# On Sparse Graph Estimation Under Statistical and Laplacian Constraints

Jitendra K Tugnait
Auburn University, Auburn, AL, USA
E-mail: tugnajk@auburn.edu

***Abstract*—We consider the problem of estimating the structure of an undirected weighted sparse graph underlying a set of signals, exploiting both smoothness of the signals as well as their statistics. We augment the objective function of Kalofolias (2016) which is motivated by a signal smoothness viewpoint and imposes a Laplacian constraint, with a penalized log-likelihood objective function with a lasso constraint, motivated from a statistical viewpoint. Both of these objective functions are designed for estimation of sparse graphs. An alternating direction method of multipliers (ADMM) algorithm is presented to optimize the augmented objective function. Numerical results based on synthetic data show that the proposed approach improves upon Kalofolias (2016) in estimating the inverse covariance, and improves upon graphical lasso in estimating the graph topology. We also implement an adaptive version of the proposed algorithm following adaptive lasso of Zou (2006), and empirically show that it leads to further improvement in performance.**

## I. Introduction

An undirected simple weighted graph is denoted $\mathcal{G} = (V, \mathcal{E}, \boldsymbol{W})$ where $V = \{1, 2, \cdots, p\} = [p]$ is the set of $p$ nodes, $\mathcal{E} \subseteq [p] \times [p]$ is the set of undirected edges, and $\boldsymbol{W} \in \mathbb{R}^{p\times p}$ stores the non-negative weights $W_{ij} \geq 0$ associated with the undirected edges. If there is an edge between nodes $i$ and $j$, then edge $\{i, j\} \in \mathcal{E}$ and $W_{ij} > 0$, otherwise $\{i, j\} \notin \mathcal{E}$ and $W_{ij} = 0$. In a simple graph $W_{ii} = 0$. In an undirected graph, if $\{i, j\} \in \mathcal{E}$, then $\{j, i\} \in \mathcal{E}$. In graphical models of data variables $x_1, x_1, \cdots, x_p$, ($\boldsymbol{x} = [x_1\ x_2\ \cdots\ x_p]^\top$), a weighted graph $\mathcal{G} = (V, \mathcal{E}, \boldsymbol{W})$ (or unweighted $\mathcal{G} = (V, \mathcal{E})$) with $|V| = p$ is used to capture relationships between the $p$ variables $x_i$s [1]–[3]. If $\{i, j\} \in \mathcal{E}$, then $x_i$ and $x_j$ are related (similar or dependent) in some sense, with higher $W_{ij}$ indicating stronger similarity or dependence.

Graphical models provide a powerful tool for analyzing multivariate data [1]–[3]. In a statistical graphical model, the conditional statistical dependency structure among $p$ random variables $x_1, x_1, \cdots, x_p$, is represented using an undirected graph $\mathcal{G} = (V, \mathcal{E})$. There is no edge between nodes $i$ and $j$ iff $x_i$ and $x_j$ are conditionally independent given the remaining $p$-2 variables. Suppose $\boldsymbol{x}$ has positive-definite covariance matrix $\boldsymbol{\Sigma}$ with inverse covariance matrix $\boldsymbol{\Omega} = \boldsymbol{\Sigma}^{-1}$. Then $\Omega_{ij}$, the $(i, j)$-th element of $\boldsymbol{\Omega}$, is zero iff $x_i$ and $x_j$ are conditionally independent. Such models for $\boldsymbol{x}$ have been extensively studied. In high-dimensional settings, one estimates $\boldsymbol{\Omega}$ under some sparsity constraints; see [4]–[8].

This work was supported by NSF Grants CCF-1617610 and ECCS-2040536.

Graphical models for data variables have been inferred from consideration other than statistical, depending upon the intended application, nature of data and available prior information [1]. One class of graphical models are based on signal smoothness [1], [9]–[11]. Given $n$ samples $\{\boldsymbol{x}(t)\}_{t=1}^n$ of the $p$ data variables $x_1, x_1, \cdots, x_p$, $\boldsymbol{x}(t) = [x_1(t)\ x_2(t)\ \cdots\ x_p(t)]^\top$. Define

$$\boldsymbol{X} = \begin{bmatrix}\mathbf{x}(1) & \mathbf{x}(2) & \cdots & \mathbf{x}(n)\end{bmatrix} \in \mathbb{R}^{p\times n}. \tag{1}$$

A measure of smoothness of signal $\boldsymbol{x}(t)$ under which the signal takes "similar" values at "neighboring" vertices of a given weighted undirected graph, is the function [1], [9]

$$\frac{1}{2}\sum_{i,j=1}^{p} W_{ij}\|\boldsymbol{X}_{i\cdot} - \boldsymbol{X}_{j\cdot}\|_2^2 = \mathrm{tr}(\boldsymbol{X}^\top \boldsymbol{L}\boldsymbol{X}) \tag{2}$$

where $\boldsymbol{X}_{i\cdot}$ denotes the $i$th row of $\boldsymbol{X}$, $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{W}$ is the (combinatorial) graph Laplacian (matrix), and $\boldsymbol{D}$ is the diagonal weighted degree matrix with $D_{ii} = \sum_{j=1}^p W_{ij}$. Based on the smoothness constraint, graph learning from data $\boldsymbol{X}$ becomes equivalent to estimation of Laplacian $\boldsymbol{L}$ [1], [9].

Another set of approaches are based on statistical considerations under the graph Laplacian constraint [1], [12]–[14] where Laplacian $\boldsymbol{L}$, after regularization, plays the role of inverse covariance $\boldsymbol{\Omega}$; $\boldsymbol{L}$ is a symmetric, non-negative-definite matrix but with non-positive off-diagonal entries. These approaches apply only when off-diagonal entries of inverse covariance are non-positive.

Graph Laplacian matrix has been extensively used for embedding, manifold learning, clustering and semi-supervised learning [15], [16]; see [1], [9] for further applications.

### A. Related Work and Our Contributions

Prior work related to the objective of our paper is discussed in detail in Sec. II. In this paper, we augment the smoothness-based objective function of Kalofolias [9] with a penalized log-likelihood objective function with a lasso constraint (as in [7]). An alternating direction method of multipliers (ADMM) algorithm is presented to optimize the augmented objective function to infer the graph topology and to estimate the inverse covariance (or, equivalently, the graph edge weights). Numerical results based on synthetic data show that the proposed approach improves upon [9] in estimating the inverse covariance or the Laplacian, and improves upon [7] in estimating the graph topology. It also outperforms [13] which

uses a Laplacian constraint. We also implement an adaptive version of the proposed algorithm following adaptive lasso of [17], and empirically show that it leads to further improvement in performance.

