering low-rank and sparse components of matrices from incomplete and noisy observations. SIAM J Optimization. 2011; 21(1):57–81.

Tarjan RE. Depth-first search and linear graph algorithms. SIAM Journal on Computing. 1972; 1(2): 146–160.

Tibshirani R. Regression shrinkage and selection via the lasso. Journal of the Royal Statistical Society, Series B. 1996; 58:267–288.

Tibshirani R, Saunders M, Rosset S, Zhu J, Knight K. Sparsity and smoothness via the fused lasso. Journal of the Royal Statistical Society, Series B. 2005; 67:91–108.

Verhaak RGW, et al. Integrated genomic analysis identifies clinically relevant subtypes of glioblastoma characterized by abnormalities in PDGFRA, IDH1, EGFR, and NF1. Cancer Cell. 2010; 17(1):98–110. [PubMed: 20129251]

Witten DM, Friedman JH, Simon N. New insights and faster computations for the graphical lasso. Journal of Computational and Graphical Statistics. 2011; 20(4):892–900.

Yang E, Ravikumar P, Allen GI, Liu Z. Graphical models via generalized linear models. Advances in Neural Information Processing Systems. 2012

Yuan M, Lin Y. Model selection and estimation in the Gaussian graphical model. Biometrika. 2007a; 94(10):19–35.

Yuan M, Lin Y. Model selection and estimation in regression with grouped variables. Journal of the Royal Statistical Society, Series B. 2007b; 68:49–67.

Zhang, B.; Wang, Y. Learning structural changes of Gaussian graphical models in controlled experiments. Proc. 26th Conference on Uncertainty in Artifical Intelligence; 2010.

Zou H. The adaptive lasso and its oracle properties. Journal of the American Statistical Association. 2006; 101:1418–1429.

Appendix A. Dual Characterization of RCON

Lemma 11

The dual representation of Ω is given by

$$\begin{aligned}\Omega(\Theta^1,\ldots,\Theta^K)= & \max_{\Lambda^1,\ldots,\Lambda^K\in\mathbb{R}^{p\times p}} && \sum_{i=1}^{K}\langle\Lambda^i,\Theta^i\rangle \\ & \text{subject to} && \left\|\begin{bmatrix}\Lambda^1+(\Lambda^1)^T\\ \vdots \\ \Lambda^K+(\Lambda^K)^T\end{bmatrix}_j\right\|_* \leq 1\, for\, j=1,2,\ldots,p,\end{aligned} \tag{22}$$

where ||·|| denotes any norm, and $||\cdot||_*$ its corresponding dual norm.

Proof

Recall that Ω is given by

$$\Omega(\Theta^1,\ldots,\Theta^K)=\min_{V^1,\ldots,V^K\in\mathbb{R}^{p\times p}} \left\| \begin{bmatrix} V^1 \\ \vdots \\ V^K \end{bmatrix} \right\| \quad \text{subject to} \quad \Theta^i=V^i+(V^i)^T,\ i=1,2,\ldots,K. \tag{23}$$

Let $Z=\begin{bmatrix} Z^1 \\ \vdots \\ Z^K \end{bmatrix}$ where $Z^k \in \mathbb{R}^{p\times p}$. Then (23) is equivalent to

$$\Omega(\Theta^1,\ldots,\Theta^K)=\min_{V^i:\Theta^i=V^i+(V^i)^T,i=1,\ldots,K}\ \max_{Z:\|Z\|_*\le 1}\ \sum_{i=1}^{K}\langle Z^i,V^i\rangle, \tag{24}$$

where $\|.\|_*$ is the dual norm to $\|.\|$. Since in (24) the cost function is bilinear in the two sets of variables and the constraints are compact convex sets, by the minimax theorem, we can swap max and min to get

$$\Omega(\Theta^1,\ldots,\Theta^K)=\max_{Z:\|Z\|_*\le 1}\ \min_{V^i:\Theta^i=V^i+(V^i)^T,i=1,\ldots,K}\sum_{i=1}^{K}\langle Z^i,V^i\rangle. \tag{25}$$

Now, note that the dual to the inner minimization problem with respect to $V^1, \ldots, V^K$ in (25) is given by

$$\underset{\Lambda^1,\ldots,\Lambda^K}{\text{maximize}}\ \sum_{i=1}^{K}\langle\Lambda^i,\Theta^i\rangle \quad \text{subject to} \quad Z^i=\Lambda^i+(\Lambda^i)^T,\ i=1,2,\ldots,K. \tag{26}$$

Plugging (26) into (25), the lemma follows.

By definition, the subdifferential of Ω is given by the set of all K-tuples $(\Lambda^1, \ldots, \Lambda^K)$ that are optimal solutions to problem (22). Note that if $(\Lambda^1, \ldots, \Lambda^K)$ is an optimal solution to (22), then any $(\Lambda^1 + Y^1, \ldots, \Lambda^K + Y^K)$ with skew-symmetric matrices $Y^1, \ldots, Y^K$ is also an optimal solution.

Appendix B. Proof of Theorem 4

The optimality conditions for the PNJGL optimization problem (6) with $K = 2$ are given by

$$-n_1(\Theta^1)^{-1}+n_1S^1+\lambda_1\Gamma^1+\lambda_2\Lambda=0, \tag{27}$$

$$-n_2(\Theta^2)^{-1}+n_2S^2+\lambda_1\Gamma^2-\lambda_2\Lambda=0, \tag{28}$$

where Γ^1 and Γ^2 are subgradients of $\|\Theta^1\|_1$ and $\|\Theta^2\|_1$, and $(\Lambda, -\Lambda)$ is a subgradient of $\Omega_q(\Theta^1 - \Theta^2)$. (Note that $\Omega_q(\Theta^1 - \Theta^2)$ is a composition of Ω_q with the linear function $\Theta^1 - \Theta^2$, and apply the chain rule.) Also note that the right-hand side of the above equations is a zero matrix of size $p \times p$.

Now suppose that Θ^1 and Θ^2 that solve (6) are supported on T. Then since $(\Theta^1)^{-1}$, $(\Theta^2)^{-1}$ are supported on T, we have that

$$\begin{array}{l} n_1 S^1_{T^c} + \lambda_1 \Gamma^1_{T^c} + \lambda_2 \Lambda_{T^c} = 0, \\ n_2 S^2_{T^c} + \lambda_1 \Gamma^2_{T^c} - \lambda_2 \Lambda_{T^c} = 0. \end{array} \quad (29)$$

Summing the two equations in (29) yields

$$(n_1 S^1_{T^c} + n_2 S^2_{T^c}) + \lambda_1 (\Gamma^1_{T^c} + \Gamma^2_{T^c}) = 0. \quad (30)$$

It thus follows from (30) that

$$\|n_1 S^1_{T^c} + n_2 S^2_{T^c}\|_\infty \le \lambda_1 \|\Gamma^1_{T^c} + \Gamma^2_{T^c}\|_\infty \le 2\lambda_1, \quad (31)$$

where here $\|\cdot\|_\infty$ indicates the maximal absolute element of a matrix, and where the second inequality in (31) follows from the fact that the subgradient of the ℓ_1 norm is bounded in absolute value by one.

We now assume, without loss of generality, that the Λ that solves (27) and (28) is symmetric. (In fact, one can easily show that there exist symmetric subgradients Γ^1, Γ^2, and Λ that satisfy (27) and (28).) Moreover, recall from Lemma 11 that $\|(\Lambda + \Lambda^T)_j\|_s \le 1$. Therefore, $\|\Lambda_j\|_s \le \frac{1}{2}$. Using Holder's inequality and noting that $\|y\|_1 = \langle y, \mathrm{sgn}(y)\rangle$ for a vector y, we obtain

$$\begin{array}{rcl} \|\Lambda_{T^c}\|_1 & = & \langle \Lambda_{T^c}, \mathrm{sgn}(\Lambda_{T^c})\rangle \\ & \le & \|\mathrm{sgn}(\Lambda_{T^c})\|_q \|\Lambda_{T^c}\|_s \\ & \le & |T^c|^{\frac{1}{q}} \|\Lambda_{T^c}\|_s \\ & \le & |T^c|^{\frac{1}{q}} \|\Lambda\|_s \\ & \le & \frac{1}{2}|T^c|^{\frac{1}{q}} p^{\frac{1}{s}}, \end{array} \quad (32)$$

where the last inequality follows from the fact that $\|\Lambda\|_s^s = \sum_{j=1}^p \|\Lambda_j\|_s^s \le p(\frac{1}{2})^s$, and where in (32), $\|A\|_q$ and $\|A\|_s$ indicate the ℓ_q and ℓ_s norms of vec(A) respectively.