**Notation**: Given $\boldsymbol{A} \in \mathbb{R}^{p\times p}$, we use $|\boldsymbol{A}|$ and $\text{tr}(\boldsymbol{A})$ to denote the determinant and trace of $\boldsymbol{A}$, respectively. Given a set $V$, $|V|$ denotes its cardinality (number of elements). For a matrix $\boldsymbol{B} \in \mathbb{R}^{p\times q}$, we define the the Frobenius norm and the vectorized $\ell_1$ norm, respectively, as $\|\boldsymbol{B}\|_F = \sqrt{\text{tr}(\boldsymbol{B}^\top \boldsymbol{B})}$ and $\|\boldsymbol{B}\|_1 = \sum_{i,j} |B_{ij}|$ where $B_{ij}$ is the $(i,j)$-th element of $\boldsymbol{B}$. We also denote $B_{ij}$ by $[\boldsymbol{B}]_{ij}$. Given $\boldsymbol{A} \in \mathbb{R}^{p\times p}$, $\boldsymbol{A}^+ = \text{diag}(\boldsymbol{A})$ is a diagonal matrix with the same diagonal as $\boldsymbol{A}$, and $\boldsymbol{A}^- = \boldsymbol{A} - \boldsymbol{A}^+$ is $\boldsymbol{A}$ with all its diagonal elements set to zero. Symbol $\boldsymbol{I}_p$ is the $p\times p$ identity matrix, $\boldsymbol{1}_p$ is a column of $p$ ones, $\boldsymbol{B}_{i\cdot}$ and $\boldsymbol{B}_{\cdot j}$ denote column vectors comprising $i$th row and $j$th column, respectively, of $\boldsymbol{B}$, and $\boldsymbol{1}_A$ denotes indicator function (1 if $A$ is true, else 0). For a symmetric matrix $\boldsymbol{A}$, $\boldsymbol{A} \succ \boldsymbol{0}$ and $\boldsymbol{A} \succeq \boldsymbol{0}$ denote that $\boldsymbol{A}$ is positive-definite and positive semi-definite, respectively.

## II. Some Existing Approaches

Here we briefly review some significant existing methods for graph estimation.

### A. Smoothness-Based Graph Learning

Consider the approach of [9]. With reference to (1) and (2), it is established in [9] that $\text{tr}(\boldsymbol{X}^\top \boldsymbol{L}\boldsymbol{X}) = \frac{1}{2}\text{tr}(\boldsymbol{W}\hat{\boldsymbol{Z}})$ where $\boldsymbol{W}, \hat{\boldsymbol{Z}} \in \mathbb{R}^{p\times p}$, $\hat{Z}_{ij} = \|\boldsymbol{X}_{i\cdot} - \boldsymbol{X}_{j\cdot}\|_2^2$ and $\boldsymbol{W}$ is the weight matrix (or the weighted adjacency matrix) with $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{W}$, $\boldsymbol{W} = \boldsymbol{W}^\top$, $W_{ij} \geq 0$ and $W_{ii} = 0$ for $1 \leq i,j \leq p$. Instead of performing a penalized minimization of $\text{tr}(\boldsymbol{X}^\top \boldsymbol{L}\boldsymbol{X})$ to estimate $\boldsymbol{L}$, [9] minimizes a penalized $\text{tr}(\boldsymbol{W}\hat{\boldsymbol{Z}})$ w.r.t. $\boldsymbol{W}$ for graph learning. Given $\boldsymbol{W}$, one has unique $\boldsymbol{L}$ and the edge-set $\mathcal{E}$. *In the rest of the paper, we will scale $\hat{\boldsymbol{Z}}$ as $\hat{\boldsymbol{Z}}/n$ and still denote the latter as $\hat{\boldsymbol{Z}}$.*

Define the space $\mathcal{W}_p$ of all valid $p\times p$ weight matrices $\boldsymbol{W}$

$$\mathcal{W}_p = \Big\{\boldsymbol{W} \in \mathbb{R}^{p\times p} : \boldsymbol{W} = \boldsymbol{W}^\top,\ W_{ij} \geq 0,\ W_{ii} = 0\Big\} \quad (3)$$

In [9] one looks for $\min_{\boldsymbol{W}\in\mathcal{W}_p} f_s(\boldsymbol{W})$ where

$$f_s(\boldsymbol{W}) = \text{tr}(\boldsymbol{W}\hat{\boldsymbol{Z}}) + \frac{\beta}{2}\|\boldsymbol{W}\|_F^2 - \alpha\sum_{i=1}^{p} \ln\Big(\sum_{j=1}^{p} W_{ij}\Big) \quad (4)$$

with parameters $\alpha > 0$ and $\beta \geq 0$ controlling the "shape.". In (4), $\text{tr}(\boldsymbol{W}\hat{\boldsymbol{Z}})$ is the main cost but minimizing it alone w.r.t. $\boldsymbol{W}$ is ill-posed ($\boldsymbol{W} = \boldsymbol{0}$ minimizes it). Using only the logarithmic barrier ($\beta = 0$) leads to very sparse graphs, and changing $\alpha$ only changes the scale of the solution. The term $\frac{\beta}{2}\|\boldsymbol{W}\|_F^2$ controls graph sparsity, with smaller $\beta$ leading to sparser graph.

*1) A Forward-Backward Primal-Dual Algorithm:* This is the algorithm used in [9]. A forward-backward algorithm based on [18] is given in [9] to optimize (4), where optimization is carried for fixed $\alpha = 1$ and then one scales $\boldsymbol{W}$ to obtain a desired $\|\boldsymbol{W}\|$. Software implementation of this algorithm is available in [19].

*2) ADMM Solution:* This is an alternative solution given in [20]. An ADMM algorithm (also a primal-dual algorithm) is given in [20] to optimize (4), although in a different context and using scaled $\hat{\boldsymbol{Z}}$.

### B. Graphical Lasso: Penalized Log-Likelihood

This is the approach first proposed in [7]. With $\hat{\boldsymbol{\Sigma}}$ denoting the sample covariance (assume zero-mean: $\hat{\boldsymbol{\Sigma}} = \frac{1}{n}\sum_{t=1}^{n} \boldsymbol{x}(t)\boldsymbol{x}^\top(t)$), seek $\boldsymbol{\Omega}$ to yield $\min_{\boldsymbol{\Omega}\succ\boldsymbol{0}} f_L(\boldsymbol{\Omega})$ where

$$f_L(\boldsymbol{\Omega}) = \text{tr}(\boldsymbol{\Omega}\hat{\boldsymbol{\Sigma}}) - \ln(|\boldsymbol{\Omega}|) + \lambda\|\boldsymbol{\Omega}^-\|_1\,, \quad (5)$$

$\lambda\|\boldsymbol{\Omega}^-\|_1$ is the lasso penalty and $\lambda > 0$. Unlike Laplacian $\boldsymbol{L}$, off-diagonal entries of $\boldsymbol{\Omega}$ may not be non-positive. This is a statistical approach. In addition to the coordinate descent approach of [7], there are numerous algorithms to optimize (5) such as [8], [21]–[23].

### C. Generalized Graph Laplacian Estimation

Here we summarize the approach of [13]. In [1], [12]–[14] approaches that make $\boldsymbol{\Omega} = \boldsymbol{L}$ (Laplacian, or some regularized version) in (5) have been considered. In particular, [13] considers fitting a generalized graph Laplacian (GGL) matrix to data where a GGL matrix $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{W} + \boldsymbol{V}$ such that $\boldsymbol{W}$ and $\boldsymbol{D}$ are the weighted adjacency matrix and the diagonal degree matrix of the graph (as before), respectively, and $\boldsymbol{V}$ is a diagonal matrix with positive diagonal elements. With the addition of $\boldsymbol{V}$, GGL matrix $\boldsymbol{L}$ becomes positive definite. More specifically, [13] considers

$$\min_{\boldsymbol{\Theta}\succ\boldsymbol{0}} \text{tr}(\boldsymbol{\Theta}\hat{\boldsymbol{K}}) - \ln(|\boldsymbol{\Theta}|),\ \ \hat{\boldsymbol{K}} = \hat{\boldsymbol{\Sigma}} + \lambda(\boldsymbol{I} - \boldsymbol{1}_p\boldsymbol{1}_p^\top) \quad (6)$$

with $\boldsymbol{\Theta}$ restricted to be a generalized graph Laplacian matrix and $\lambda$ is as in (5). A software implementation of this algorithm is available in [24]. This is a statistical approach with Laplacian constraint.

### D. Adaptive Lasso

This modification of graphical lasso follows from [17], which, however, is focused on regression problems. With $\hat{\boldsymbol{\Omega}} = \arg\min_{\boldsymbol{\Omega}\succ\boldsymbol{0}} f_L(\boldsymbol{\Omega})$ from Sec. II-B, modify (5) as

$$\min_{\boldsymbol{\Omega}\succ\boldsymbol{0}} \text{tr}(\boldsymbol{\Omega}\hat{\boldsymbol{\Sigma}}) - \ln(|\boldsymbol{\Omega}|) + \lambda \sum_{i,j=1,\, i\neq j}^{p} \Omega_{ij}/|\hat{\Omega}_{ij}|\,, \quad (7)$$

i.e., use penalty varying with $(i,j)$ as $\lambda/|\hat{\Omega}_{ij}|$; for $|\hat{\Omega}_{ij}| = 0$, we use $\lambda/\epsilon$ with $0 < \epsilon \ll 1$. This approach (approximately) debiases estimate of $\boldsymbol{\Omega}$.

## III. Proposed Approach

We propose to augment the smoothness-based objective function of Kalofolias [9] with a penalized log-likelihood objective function with a lasso constraint (as in [7]). As has been observed by several researchers (see, e.g., [13]), $\boldsymbol{W}$ estimated via [9] yields a (severely) biased estimate of the Laplacian (whether combinatorial or generalized) $\boldsymbol{L}$. But, as the results of [9], [10] show, their approach performs quite well in estimating the edges of the graph, with the performance

measured in terms of the $F_1$-score. Recall that the $F_1$-score is defined as $F_1 = (2 \times \text{precision} \times \text{recall})/(\text{precision} + \text{recall})$ where precision $= |\hat{\mathcal{E}} \cap \mathcal{E}_0|/|\hat{\mathcal{E}}|$, recall $= |\hat{\mathcal{E}} \cap \mathcal{E}_0|/|\mathcal{E}_0|$, and $\mathcal{E}_0$ and $\hat{\mathcal{E}}$ denote the true and estimated edge sets, respectively. On the other hand, while the penalized log-likelihood objective function with a lasso constraint performs well in general, both in terms of the $F_1$-score and accuracy of estimation of the precision matrix $\boldsymbol{\Omega}$, it does not guarantee that the off-diagonal elements of $\boldsymbol{\Omega}$ will be non-positive. Since our objective is to estimate a Laplacian matrix, we propose to combine the smoothness-based objective function of Kalofolias [9] with a penalized log-likelihood objective function with a lasso constraint. Our goal is to maintain the edge detection efficacy of [9] (high $F_1$-score) while improving the estimation accuracy of $\boldsymbol{W}$ by exploiting its relationship to $\boldsymbol{\Omega}$.

### A. Objective Function

Combine (4) and (5) to define the cost

$$L_{aug}(\boldsymbol{\Omega}, \boldsymbol{W}) = \gamma f_L(\boldsymbol{\Omega}) + (1-\gamma) f_s(\boldsymbol{W}) + \lambda_g \sum_{i \neq j}^{p} \sqrt{W_{ij}^2 + \Omega_{ij}^2} \tag{8}$$

where $\gamma \in (0,1)$ yields a convex combination of (4) and (5). Cost $L_{aug}(\boldsymbol{\Omega}, \boldsymbol{W})$ is strictly convex in $\boldsymbol{\Omega}$ and $\boldsymbol{W}$ for $\boldsymbol{W} \in \mathcal{W}_p$ and $\boldsymbol{\Omega} \succ \mathbf{0}$. We propose to minimize $L_{aug}(\boldsymbol{\Omega}, \boldsymbol{W})$ w.r.t. $\boldsymbol{W} \in \mathcal{W}_p$ and $\boldsymbol{\Omega} \succ \mathbf{0}$. The group-lasso penalty [4] $\lambda_g \sum_{i \neq j}^{p} \sqrt{W_{ij}^2 + \Omega_{ij}^2}$ in $L_{aug}(\boldsymbol{\Omega}, \boldsymbol{W})$ makes both $\Omega_{ij}$ and $W_{ij}$ sparse for the same edge $\{i, j\}$ of the graph, whereas $\lambda \|\boldsymbol{\Omega}^-\|_1$ and $\frac{\beta}{2}\|\boldsymbol{W}\|_F^2$ control sparsity in $\Omega_{ij}$ and $W_{ij}$ individually.