From (29), we have for each $k \in \{1, 2\}$ that

$$\begin{aligned} n_k\|S^k_{T^c}\|_1 &\leq \|\lambda_1\Gamma^k_{T^c}+\lambda_2\Lambda_{T^c}\|_1 \\ &\leq \lambda_1\|\Gamma^k_{T^c}\|_1+\lambda_2\|\Lambda_{T^c}\|_1 \\ &\leq \lambda_1|T^c|+\lambda_2\frac{|T^c|^{\frac{1}{q}}p^{\frac{1}{s}}}{2}, \end{aligned}$$

where the last inequality follows from the fact that the elements of Γ^k are bounded in absolute value by one, and (32). The theorem now follows by noting from (29) that for each $k \in \{1, 2\}$,

$$n_k\|S^k_{T^c}\|_\infty \leq \lambda_1\|\Gamma^k_{T^c}\|_\infty+\lambda_2\|\Lambda_{T^c}\|_\infty \leq \lambda_1+\frac{\lambda_2}{2}.$$

Appendix C. Proof of Theorem 7

Proof

The optimality conditions for the CNJGL problem (7) are given by

$$-n_k(\Theta^k)^{-1}+n_kS^k+\lambda_1\Gamma^k+\lambda_2\Lambda^k=0, \quad k=1,\ldots,K, \quad (33)$$

where Γ^k is a subgradient of $\|\Theta^k\|_1$. Also, the K-tuple $(\Lambda^1, \ldots, \Lambda^K)$ is a subgradient of $\Omega_q(\Theta^1 - \text{diag}(\Theta^1), \ldots, \Theta^K - \text{diag}(\Theta^K))$, and the right-hand side is a $p \times p$ matrix of zeros. We can assume, without loss of generality, that the subgradients Γ^k and Λ^k that satisfy (33) are symmetric, since Lemma 11 indicates that if $(\Lambda^1, \ldots, \Lambda^K)$ is a subgradient of $\Omega_q(\Theta^1 - \text{diag}(\Theta^1), \ldots, \Theta^k - \text{diag}(\Theta^k))$, then $((\Lambda^1 + (\Lambda^1)^T)/2, \ldots, (\Lambda^K + (\Lambda^K)^T)/2)$ is a subgradient as well.

Now suppose that $\Theta^1, \ldots, \Theta^K$ that solve (7) are supported on T. Since $(\ldots^k)^{-1}$ is supported on T for all k, we have

$$n_kS^k_{T^c}+\lambda_1\Gamma^k_{T^c}+\lambda_2\Lambda^k_{T^c}=0. \quad (34)$$

We use the triangle inequality for the ℓ_1 norm (applied elementwise to the matrix) to get

$$n_k\|S^k_{T^c}\|_1 \leq \lambda_1\|\Gamma^k_{T^c}\|_1+\lambda_2\|\Lambda^k_{T^c}\|_1. \quad (35)$$

We have $\|\Gamma^k\|_\infty \leq 1$ since Γ^k is a subgradient of the ℓ_1 norm, which gives $\|\Gamma^k_{T^c}\|_1 \leq |T^c|$.

Also Γ^k is a part of a subgradient to Ω_q, so by Lemma 11, $\|(\Lambda^k + (\Lambda^k)^T)_j\|_s \leq 1$ for $j \in \{1, 2, \ldots, p\}$. Since Λ^k is symmetric, we have that $\|\Lambda^k_j\|_s \leq \frac{1}{2}$. Using the same reasoning as in (32) of Appendix B, we obtain

$$\|\Lambda^k_{T^c}\|_1 \le \frac{1}{2}|T^c|^{\frac{1}{q}}p^{\frac{1}{s}}. \quad (36)$$

Combining (35) and (36) yields

$$n_k\|S^k_{T^c}\|_1 \le \lambda_1|T^c|+\tfrac{\lambda_2}{2}|T^c|^{\frac{1}{q}}p^{\frac{1}{s}}.$$

The theorem follows by noting from (34) that

$$n_k\|S^k_{T^c}\|_\infty \le \lambda_1\|\Gamma^k_{T^c}\|_\infty+\lambda_2\|\Lambda^k_{T^c}\|_\infty \le \lambda_1+\tfrac{\lambda_2}{2}.$$

Appendix D. Proof of Theorem 10

Assume that the sufficient condition holds. In order to prove the theorem, we must show that

$$\begin{array}{c}\sum_{k=1}^{K} n_k(-\mathrm{logdet}(\Theta^k)+\langle\Theta^k,S^k\rangle))+\lambda_1\sum_{k=1}^{k}\|\Theta^k\|_1+\lambda_2 h(\Theta^1,\dots,\Theta^K)\\ >\sum_{k=1}^{K} n_k(-\mathrm{logdet}(\Theta^k_T)+\langle\Theta^k_T,S^k\rangle))+\lambda_1\sum_{k=1}^{k}\|\Theta^k_T\|_1+\lambda_2 h(\Theta^1_T,\dots,\Theta^K_T).\end{array}$$

By assumption,

$$h(\Theta^1,\dots,\Theta^K)>h(\Theta^1_T,\dots,\Theta^K_T). \quad (37)$$

We will now show that

$$n_k\langle\Theta^k,S^k\rangle+\lambda_1\|\Theta^k\|_1 \ge n_k\langle\Theta^k_T,S^k\rangle+\lambda_1\|\Theta^k_T\|_1, \quad (38)$$

or equivalently, that

$$-n_k\langle\Theta^k_{T^c},S^k\rangle \le \lambda_1\|\Theta^k_{T^c}\|_1. \quad (39)$$

Note that $\langle\Theta^k_{T^c},S^k\rangle=\langle\Theta^k_{T^c},S^k_{T^c}\rangle$. By the sufficient condition, $n_k\|S^k_{T^c}\|_\infty \le \lambda_1$. So

$$\begin{array}{rcl}-n_k\langle\Theta^k_{T^c},S^k\rangle & = & -n_k\langle\Theta^k_{T^c},S^k_{T^c}\rangle\\ & \le & \|n_kS^k_{T^c}\|_\infty\|\Theta^k_{T^c}\|_1\\ & \le & \lambda_1\|\Theta^k_{T^c}\|_1.\end{array}$$

So (39) holds, and hence (38) holds.

Finally, we apply Fischer's inequality, which states that $\det(\Theta^k) \leq \det(\Theta_T^k)$, and so

$$-\text{logdet}(\Theta^k) \geq -\text{logdet}(\Theta_T^k). \quad (40)$$

Combining (37), (38), and (40), the theorem holds.

Appendix E. Connection Between RCON and Obozinski et al. (2011)

We now show that the RCON penalty with $q = 2$ can be derived from the overlap norm of Obozinski et al. (2011). For simplicity, here we restrict ourselves to the RCON with $K = 1$. The general case of $K \geq 1$ can be shown via a simple extension of this argument.

Given any symmetric $p \times p$ matrix Θ, let Θ_Δ be the $p \times p$ upper-triangular matrix such that $\Theta = \Theta_\Delta + \Theta_\Delta^T$. That is,

$$(\Theta_\Delta)_{kl} = \begin{cases} \Theta_{kl} & \text{if } k<l \\ \Theta_{kk}/2 & \text{if } k=l \\ 0 & \text{if } k>l. \end{cases} \quad (41)$$

Now define p groups, $g_1, \ldots, g_p$, each of which contains p variables, as displayed in Figure 15. Note that these groups overlap: if $k \leq l$, then the (k, l) element of a matrix is contained in both the kth and lth groups.

The overlap norm corresponding to these groups is given by

$$\begin{aligned} \Omega^O(\Theta) = & \min_{V^1,\ldots,V^p \in \mathbb{R}^{p\times p}} && \sum_{j=1}^p \|V^j\|_F \\ & \text{subject to} && \Theta_\Delta = \sum_{j=1}^p V^j, \text{supp}(V^j) \subseteq g_j, \end{aligned}$$

where the relation between Θ and Θ_Δ is as in Equation (41). We can rewrite this as

$$\begin{aligned} \Omega^O(\Theta) = & \min_{V^1,\ldots,V^p \in \mathbb{R}^{p\times p}} && \sum_{j=1}^p \|V^j\|_F \\ & \text{subject to} && \Theta = \sum_{j=1}^p V^j + \left(\sum_{j=1}^p V^j\right)^T, \text{supp}(V^j) \subseteq g_j. \end{aligned} \quad (42)$$

Now, define a $p \times p$ matrix A such that

$$A_{ij} = \begin{cases} (V^j)_{ij} & \text{if } i<j \\ (V^j)_{ji} & \text{if } i>j \\ (V^j)_{jj} & \text{if } i=j \end{cases}.$$

Note that $A+A^T=\sum_{j=1}^{p}V^j+(\sum_{j=1}^{p}V^j)^T$. Furthermore, $\|V^j\|_F = \|A_j\|_2$, where A_j denotes the jth column of A. So we can rewrite (42) as

$$\Omega^O(\Theta)=\begin{array}{ll}\underset{V^1,\ldots,V^p\in\mathbb{R}^{p\times p}}{\min} & \sum_{j=1}^{p}\|A_j\|_2\\ \text{subject to} & \Theta=A+A^T.\end{array}$$

This is exactly the RCON penalty with $K = 1$ and $q = 2$. Thus, with a bit of work, we have derived the RCON from the overlap norm (Obozinski et al., 2011). Our penalty is useful because it accommodates groups given by the rows and columns of a symmetric matrix in an elegant and convenient way.