### B. ADMM Solution Outline

We will use ADMM [22] after variable splitting to minimize $L_{aug}(\boldsymbol{\Omega}, \boldsymbol{W})$. We note that $L_{aug}(\boldsymbol{\Omega}, \boldsymbol{W})$ is strictly convex, and its domain is Cartesian product of the set $\mathcal{W}_p$ defined in (3) and the set of strictly positive definite matrices $\boldsymbol{\Omega}$ (the latter because of $-\ln(|\boldsymbol{\Omega}|)$, i.e., use of the log-determinant barrier function [25]). The objective function $L_{aug}(\boldsymbol{\Omega}, \boldsymbol{W})$ is also closed, proper and lower semi-continuous.

Using variable splitting, we split $f_L(\boldsymbol{\Omega})$ into $\tilde{f}_L(\boldsymbol{\Omega}, \boldsymbol{V}_1)$:

$$\tilde{f}_L(\boldsymbol{\Omega}, \boldsymbol{V}_1) = \text{tr}(\boldsymbol{\Omega}\hat{\boldsymbol{\Sigma}}) - \ln(|\boldsymbol{\Omega}|) + \lambda \|\boldsymbol{V}_1^-\|_1 \tag{9}$$

$$\text{subject to} \quad \boldsymbol{\Omega} = \boldsymbol{V}_1 \in \mathbb{R}^{p \times p}. \tag{10}$$

We split $f_s(\boldsymbol{W})$ into $\tilde{f}_s(\boldsymbol{W}, \boldsymbol{V}_2, \boldsymbol{d})$ with $\boldsymbol{d} = [d_1, \cdots, d_p]^\top$:

$$\tilde{f}_s(\boldsymbol{W}, \boldsymbol{V}_2, \boldsymbol{d}) = \text{tr}(\boldsymbol{W}\hat{\boldsymbol{Z}}) + \frac{\beta}{2}\|\boldsymbol{W}\|_F^2 - \alpha \sum_{i=1}^{p} \ln(d_i) \tag{11}$$

$$\text{subject to} \quad \boldsymbol{W} = \boldsymbol{V}_2 \in \mathbb{R}^{p \times p}, \ \boldsymbol{W}\mathbf{1}_p = \boldsymbol{d}, \tag{12}$$

where $\sum_{j=1}^{p} W_{ij} = d_i$ is the $i$th element of $\boldsymbol{W}\mathbf{1}_p$. Using $\tilde{f}_L(\boldsymbol{\Omega}, \boldsymbol{V}_1)$ and $\tilde{f}_s(\boldsymbol{W}, \boldsymbol{V}_2, \boldsymbol{d})$ for $f_L(\boldsymbol{\Omega})$ and $f_s(\boldsymbol{W})$, respectively, in $L_{aug}(\boldsymbol{\Omega}, \boldsymbol{W})$, we have the cost with split variables

$$\tilde{L}_{aug}(\boldsymbol{\Omega}, \boldsymbol{W}, \boldsymbol{V}_1, \boldsymbol{V}_2, \boldsymbol{d}) = \gamma \tilde{f}_L(\boldsymbol{\Omega}, \boldsymbol{V}_1) + (1-\gamma)\tilde{f}_s(\boldsymbol{W}, \boldsymbol{V}_2, \boldsymbol{d}) + \lambda_g \sum_{i \neq j}^{p} \sqrt{V_{1ij}^2 + V_{2ij}^2} \tag{13}$$

$$\text{subject to} \quad \boldsymbol{\Omega} = \boldsymbol{V}_1, \ \boldsymbol{W} = \boldsymbol{V}_2, \ \boldsymbol{W}\mathbf{1}_p = \boldsymbol{d}. \tag{14}$$

We minimize $\tilde{L}_{aug}(\boldsymbol{\Omega}, \boldsymbol{W}, \boldsymbol{V}_1, \boldsymbol{V}_2, \boldsymbol{d})$ with respect to $\boldsymbol{\Omega}, \boldsymbol{W}, \boldsymbol{V}_1, \boldsymbol{V}_2, \boldsymbol{d}$ subject to the constraints in (14), following the ADMM approach. The scaled augmented Lagrangian [22] for this problem is

$$\begin{aligned} L_\rho =& \gamma(\text{tr}(\boldsymbol{\Omega}\hat{\boldsymbol{\Sigma}}) - \ln(|\boldsymbol{\Omega}|)) + (1-\gamma)\Big(\text{tr}(\boldsymbol{W}\hat{\boldsymbol{Z}}) - \alpha \sum_{i=1}^{p} \ln(d_i) + \frac{\beta}{2} \sum_{i \neq j}^{p} W_{ij}^2\Big) + \lambda \|\boldsymbol{V}_1^-\|_1 \\ &+ \lambda_g \sum_{i \neq j}^{p} \sqrt{V_{2ij}^2 + V_{1ij}^2} \\ &+ \frac{\rho}{2}\Big(\|\boldsymbol{V}_1 - \boldsymbol{\Omega} + \boldsymbol{U}_1\|_F^2 + \|\boldsymbol{V}_2 - \boldsymbol{W} + \boldsymbol{U}_2\|_F^2 + \|\boldsymbol{d} - \boldsymbol{W}\mathbf{1}_p + \boldsymbol{u}_3\|_2^2\Big) \end{aligned} \tag{15}$$

where we have absorbed $\gamma$ in $\lambda$ in $\lambda\|\boldsymbol{V}_1^-\|_1$, $\boldsymbol{U}_1, \boldsymbol{U}_2 \in \mathbb{R}^{p \times p}$ and $\boldsymbol{u}_3 \in \mathbb{R}^p$ are the dual variables, and $\rho > 0$ is a penalty parameter. Note also that since $W_{ii} = 0$ for every $i$, we have written $\frac{\beta}{2}\|\boldsymbol{W}\|_F^2$ as $\frac{\beta}{2}\sum_{i \neq j}^{p} W_{ij}^2$ in (15). The Lagrangian $L_\rho$ is optimized iteratively. Given the results $\boldsymbol{\Omega}^{(k)}, \boldsymbol{W}^{(k)}, \boldsymbol{V}_1^{(k)}, \boldsymbol{V}_2^{(k)}, \boldsymbol{d}^{(k)}, \boldsymbol{U}_1^{(k)}, \boldsymbol{U}_2^{(k)}$ and $\boldsymbol{u}_3^{(k)}$, of the $k$th iteration, in the $(k+1)$st iteration, an ADMM algorithm executes the following three updates [22]:

(a) Minimize $L_\rho$ w.r.t. $\boldsymbol{\Omega}$ and $\boldsymbol{W}$ resulting in separable optimizations in $\boldsymbol{\Omega}$ and $\boldsymbol{W}$. Define

$$L_{a1}(\boldsymbol{\Omega}) = \gamma\Big(\text{tr}(\boldsymbol{\Omega}\hat{\boldsymbol{\Sigma}}) - \ln(|\Omega|)\Big) + \frac{\rho}{2}\|\boldsymbol{V}_1^{(k)} - \boldsymbol{\Omega} + \boldsymbol{U}_1^{(k)}\|_F^2 \tag{16}$$

$$L_{a2}(\boldsymbol{W}) = (1-\gamma)\Big(\text{tr}(\boldsymbol{W}\hat{\boldsymbol{Z}}) + \frac{\beta}{2}\sum_{i \neq j}^{p} W_{ij}^2\Big) + \frac{\rho}{2}\|\boldsymbol{V}_2^{(k)} - \boldsymbol{W} + \boldsymbol{U}_2^{(k)}\|_F^2 + \frac{\rho}{2}\|\boldsymbol{d}^{(k)} - \boldsymbol{W}\mathbf{1}_p + \boldsymbol{u}_3^{(k)}\|_2^2 \tag{17}$$

Minimization of $L_\rho$ w.r.t. $\boldsymbol{\Omega}$ and $\boldsymbol{W}$ is then equivalent to minimization of $L_{a1}(\boldsymbol{\Omega})$ w.r.t. $\boldsymbol{\Omega}$ and minimization of $L_{a2}(\boldsymbol{W})$ w.r.t. $\boldsymbol{W}$ separately.

(a-i) Let $\boldsymbol{\Omega}^{(k+1)} \leftarrow \arg\min_{\boldsymbol{\Omega} \succ \mathbf{0}} L_{a1}(\boldsymbol{\Omega})$ .

(a-ii) Let $\boldsymbol{W}^{(k+1)} \leftarrow \arg\min_{\boldsymbol{W} \in \mathcal{W}_p} L_{a2}(\boldsymbol{W})$ .

(b) Minimize $L_\rho$ w.r.t. $\boldsymbol{V}_1$, $\boldsymbol{V}_2$ and $\boldsymbol{d}$ resulting in separable

optimizations in $(\boldsymbol{V}_1, \boldsymbol{V}_2)$ and $\boldsymbol{d}$. Define

$$\begin{aligned} L_{b1}(\boldsymbol{V}_1, \boldsymbol{V}_2) =& \lambda \|\boldsymbol{V}_1^-\|_1 + \lambda_g \sum_{i \neq j}^{p} \sqrt{V_{2ij}^2 + V_{1ij}^2} \\ &+ \frac{\rho}{2} \|\boldsymbol{V}_1 - \boldsymbol{\Omega}^{(k+1)} + \boldsymbol{U}_1^{(k)}\|_F^2 \\ &+ \frac{\rho}{2} \|\boldsymbol{V}_2 - \boldsymbol{W}^{(k+1)} + \boldsymbol{U}_2^{(k)}\|_F^2) \end{aligned} \quad (18)$$

$$\begin{aligned} L_{b2}(\boldsymbol{d}) =& -\alpha \sum_{i=1}^{p} \ln(d_i) \\ &+ \frac{\rho}{2} \|\boldsymbol{d} - \boldsymbol{W}^{(k+1)} \mathbf{1}_p + \boldsymbol{u}_3^{(k)}\|_2^2 \end{aligned} \quad (19)$$

Minimization of $L_\rho$ w.r.t. $\boldsymbol{V}_1$, $\boldsymbol{V}_2$ and $\boldsymbol{d}$ is then equivalent to minimization of $L_{b1}(\boldsymbol{V}_1, \boldsymbol{V}_2)$ w.r.t. $\boldsymbol{V}_1$, $\boldsymbol{V}_2$ and minimization of $L_{b2}(\boldsymbol{d})$ w.r.t. $\boldsymbol{d}$ separately.

(b-i) Let $(\boldsymbol{V}_1^{(k+1)}, \boldsymbol{V}_2^{(k+1)}) \leftarrow \arg\min_{\boldsymbol{V}_1, \boldsymbol{V}_2} L_{b1}(\boldsymbol{V}_1, \boldsymbol{V}_2)$.

(b-ii) Let $\boldsymbol{d}^{(k+1)} \leftarrow \arg\min_{\boldsymbol{d}} L_{b2}(\boldsymbol{d})$ .

(c) Dual updates:

$$\boldsymbol{U}_1^{(k+1)} \leftarrow \boldsymbol{U}_1^{(k)} + \boldsymbol{V}_1^{(k+1)} - \boldsymbol{\Omega}^{(k+1)} \quad (20)$$

$$\boldsymbol{U}_2^{(k+1)} \leftarrow \boldsymbol{U}_2^{(k)} + \boldsymbol{V}_2^{(k+1)} - \boldsymbol{W}^{(k+1)}, \quad (21)$$

$$\boldsymbol{u}_3^{(k+1)} \leftarrow \boldsymbol{u}_3^{(k)} + \boldsymbol{d}^{(k+1)} - \boldsymbol{W}^{(k+1)} \mathbf{1}_p . \quad (22)$$

### C. Detailed ADMM Solution

We now discuss solutions to updates (a) and (b). Update (c) is a standard part of dual update in ADMM when using a scaled augmented Lagrangian formulation [22].

*Update (a-i)*. For update (a-i) we need

$$\mathbf{0} = \frac{\partial L_{a1}(\boldsymbol{\Omega})}{\partial \boldsymbol{\Omega}} = \gamma(\hat{\boldsymbol{\Sigma}} - \boldsymbol{\Omega}^{-1}) - \rho(\boldsymbol{V}_1^{(k)} - \boldsymbol{\Omega} + \boldsymbol{U}_1^{(k)}) . \quad (23)$$

A similar problem has been solved in [22, Sec. 6.5]. Consider eigen-decomposition of symmetric matrix $\hat{\boldsymbol{\Sigma}} - (\rho/\gamma)(\boldsymbol{V}_1^{(k)} + \boldsymbol{U}_1^{(k)})$ given by

$$\hat{\boldsymbol{\Sigma}} - (\rho/\gamma)(\boldsymbol{V}_1^{(k)} + \boldsymbol{U}_1^{(k)}) = \boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^\top \quad (24)$$

where $\boldsymbol{\Lambda}$ is a diagonal matrix with eigenvalues on the diagonal and $\boldsymbol{Q}$ has corresponding orthogonal eigenvectors as its columns, resulting in $\boldsymbol{Q}\boldsymbol{Q}^\top = \boldsymbol{Q}^\top\boldsymbol{Q} = \boldsymbol{I}_p$. Then the solution to (23) is given by

$$\boldsymbol{\Omega}^{(k+1)} = \boldsymbol{Q}\tilde{\boldsymbol{\Lambda}}\boldsymbol{Q}^\top \quad (25)$$

where $\tilde{\boldsymbol{\Lambda}}$ is the diagonal matrix with $\ell$th diagonal element

$$\tilde{\Lambda}_{\ell\ell} = \frac{-\Lambda_{\ell\ell} + \sqrt{\Lambda_{\ell\ell}^2 + 4(\rho/\gamma)}}{2(\rho/\gamma)} . \quad (26)$$

Since $\rho > 0$, $\tilde{\Lambda}_{\ell\ell} > 0$ for every $\ell = 1, 2, \cdots, p$, and therefore, $\boldsymbol{\Omega}^{(k+1)} \succ \mathbf{0}$. It is shown in [22, Sec. 6.5] that $\boldsymbol{\Omega}^{(k+1)}$ given by (25) satisfies (23).

*Update (a-ii)*. First some notation. Recall that $\boldsymbol{W}_{i\cdot} \in \mathbb{R}^p$ denotes the $i$th row of $\boldsymbol{W} \in \mathbb{R}^{p\times}$. Let $\check{\boldsymbol{W}}_{i\cdot} \in \mathbb{R}^{p-1}$ denote the vector obtained from $\boldsymbol{W}_{i\cdot}$ by deleting its $i$th row. Recall that $W_{ii} = 0$ for $i = 1, 2, \cdots, p$, therefore, we do not have to solve for them. Similarly, let $\check{\hat{\boldsymbol{Z}}}_{i\cdot}$, $\check{\boldsymbol{V}}_{2i\cdot}$, $\check{\boldsymbol{U}}_{2i\cdot}$ denote vectors in $\mathbb{R}^{p-1}$, obtained from $\hat{\boldsymbol{Z}}_{i\cdot}$, $\boldsymbol{V}_{2i\cdot}$, $\boldsymbol{U}_{2i\cdot}$, respectively, by deleting their $i$th row. Using this notation and noting that $\hat{\boldsymbol{Z}}$ and $\boldsymbol{W}$ are symmetric, we can rewrite

$$\mathrm{tr}(\boldsymbol{W}\hat{\boldsymbol{Z}}) = \sum_{i=1}^{p} \check{\boldsymbol{W}}_{i\cdot}^\top \check{\hat{\boldsymbol{Z}}}_{i\cdot} , \quad \sum_{i\neq j}^{p} W_{ij}^2 = \sum_{i=1}^{p} \check{\boldsymbol{W}}_{i\cdot}^\top \check{\boldsymbol{W}}_{i\cdot} . \quad (27)$$

Similarly, we have

$$\begin{aligned} \|\boldsymbol{V}_2^{(k)} - \boldsymbol{W} + \boldsymbol{U}_2^{(k)}\|_F^2 =& \sum_{i=1}^{p} \check{\boldsymbol{W}}_{i\cdot}^\top \check{\boldsymbol{W}}_{i\cdot} \\ &- 2\sum_{i=1}^{p} (\check{\boldsymbol{V}}_{2i\cdot}^{(k)} + \check{\boldsymbol{U}}_{2i\cdot}^{(k)})^\top \check{\boldsymbol{W}}_{i\cdot} \\ &+ \sum_{i=1}^{p} (\check{\boldsymbol{V}}_{2i\cdot}^{(k)} + \check{\boldsymbol{U}}_{2i\cdot}^{(k)})^\top (\check{\boldsymbol{V}}_{2i\cdot}^{(k)} + \check{\boldsymbol{U}}_{2i\cdot}^{(k)}), \end{aligned} \quad (28)$$

$$\begin{aligned} \|\boldsymbol{d}^{(k)} - \boldsymbol{W}\mathbf{1}_p + \boldsymbol{u}_3^{(k)}\|_2^2 =& \sum_{i=1}^{p} \check{\boldsymbol{W}}_{i\cdot}^\top \mathbf{1}_{p-1}\mathbf{1}_{p-1}^\top \check{\boldsymbol{W}}_{i\cdot} \\ &- 2\sum_{i=1}^{p} (d_i^{(k)} + u_{3i}^{(k)}) \mathbf{1}_{p-1}^\top \check{\boldsymbol{W}}_{i\cdot} + \sum_{i=1}^{p} (d_i^{(k)} + u_{3i}^{(k)})^2 \end{aligned} \quad (29)$$

where we have used the fact that since $\boldsymbol{\Omega}$ and $\boldsymbol{W}$ are symmetric, so are $\boldsymbol{V}_1$, $\boldsymbol{V}_2$, $\boldsymbol{U}_1$ and $\boldsymbol{U}_2$. Using (27)-(29), we can express $L_{a2}(\boldsymbol{W})$ as

$$\frac{1}{1-\gamma} L_{a2}(\boldsymbol{W}) = \sum_{i=1}^{p} L_{a2i}(\check{\boldsymbol{W}}_{i\cdot}) \quad (30)$$

where ($\boldsymbol{I}_{p-1}$ denotes $(p-1)\times(p-1)$ identity matrix)