Appendix F. Derivation of Updates for ADMM Algorithms

We derive the updates for ADMM algorithm when applied to PNJGL and CNJGL formulations respectively. We first begin with the PNJGL formulation.

F.1 Updates for ADMM Algorithm for PNJGL

Let $\mathscr{L}_\rho$ $(\Theta^1, \Theta^2, Z^1, Z^2, V, W, F, G, Q^1, Q^2)$ denote the augmented Lagrangian (12). In each iteration of the ADMM algorithm, each primal variable is updated while holding the other variables fixed. The dual variables are updated using a simple dual-ascent update rule. Below, we derive the update rules for the primal variables.

F.1.1 Θ^1 Update

Note that

$$\begin{aligned}\Theta^1 &= \underset{\Theta}{\text{argmin}}\quad \mathscr{L}_\rho(\Theta,\Theta^2,Z^1,Z^2,V,W,F,G,Q^1,Q^2)\\ &= \underset{\Theta}{\text{argmin}}\quad n_1(-\text{logdet}\Theta)+\rho\|\Theta-\tfrac{1}{2}\left((\Theta^2+V+W+Z^1)-\tfrac{1}{\rho}(F+Q^1+n_1S^1)\right)\|_F^2.\end{aligned}$$

Now it follows from the definition of the Expand operator that

$$\Theta^1 \leftarrow \text{Expand}\left(\frac{1}{2}(\Theta^2+V+W+Z^1)-\frac{1}{2\rho}(Q^1+n_1S^1+F),\rho,n_1\right).$$

The update for Θ^2 can be derived in a similar fashion.

F.1.2 *Z*^{1} Update

$$\begin{aligned} Z^1 &= \underset{Z}{\operatorname{argmin}} \ \mathscr{L}_\rho(\Theta^1, \Theta^2, Z, Z^2, V, W, F, G, Q^1, Q^2) \\ &= \underset{Z}{\operatorname{argmin}} \ \tfrac{1}{2}\|Z^1 - (\Theta^1 + \tfrac{Q^1}{\rho})\|_F^2 + \tfrac{\lambda_1}{\rho}\|Z^1\|_1. \end{aligned}$$

By the definition of the soft-thresholding operator $\mathscr{T}_1$, it follows that

$$Z^1 = \mathscr{T}_1\left(\Theta^1 + \frac{Q^1}{\rho}, \frac{\lambda_1}{\rho}\right).$$

The update for Z^2 is similarly derived.

F.1.3 *V* Update

$$\begin{aligned} V &= \underset{X}{\operatorname{argmin}} \ \mathscr{L}_\rho(\Theta^1, \Theta^2, Z^1, Z^2, X, W, F, G, Q^1, Q^2) \\ &= \underset{X}{\operatorname{argmin}} \ \tfrac{\lambda_2}{2\rho}\sum_{j=1}^{p}\|X_j\|_q + \tfrac{1}{2}\|X - \tfrac{1}{2}\left((W^T + \Theta^1 - \Theta^2 - W) + \tfrac{1}{\rho}(F - G)\right)\|_F^2. \end{aligned}$$

By the definition of the soft-scaling operator $\mathscr{T}_2$, it follows that

$$V = \mathscr{T}_2\left(\frac{1}{2}(W^T - W + \Theta^1 - \Theta^2) + \frac{1}{2\rho}(F - G), \frac{\lambda_2}{2\rho}\right).$$

The update for *W* is easy to derive and we therefore skip it.

F.2 Updates for ADMM Algorithm for CNJGL

Let $\mathscr{L}_\rho(\{\Theta^i\}, \{Z^i\}, \{\tilde{V}^i\}, \{W^i\}, \{F^i\}, \{G^i\}, \{Q^i\})$ denote the augmented Lagrangian (16). Below, we derive the update rules for the primal variables $\{\tilde{V}^i\}$. The update rules for the other primal variables are similar to the derivations discussed for PNJGL, and hence we omit their derivations.

The update rules for $\tilde{V}^1, \tilde{V}^2, \ldots, \tilde{V}^K$ are coupled, so we derive them simultaneously. Note that

$$\begin{aligned}\left\{\tilde{V}^i\right\}_{i=1}^K &= \underset{A^1,\ldots,A^K}{\operatorname{argmin}} \; \mathscr{L}_\rho\left(\{\Theta^i\}_{i=1}^K, \{Z^i\}_{i=1}^K, \{A^i\}_{i=1}^K, \{W^i\}_{i=1}^K, \{F^i\}_{i=1}^K, \{G^i\}_{i=1}^K, \{Q^i\}_{i=1}^K\right)\\ &= \underset{A^1,\ldots,A^K}{\operatorname{argmin}} \; \lambda_2 \sum_{j=1}^{p} \left\| \begin{bmatrix} A^1 - \operatorname{diag}(A^1) \\ \vdots \\ A^K - \operatorname{diag}(A^K) \end{bmatrix}_j \right\|_q + \\ &\qquad \rho \sum_{i=1}^{K} \left\| A^i - \frac{1}{2}\left((W^i)^T + \Theta^i - W^i + \frac{1}{\rho}(F^i - G^i)\right) \right\|_F^2 .\end{aligned}$$

Let $C^i = \frac{1}{2}((W^i)^T + \Theta^i - W^i + \frac{1}{\rho}(F^i - G^i))$. Then the update

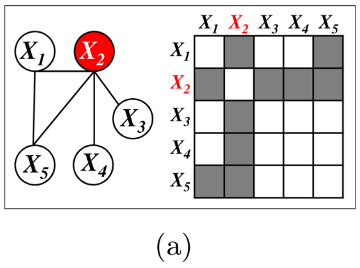
$$\begin{bmatrix} \tilde{V}^1 \\ \vdots \\ \tilde{V}^K \end{bmatrix} \leftarrow \mathscr{T}_q\left(\begin{bmatrix} C^1 - \operatorname{diag}(C^1) \\ \vdots \\ C^K - \operatorname{diag}(C^K) \end{bmatrix}, \frac{\lambda_2}{2\rho}\right) + \begin{bmatrix} \operatorname{diag}(C^1) \\ \vdots \\ \operatorname{diag}(C^K) \end{bmatrix}$$

follows by inspection.

Appendix G. Additional Simulation Results

Here we present more detailed results for an instance of the simulation study described in Section 6, for the case $n = 25$. Figure 16 illustrates how the PPC, TPPC, PCC and TPCC metrics are computed. As described in Table 1, for PNJGL, PPC is given by the number of columns of $\hat{V}$ whose ℓ_2 norms exceed the threshold t_s. Figure 16(a) indicates that the two perturbed nodes in the data are identified as perturbed by PNJGL. Furthermore, given the large gap between the perturbed and non-perturbed columns, PPC is relatively insensitive to the choice of t_s. Similar results apply to the TPPC, PCC and TPCC metrics.

In order to generate Figure 16, PNJGL, FGL, CNJGL, GGL, and GL were performed using tuning parameter values that led to the best identification of perturbed and cohub nodes. However, the results displayed in Figure 16 were quite robust to the choice of tuning parameter.



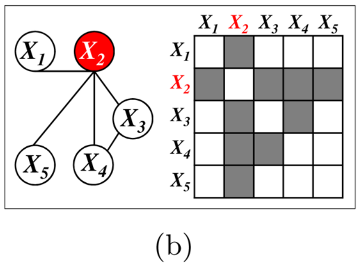


Figure 1.

Two networks share a *common hub* (co-hub) node. X_2 serves as a hub node in both networks. (*a*): Network 1 and its adjacency matrix. (*b*): Network 2 and its adjacency matrix.

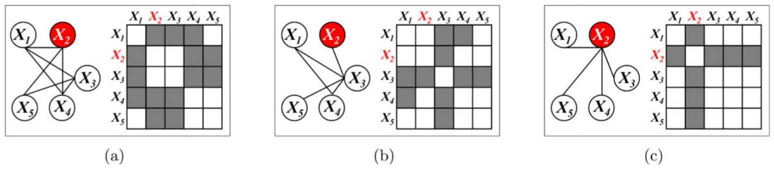
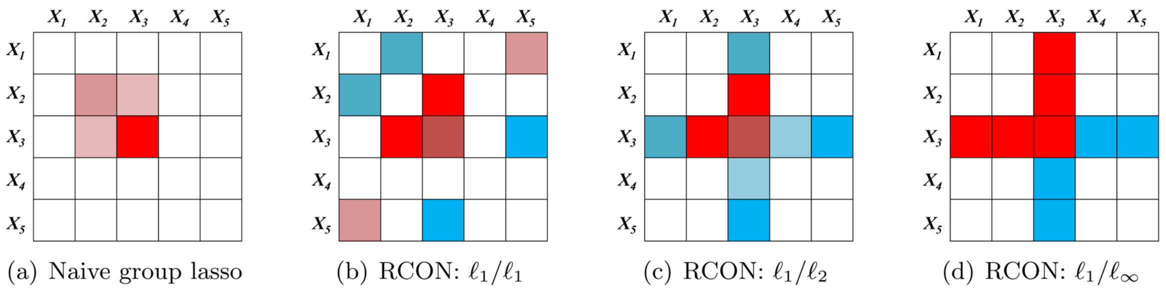


Figure 2.

Two networks that differ due to*node perturbation* of X_2. (*a*): Network 1 and its adjacency matrix. (*b*): Network 2 and its adjacency matrix. (*c*)*: Left:* Edges that differ between the two networks. *Right:* Shaded cells indicate edges that differ between Networks 1 and 2.



X_1 X_2 X_3 X_4 X_5
(a) Naive group lasso
(b) RCON: ℓ_1/ℓ_1
(c) RCON: ℓ_1/ℓ_2
(d) RCON: ℓ_1/ℓ_∞

Figure 3.

Toy example of the results from applying various penalties in order to estimate a 5×5 matrix, under a symmetry constraint. Zero elements are shown in white; nonzero elements are shown in shades of red (positive elements) and blue (negative elements). (*a*): The naive group lasso applied to the columns of the matrix yields non-zero elements that are the *intersection*, rather than the *union*, of a set of rows and columns. (*b*): The RCON penalty using an ℓ_1/ℓ_1 norm results in unstructured sparsity in the estimated matrix. (*c*): The RCON penalty using an ℓ_1 / ℓ_2 norm results in entire rows and columns of non-zero elements. (*d*): The RCON penalty using an ℓ_1/ℓ_∞ norm results in entire rows and columns of non-zero elements; many take on a single maximal (absolute) value.

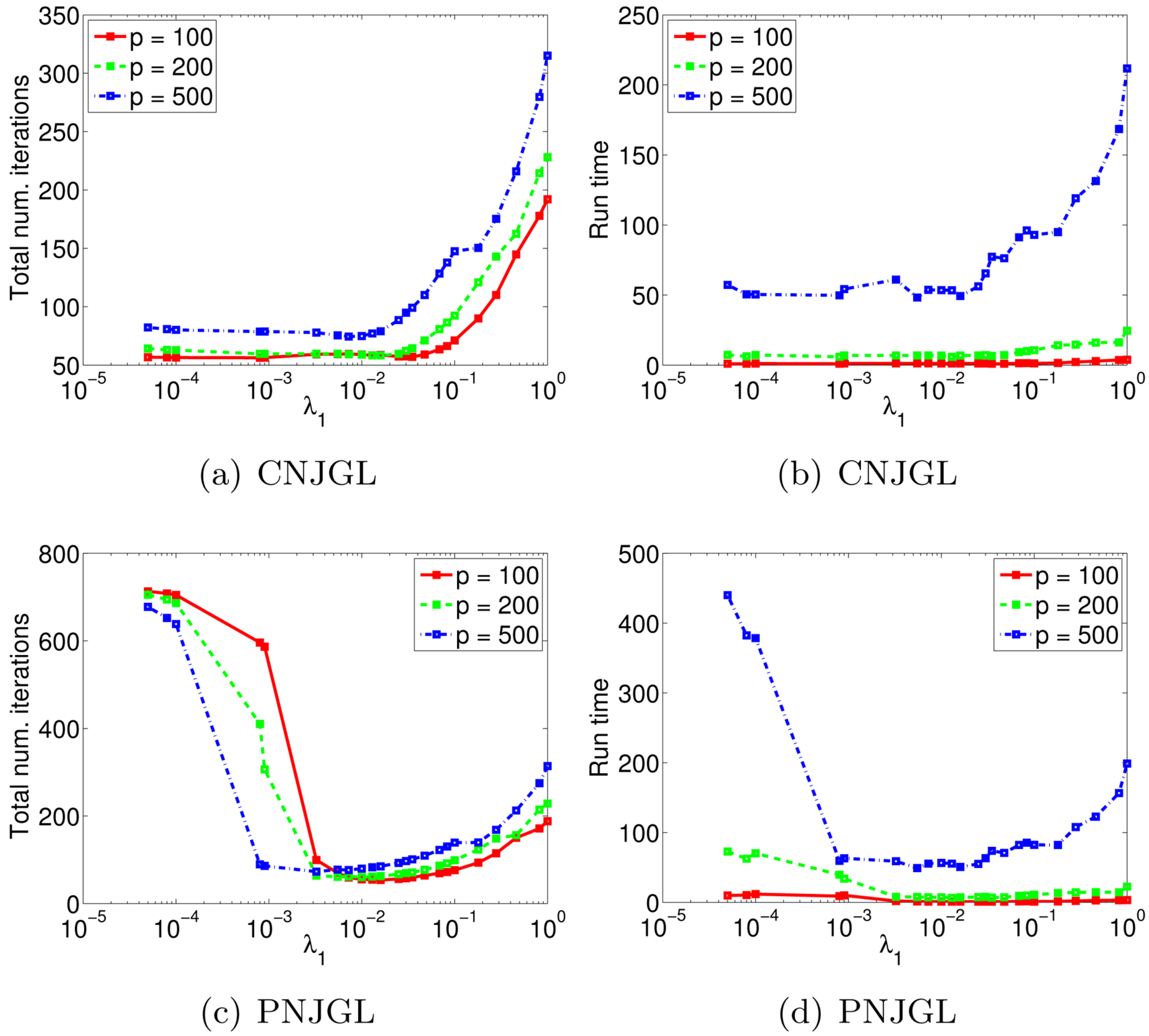


Figure 4.

(*a*): The total number of iterations for the CNJGL algorithm, as a function of λ_1. (*b*): Run-time (in seconds) of the CNJGL algorithm, as a function of λ_1. (*c*)–(*d*): As in (a)–(b), but for the PNJGL algorithm. All results are averaged over 20 random generations of synthetic data.

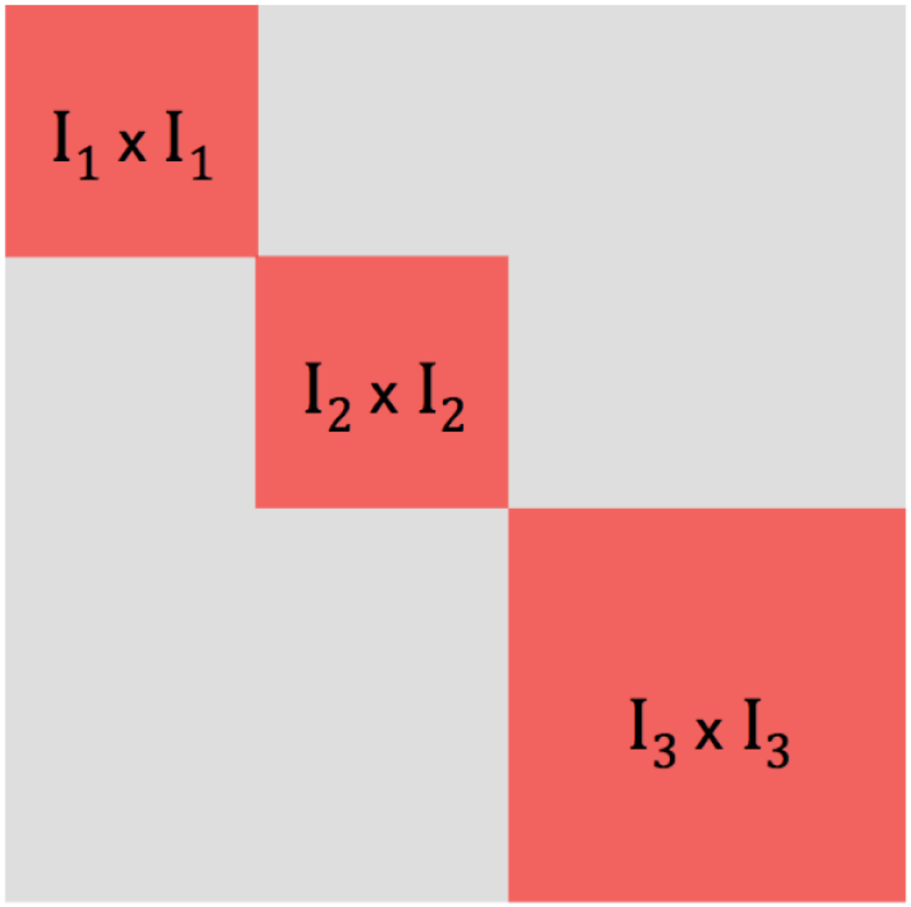


Figure 5.