$$L_{a2i}(\check{\boldsymbol{W}}_{i\cdot}) = \frac{1}{2}\check{\boldsymbol{W}}_{i\cdot}^\top \boldsymbol{A} \check{\boldsymbol{W}}_{i\cdot} + (\boldsymbol{b}^{(i)})^\top \check{\boldsymbol{W}}_{i\cdot} + c^{(i)} , \quad (31)$$

$$\boldsymbol{A} = (\beta + \bar{\rho})\boldsymbol{I}_{p-1} + \bar{\rho}\mathbf{1}_{p-1}\mathbf{1}_{p-1}^\top , \quad (32)$$

$$\bar{\rho} = \frac{\rho}{1-\gamma} , \quad (33)$$

$$\boldsymbol{b}^{(i)} = \check{\hat{\boldsymbol{Z}}}_{i\cdot} - \bar{\rho}(\check{\boldsymbol{V}}_{2i\cdot}^{(k)} + \check{\boldsymbol{U}}_{2i\cdot}^{(k)} + (d_i^{(k)} + U_{3i}^{(k)})\mathbf{1}_{p-1}) , \quad (34)$$

$$c^{(i)} = \frac{\bar{\rho}}{2}\Big((d_i^{(k)} + u_{3i}^{(k)})^2 + \|\check{\boldsymbol{V}}_{2i\cdot}^{(k)} + \check{\boldsymbol{U}}_{2i\cdot}^{(k)}\|_2^2\Big) \quad (35)$$

and $c^{(i)}$ is not a function of $\check{\boldsymbol{W}}_{i\cdot}$. Minimization of $L_{a2}(\boldsymbol{W})$ w.r.t. $\boldsymbol{W} \in \mathbb{R}^{p\times p}$ is then equivalent to minimization of $L_{a2i}(\check{\boldsymbol{W}}_{i\cdot})$ w.r.t. $\check{\boldsymbol{W}}_{i\cdot} \in \mathbb{R}^{p-1}$ separately for each $i = 1, 2, \cdots, p$, exploiting the fact that $W_{ii} = 0$.

To minimize $L_{ai}(\check{\boldsymbol{W}}_{i\cdot})$ w.r.t. $\check{\boldsymbol{W}}_{i\cdot}$ subject to $W_{ij} \geq 0$, we consider [26] who minimizes $(1/2)\boldsymbol{y}^\top\boldsymbol{P}\boldsymbol{y} - \boldsymbol{y}^\top\boldsymbol{h}$ w.r.t. $\boldsymbol{y} \in \mathbb{R}^q$ subject to $y_\ell \geq 0$ $\forall \ell$, where $\boldsymbol{P} \succ \mathbf{0}$ and $\boldsymbol{h}$ is arbitrary. The monotonically convergent iterative solution of [26] is

$$y_\ell \leftarrow y_\ell \frac{2[\boldsymbol{P}^{(-)}\boldsymbol{y}]_\ell + h_\ell^{(+)} + \delta}{[\mathrm{abs}(\boldsymbol{P})\boldsymbol{y}]_\ell + h_\ell^{(-)} + \delta} \quad (36)$$

where $0 < \delta \ll 1$, $\boldsymbol{P}^{(+)} = \max(\boldsymbol{P}, 0)$, $\boldsymbol{P}^{(-)} = \max(-\boldsymbol{P}, 0)$, $\mathrm{abs}(\boldsymbol{P}) = \boldsymbol{P}^{(+)} + \boldsymbol{P}^{(-)}$, and "max" operation is elementwise.

In our problem we have $\boldsymbol{P} = \boldsymbol{A}$, $\boldsymbol{h} = \boldsymbol{b}^{(i)}$, $\boldsymbol{P}^{(-)} = \boldsymbol{0}$ since all elements of $\boldsymbol{A}$ are non-negative, and $\boldsymbol{P}^{(+)} = \boldsymbol{P} = \boldsymbol{A}$, hence $\mathrm{abs}(\boldsymbol{P}) = \boldsymbol{A}$.

*Update (b-i).* For update (b-i), notice that $L_{b1}(\boldsymbol{V}_1, \boldsymbol{V}_2)$ is completely separable w.r.t. $(V_{1ij}, V_{2ij})$ since

$$L_{b1}(\boldsymbol{V}_1, \boldsymbol{V}_2) = \sum_{i,j}^{p} L_{b1ij}(V_{1ij}, V_{2ij}) \tag{37}$$

$$\begin{aligned} L_{b1ij}(V_{1ij}, V_{2ij}) = &\Big(\lambda|V_{1ij}| + \lambda_g\sqrt{V_{2ij}^2 + V_{1ij}^2}\Big)\,\mathbf{1}_{i\neq j} \\ &+ \frac{\rho}{2}(V_{1ij} - \Omega_{ij}^{(k+1)} + U_{1ij}^{(k)})^2 \\ &+ \frac{\rho}{2}(V_{2ij} - W_{ij}^{(k+1)} + U_{2ij}^{(k)})^2 \end{aligned} \tag{38}$$

Therefore, we solve for

$$(V_{1ij}^{(k+1)}, V_{2ij}^{(k+1)}) \leftarrow \arg\min_{V_{1ij}, V_{2ij}} L_{b1ij}(V_{1ij}, V_{2ij})\,.$$

This is a sparse-group lasso problem. Set $G_{1ij} = \Omega_{ij}^{(k+1)} - U_{1ij}^{(k)}$ and $G_{2ij} = W_{ij}^{(k+1)} - U_{2ij}^{(k)}$. For $i = j$, we have $V_{1ij}^{(k+1)} = G_{1ij}$ and $V_{2ij}^{(k+1)} = G_{2ij}$. For $i \neq j$, following [4], [27], its solution is

$$V_{mij}^{(k+1)} = \begin{cases} S(G_{1ij}, \frac{\lambda}{\rho})\Big(1 - \frac{\lambda_g/\rho}{\sqrt{S^2(G_{1ij}, \frac{\lambda}{\rho}) + G_{2ij}^2}}\Big) & \text{if } m = 1 \\ G_{2ij}\Big(1 - \frac{\lambda_g/\rho}{\sqrt{S^2(G_{1ij}, \frac{\lambda}{\rho}) + G_{2ij}^2}}\Big) & \text{if } m = 2 \end{cases}$$

where for a real scalar $a$ and $\kappa > 0$, $(a)_+ := \max(0, a)$, $S(a, \kappa) := (1 - \kappa/|a|)_+ a$ denotes scalar soft thresholding.