A $p \times p$ matrix is displayed, for which I_1, I_2, I_3 denote a partition of the index set $\{1, 2, \ldots, p\}$. $T = \cup_{i=1}^{L}\{I_i \times I_i\}$ is shown in red, and T^c is shown in gray.

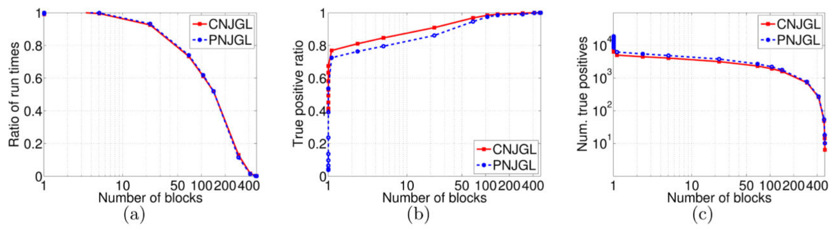


Figure 6.
Speed-ups for CNJGL and PNJGL on a simulation set-up with p = 500 and n_1 = n_2 = 250. The true inverse covariance matrices are block-diagonal with two equally-sized sparse blocks. The x-axis in each panel displays the number of blocks into which the CNJGL or PNJGL problems are decomposed using the sufficient conditions; this is a surrogate for λ_1. The y-axes display *(a)*: the ratio of run-times with and without the sufficient conditions; *(b)*: the true positive ratio of the edges estimated; and *(c)*: the total number of true positive edges estimated.

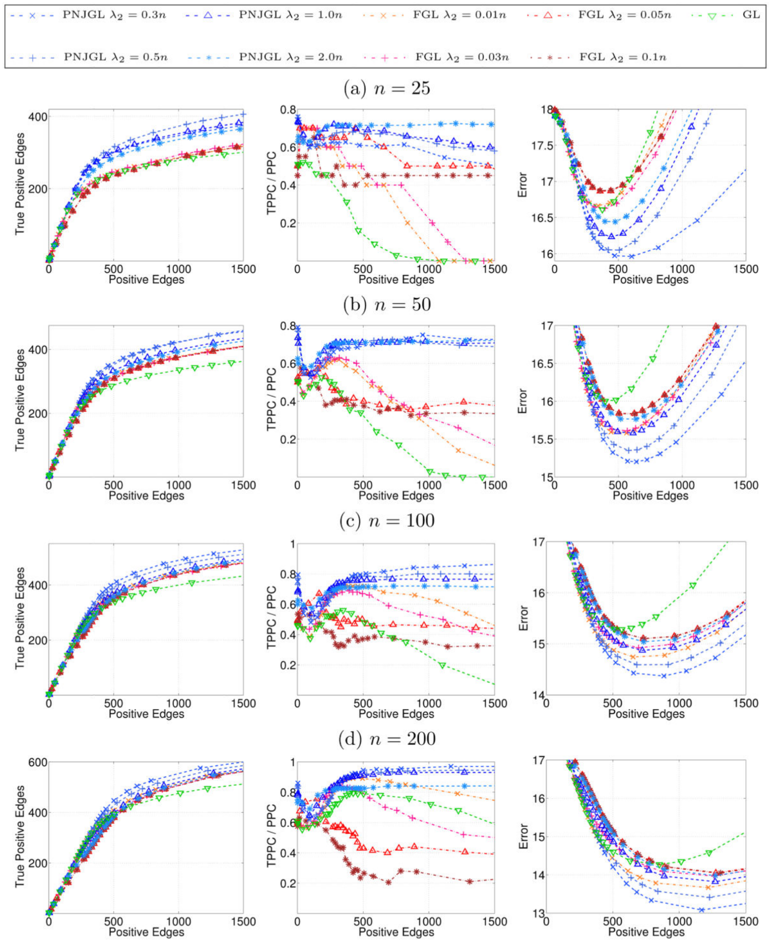


Figure 7.

Simulation results on Erdos-Renyi network (Section 6.1.1) for PNJGL with $q = 2$, FGL, and GL, for *(a):* $n = 25$, *(b):* $n = 50$, *(c):* $n = 100$, *(d):* $n = 200$, when $p = 100$. Each colored line corresponds to a fixed value of λ_2, as λ_1 is varied. Axes are described in detail in Table 1. Results are averaged over 100 random generations of the data.

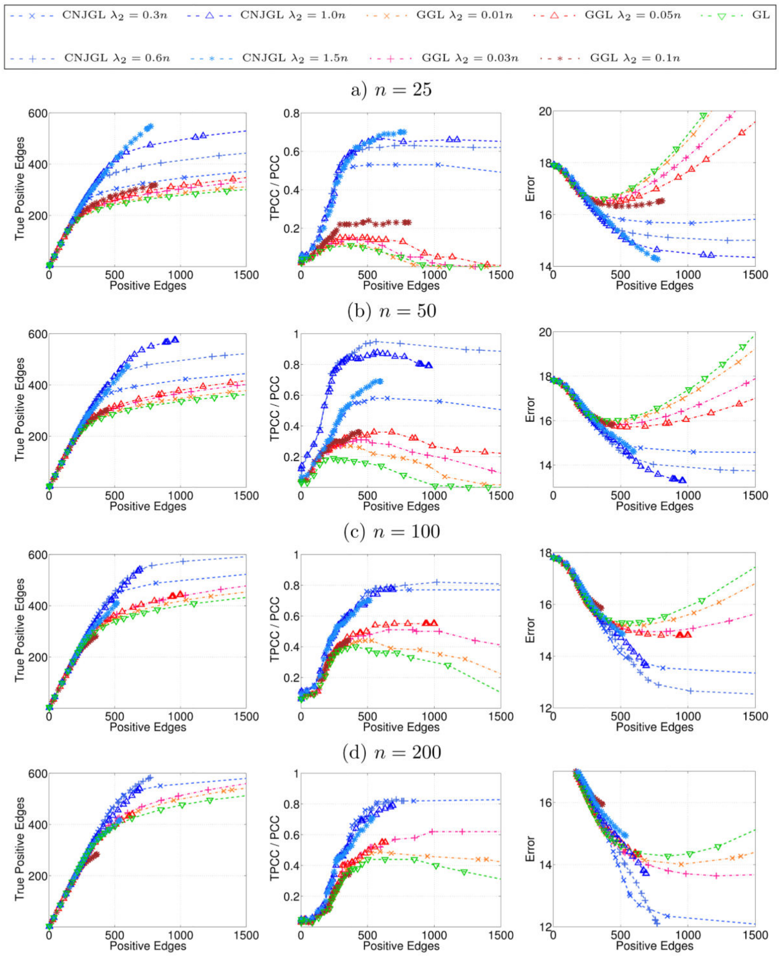
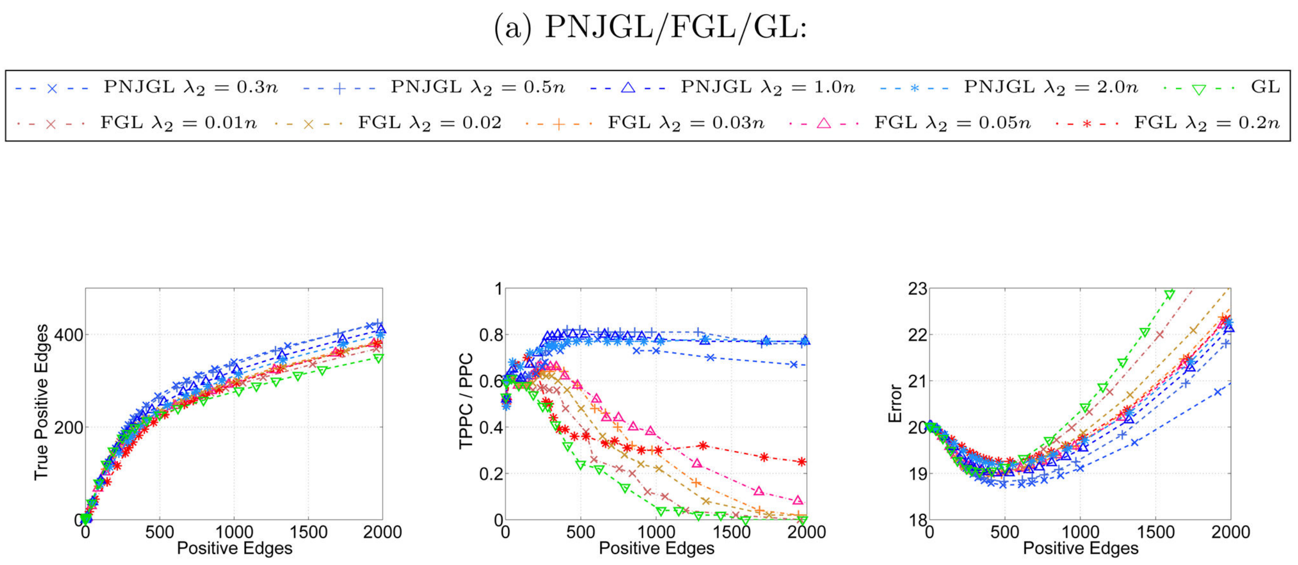


Figure 8.

Simulation results on Erdos-Renyi network (Section 6.1.1) for CNJGL with $q = 2$, GGL, and GL, for *(a):* $n = 25$, *(b):* $n = 50$, *(c):* $n = 100$, *(d):* $n = 200$, when $p = 100$. Each colored line corresponds to a fixed value of λ_2, as λ_1 is varied. Axes are described in detail in Table 1. Results are averaged over 100 random generations of the data.



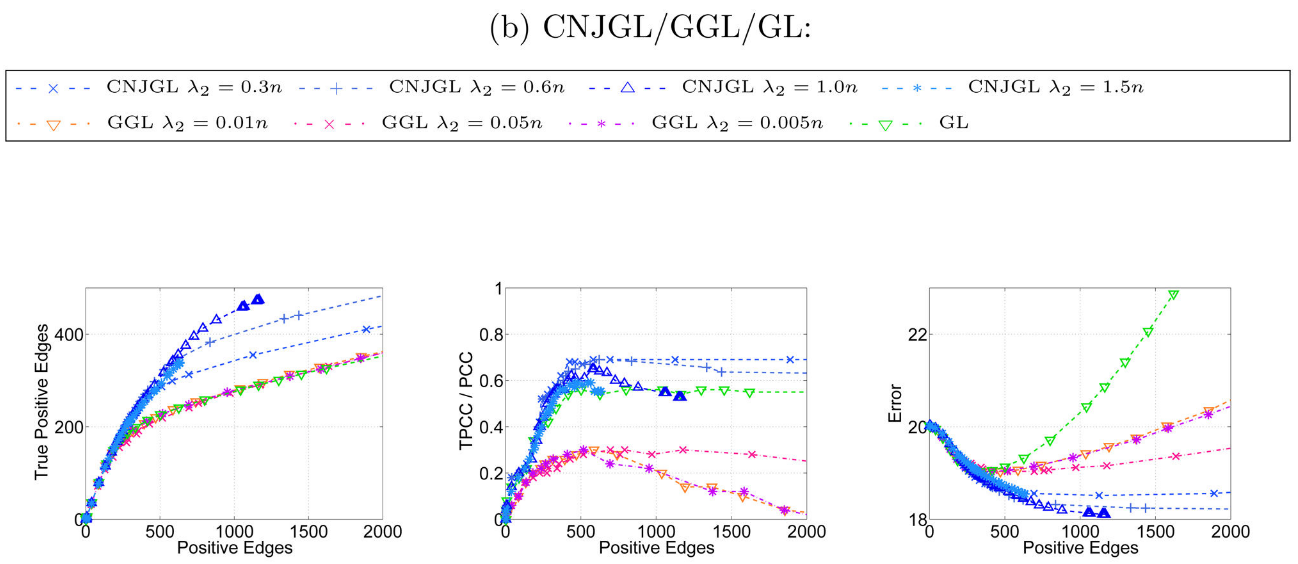


Figure 9.

Simulation results on scale-free network (Section 6.1.2) for *(a):* PNJGL with $q = 2$, FGL, and GL, and *(b):* CNJGL with $q = 2$, GGL, and GL, with $p = 100$ and $n = 50$. Each colored line corresponds to a fixed value of λ_2, as λ_1 is varied. Axes are described in detail in Table 1. Results are averaged over 50 random generations of the data.

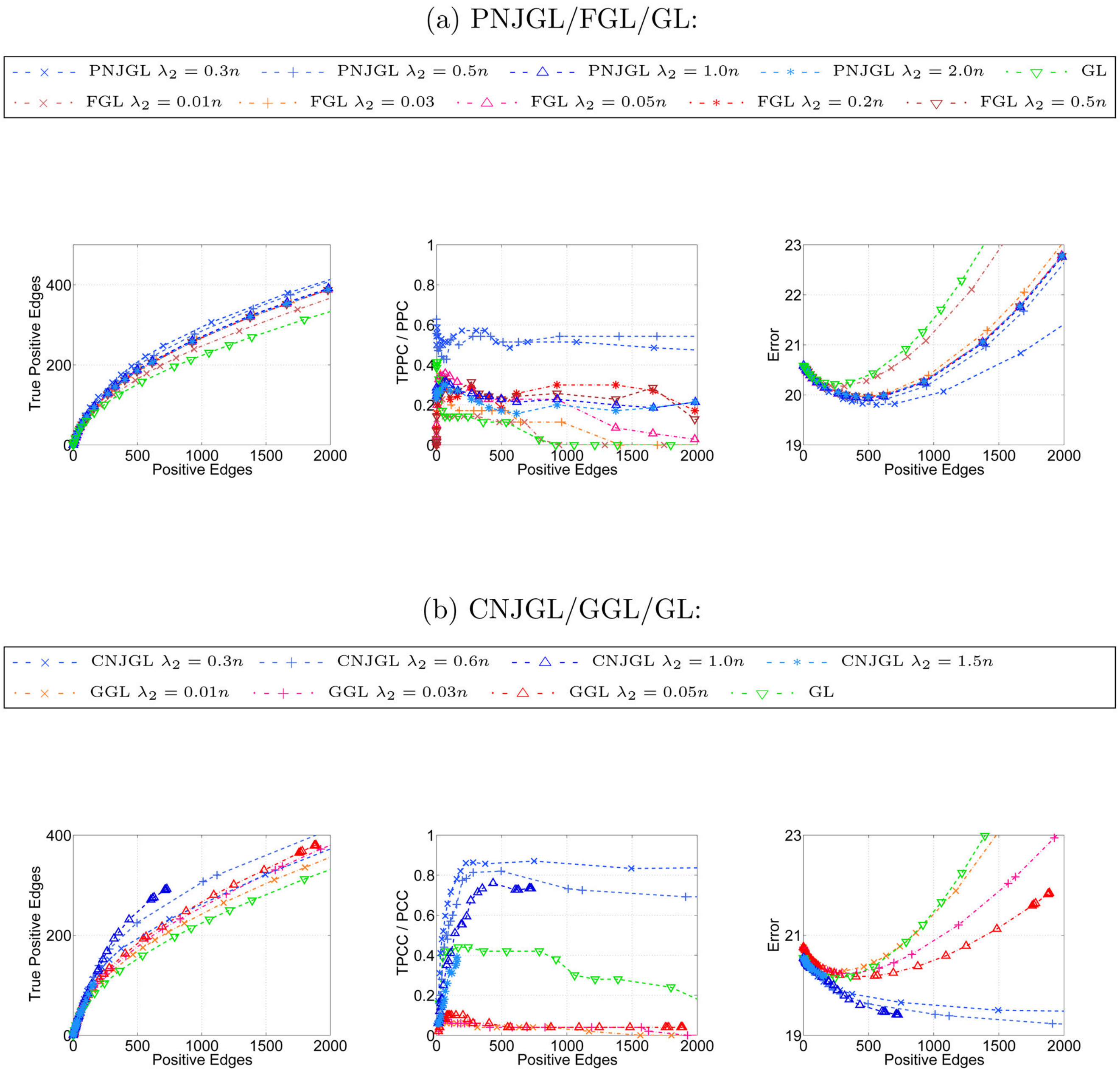


Figure 10.

Simulation results on community network (Section 6.1.3) for *(a):* PNJGL with $q = 2$, FGL, and GL, and *(b):* CNJGL with $q = 2$, GGL, and GL, with $p = 100$ and $n = 50$. Each colored line corresponds to a fixed value of λ_2, as λ_1 is varied. Axes are described in detail in Table 1. Results are averaged over 50 random generations of the data.

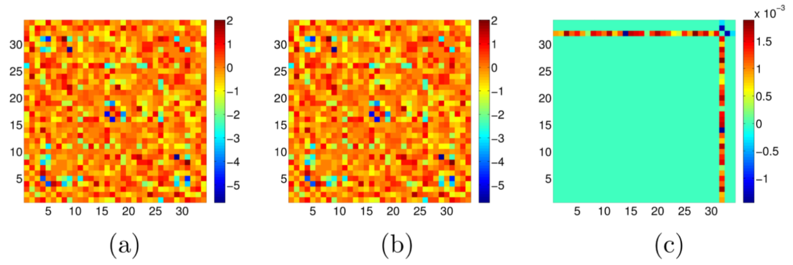
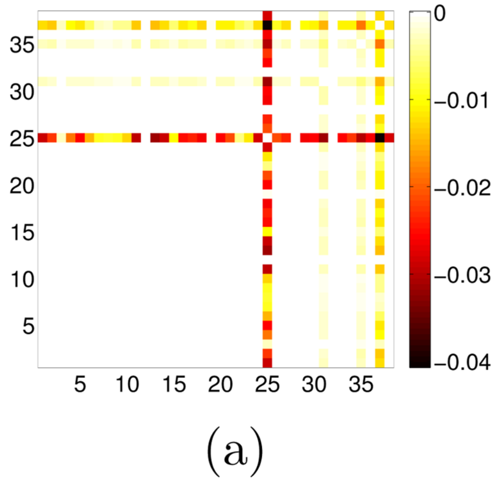


Figure 11.

GBM data analysis for PNJGL with $q = 2$. The sample covariance matrices S^1 and S^2 were generated from samples with two cancer subtypes, with sizes $n_1 = 53$ and $n_2 = 56$. Only the 34 genes contained in the Reactome "TCR Signaling" pathway were included in this analysis. Of these genes, three are frequently mutated in GBM: PTEN, PIK3R1, and PIK3CA. These three genes correspond to the last three columns in the matrices displayed (columns 32 through 34). PNJGL was performed with $\lambda_1 = 0$ and $\lambda_2 = 2$. We display *(a):* the estimated matrix $\hat{\Theta^1}$; *(b):* the estimated matrix $\hat{\Theta^2}$; and *(c):* the difference matrix $\hat{\Theta^1} - \hat{\Theta^2}$. The gene PTEN is identified as perturbed.



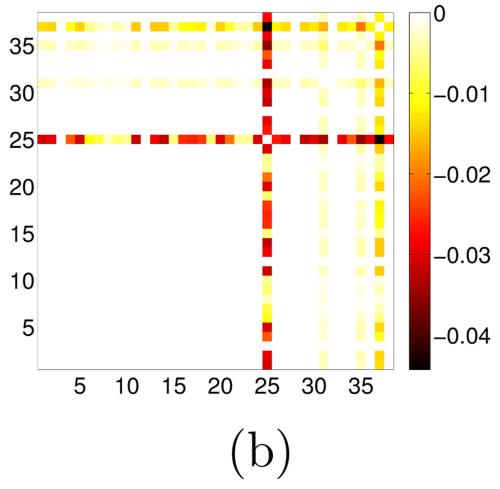


Figure 12.
GBM data analysis for CNJGL with $q = 2$. The sample covariance matrices S^1 and S^2 were generated from samples with two cancer subtypes, with sizes $n_1 = 53$ and $n_2 = 56$. Only the 38 genes contained in the Reactome “G2/M checkpoints” pathway were included in this analysis. Of these genes, 15 have been previously identified as regulators. These 15 genes correspond to the last 15 columns in the matrices (columns 24 through 38). CNJGL was performed with $\lambda_1 = 13$ and $\lambda_2 = 410$. We display *(a):* the estimated matrix $\Theta^{\hat{1}}$; *(b):* the estimated matrix $\Theta^{\hat{2}}$. Four of the regulator genes are identified by CNJGL. These genes are CDC6, MCM6, CCNB1, and CCNB2.

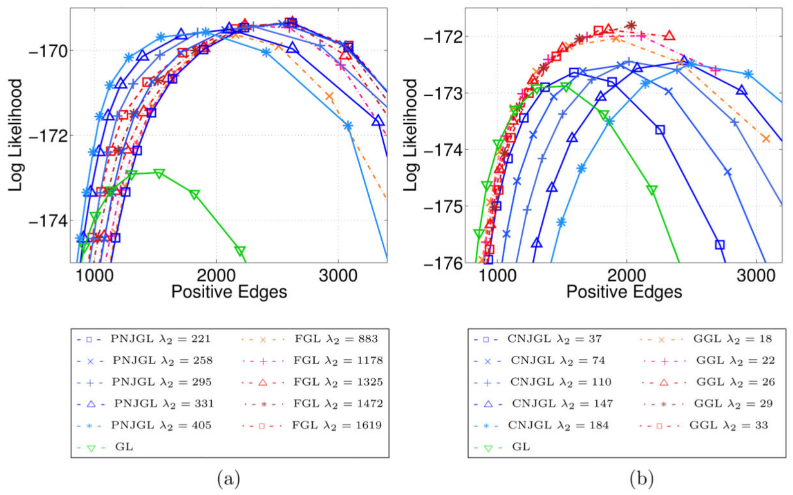


Figure 13.

On the webpage data, five-fold cross-validation was performed for *(a):* PNJGL, FGL, and GL; and *(b):* CNJGL, GGL, and GL. Each colored line corresponds to a fixed value of λ_2, as λ_1 is varied. Positive edges are defined in Table 1. The cross-validated log likelihood is displayed.

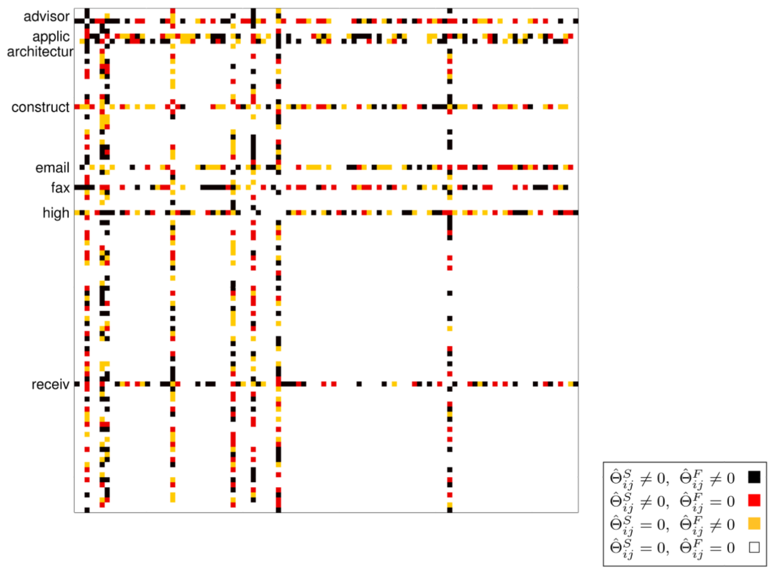


Figure 14.

Student and faculty webpage precision matrices, $\Theta^{\hat{S}}$ and $\Theta^{\hat{F}}$, for PNJGL performed with $\lambda_1 = 27$, $\lambda_2 = 381$. Eight perturbed nodes are labeled. The color of each square in the figure indicates whether the corresponding edge is present in both networks, absent in both networks, or present in only the student or only the faculty network.

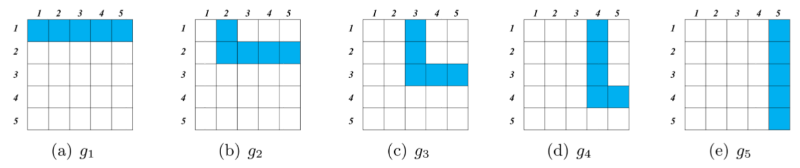


Figure 15.

Depiction of groupsg_1, …, g_5 for a 5 × 5 matrix. Each group's elements are shown in blue.

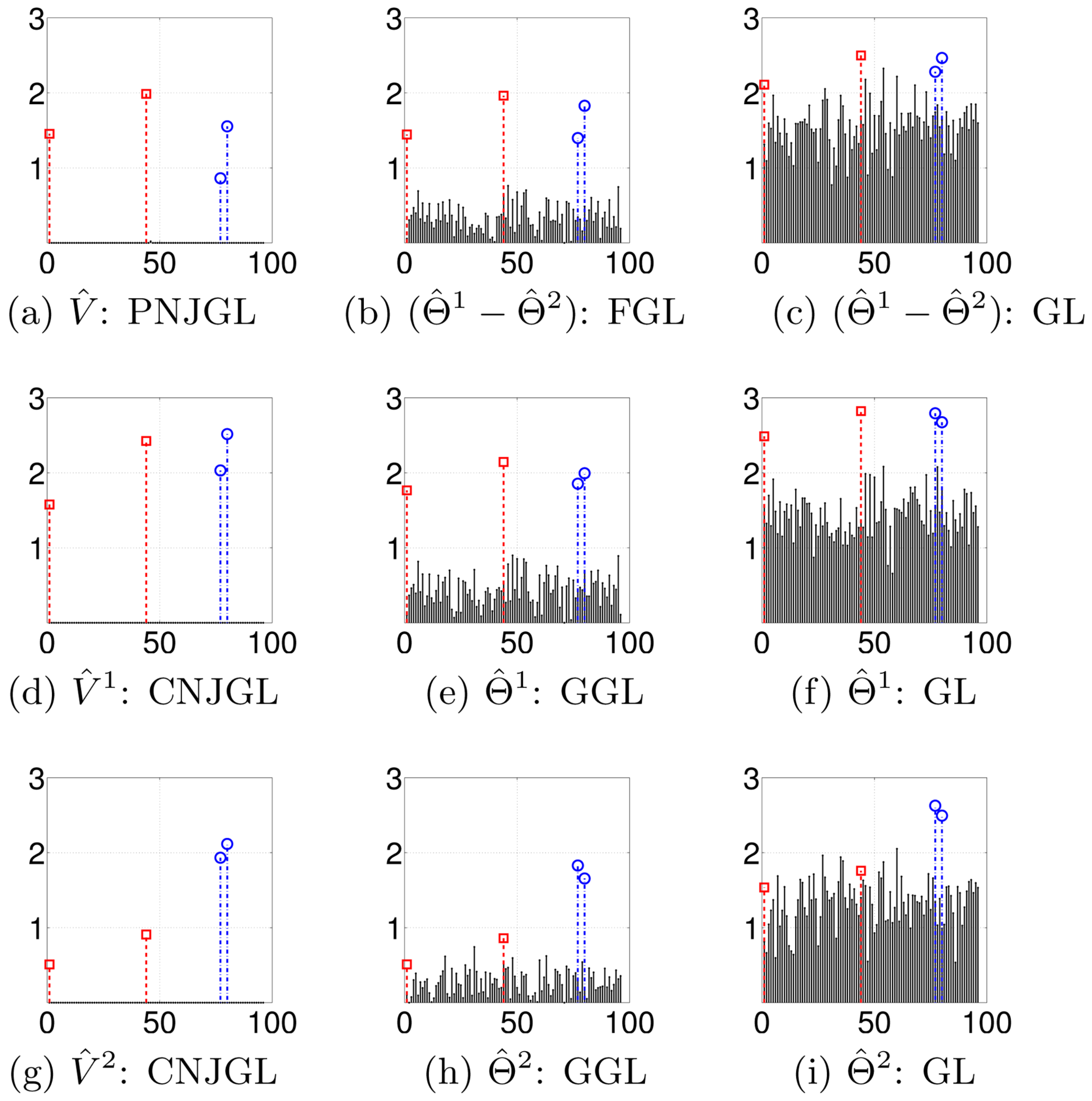


Figure 16.

In all plots, the *x*-axis indexes the columns of the indicated matrix, and the *y*-axis displays the ℓ_2 norms of the columns of the indicated matrix, with diagonal elements removed. The sample size is $n = 25$. Perturbed nodes are indicated in red (with square markers), and cohub nodes are indicated in blue (with circle markers). *(a)–(c):* Detection of perturbed nodes by PNJGL with $q = 2$, FGL, and GL. *(d)–(i):* Detection of cohub nodes by CNJGL with $q = 2$, GGL, and GL. *(a):* PNJGL with $q = 2$ was performed with $\lambda_1 = 2.5$ and $\lambda_2 = 12.5$. *(b):* FGL was performed with $\lambda_1 = 2.5$ and $\lambda_2 = 0.75$. *(c):* GL was performed with $\lambda = 1.5$. *(d)*, *(g):*

CNJGL was performed with $q = 2$ and $\lambda_1 = 0.5$, $\lambda_2 = 37.5$. *(e)*, *(h):* GGL was performed with $\lambda_1 = 0.5$ and $\lambda_2 = 2.5$. *(f)*, *(i):* GL was performed with $\lambda = 0.75$.

Table 1

Metrics used to quantify algorithm performance. Here Θ^1 and Θ^2 denote the true inverse covariance matrices, and $\hat{\Theta^1}$ and $\hat{\Theta^2}$ denote the two estimated inverse covariance matrices. Here $1\{A\}$ is an indicator variable that equals one if the event A holds, and equals zero otherwise. (1) Metrics based on recovery of the support of Θ^1 and Θ^2. Here $t_0 = 10^{-6}$. (2) Metrics based on identification of perturbed nodes and co-hub nodes. The metrics PPC and TPPC quantify node perturbation, and are applied to PNJGL, FGL, and GL. The metrics PCC and TPCC relate to co-hub detection, and are applied to CNJGL, GGL, and GL. We let $t_s = \mu + 5.5\sigma$, where μ is the mean and σ is the standard deviation of $\{\|\hat{V}_{-i,i}\|_2\}_{i=1}^p$ (PPC or TPPC for PNJGL), $\{\|(\hat{\Theta}^1 - \hat{\Theta}^2)_{-i,i}\|_2\}_{i=1}^p$ (PPC or TPPC for FGL/GL), $\{\|\hat{V}^1_{-i,i}\|_2\}_{i=1}^p$ and $\{\|\hat{V}^2_{-i,i}\|_2\}_{i=1}^p$ (PPC or TPPC for CNJGL), or $\{\|\hat{\Theta}^1_{-i,i}\|_2\}_{i=1}^p$ and $\{\|\hat{\Theta}^2_{-i,i}\|_2\}_{i=1}^p$ (PPC or TPPC for GGL/GL). However, results are very insensitive to the value of t_s, as is shown in Appendix G. (3) Frobenius error of estimation of Θ^1 and Θ^2.

(1)	<u>Positive edges:</u> $\sum_{i<j}\left(1\{\lvert\hat{\Theta}^1_{ij}\rvert > t_0\} + 1\{\lvert\hat{\Theta}^2_{ij}\rvert > t_0\}\right)$ <u>True positive edges:</u> $\sum_{i<j}\left(1\{\lvert\Theta^1_{ij}\rvert > t_0 \text{ and } \lvert\hat{\Theta}^1_{ij}\rvert > t_0\} + 1\{\lvert\Theta^2_{ij}\rvert > t_0 \text{ and } \lvert\hat{\Theta}^2_{ij}\rvert > t_0\}\right)$
(2)	<u>Positive perturbed columns</u> (PPC): $\text{PNJGL:} \sum_{i=1}^p 1\left\{\|\hat{V}_{-i,i}\|_2 > t_s\right\};$ $\text{FGL/GL:} \sum_{i=1}^p 1\{\|(\hat{\Theta}^1 - \hat{\Theta}^2)_{-i,i}\|_2 > t_s\}$ <u>True positive perturbed columns</u> (TPPC): PNJGL: $\Sigma_{i \in I_p} 1\{\|\hat{V}_{-i,i}\|_2 > t_s\}$; FGL/GL: $\Sigma_{i \in I_p} 1\{\|(\hat{\Theta^1} - \hat{\Theta^2})_{-i,i}\|_2 > t_s\}$, where I_P is the set of perturbed column indices. <u>Positive co-hub columns</u> (PCC): $\text{CNJGL:} \sum_{i=1}^p 1\left\{\|\hat{V}^1_{-i,i}\|_2 > t_s \text{ and } \|\hat{V}^2_{-i,i}\|_2 > t_s\right\};$ $\text{GGL/GL:} \sum_{i=1}^p 1\left\{\|\hat{\Theta}^1_{-i,i}\|_2 > t_s \text{ and } \|\hat{\Theta}^2_{-i,i}\|_2 > t_s\right\}$ <u>True positive co-hub columns</u> (TPCC): $\text{CNJGL:} \sum_{i \in I_c} 1\left\{\|\hat{V}^1_{-i,i}\|_2 > t_s \text{ and } \|\hat{V}^2_{-i,i}\|_2 > t_s\right\};$ $\text{CGL/GL:} \sum_{i \in I_c} 1\left\{\|\hat{\Theta}^1_{-i,i}\|_2 > t_s \text{ and } \|\hat{\Theta}^2_{-i,i}\|_2 > t_s\right\}$ where I_C is the set of co-hub column indices.
(3)	<u>Error:</u> $\sqrt{\sum_{i<j}(\Theta^1_{ij} - \hat{\Theta}^1_{ij})^2} + \sqrt{\sum_{i<j}(\Theta^2_{ij} - \hat{\Theta}^2_{ij})^2}$