*Update (b-ii).* In update (b-ii), we notice that

$$L_{b2}(\boldsymbol{d}) = \sum_{i=1}^{p} L_{b2i}(d_i) \tag{39}$$

$$L_{b2i}(d_i) = -\,\alpha \ln(d_i) + (\rho/2)(d_i - \boldsymbol{W}_{i.}^{(k+1)}\mathbf{1}_p + u_{3i}^{(k)})^2 \tag{40}$$

Therefore, minimization of $L_{b2}(\boldsymbol{d})$ w.r.t. $\boldsymbol{d} \in \mathbb{R}^p$ is then equivalent to minimization of $L_{b2i}(d_i)$ w.r.t. scalar $d_i$ separately for each $i = 1, 2, \cdots, p$. Setting

$$\begin{aligned} 0 = &\frac{\partial L_{b2i}(d_i)}{\partial d_i} = -\frac{\alpha}{d_i} + \rho\,(d_i - g_i) \\ \text{where } \; g_i = &\boldsymbol{W}_{i.}^{(k+1)}\mathbf{1}_p - u_{3i}^{(k)}\,. \end{aligned} \tag{41}$$

This leads to a quadratic equation in $d_i$. Since the $i$th node degree $d_i$ must be positive, we take

$$d_i^{(k+1)} = \frac{1}{2}\left(g_i + \sqrt{g_i + (4\alpha/\rho)}\right) \tag{42}$$

where $g_i$ is specified in (41). This completes the solution.

### *D. Algorithm*

A pseudocode for the ADMM algorithm described in Sec. III-C is given in Algorithm 1 where we use the stopping (convergence) criterion following [22, Sec. 3.3.1] and varying penalty parameter $\rho$ following [22, Sec. 3.4.1]. The stopping criterion is based on primal and dual residuals being small

**Algorithm 1** Proposed ADMM Algorithm

**Input:** Number of samples $n$, number of nodes $p$, data $\{\boldsymbol{x}(t)\}_{t=1}^n$, $\boldsymbol{x} \in \mathbb{R}^p$, parameters $\lambda_{ij}$, $\lambda_{gij}$, $\alpha$, $\beta$, $\gamma$ and $\rho_0$, tolerances $\tau_{abs}$ and $\tau_{rel}$, variable penalty factor $\mu$, maximum number of iterations $k_{max}$. $\lambda_{ij} = \lambda$, $\lambda_{gij} = \lambda_g$ for the proposed approach and $\lambda_{ij} = \lambda/\hat{\Omega}_{ij}$, $\lambda_{gij} = \lambda_g/\sqrt{\hat{\Omega}_{ij}^2 + \hat{W}_{ij}^2}$ for its adaptive lasso version where $\hat{\Omega}_{ij}$ and $\hat{W}_{ij}$ are the result of non-adaptive version.

**Output:** estimated $\hat{\boldsymbol{W}}$, Laplacian $\hat{\boldsymbol{L}}$ and edge-set $\hat{\mathcal{E}}$

Calculate sample covariance $\hat{\boldsymbol{\Sigma}} = \frac{1}{n}\sum_{t=1}^n \boldsymbol{x}(t)\boldsymbol{x}^\top(t)$ (after centering $\boldsymbol{x}(t)$) and the distance-squared matrix $\hat{\boldsymbol{Z}}$ with $(i,j)$th element $\hat{Z}_{ij} = \frac{1}{n}\sum_{t=1}^n (x_i(t) - x_j(t))^2$.

Initialize: $\boldsymbol{U}_m^{(0)} = \boldsymbol{V}_m^{(0)} = \boldsymbol{0} \in \mathbb{R}^{(p)\times(p)}$, $m = 1, 2$, $\boldsymbol{u}_3^{(0)} = \boldsymbol{0} \in \mathbb{R}^p$, $\boldsymbol{d}^{(0)} = \mathbf{1}_p$, $\boldsymbol{\Omega}^{(0)} = \boldsymbol{W}^{(0)} = (\mathrm{diag}(\hat{\boldsymbol{\Sigma}}))^{-1}$, $\rho^{(0)} = \rho_0$

converged = FALSE, $k = 0$

**while** converged = FALSE AND $k \leq k_{max}$, **do**

  Eigen-decompose $\hat{\boldsymbol{\Sigma}} - \frac{\rho^{(k)}}{\gamma}\left(\boldsymbol{V}_1^{(k)} + \boldsymbol{U}_1^{(k)}\right)$ as $\hat{\boldsymbol{\Sigma}} - \frac{\rho^{(k)}}{\gamma}\left(\boldsymbol{V}_1^{(k)} + \boldsymbol{U}_1^{(k)}\right) = \boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^\top$ with diagonal matrix $\boldsymbol{\Lambda}$ consisting of eigenvalues. Define diagonal matrix $\tilde{\boldsymbol{\Lambda}}$ with $\ell$th diagonal element $\tilde{\Lambda}_{\ell\ell} = (-\Lambda_{\ell\ell} + \sqrt{\Lambda_{\ell\ell}^2 + 4\rho^{(k)}})/(2\rho^{(k)})$. Set $\boldsymbol{\Omega}^{(k+1)} = \boldsymbol{Q}\tilde{\boldsymbol{\Lambda}}\boldsymbol{Q}^\top$.

  Set $\bar{\rho} = \frac{\rho^{(k)}}{1-\gamma}$ and $\boldsymbol{A} = (\beta + \bar{\rho})\boldsymbol{I}_{p-1} + \bar{\rho}\mathbf{1}_{p-1}\mathbf{1}_{p-1}^\top$.

  **for** $i = 1, 2, \cdots, p$ **do**

    Obtain $i$th row $\hat{\boldsymbol{Z}}_{i.}$ of $\hat{\boldsymbol{Z}}$ and delete its $i$th element to yield $\check{\hat{\boldsymbol{Z}}}_{i.} \in \mathbb{R}^{p-1}$. Set $\boldsymbol{b}^{(i)} = \check{\hat{\boldsymbol{Z}}}_{i.} - \bar{\rho}(\check{\boldsymbol{V}}_{2i.}^{(k)} + \check{\boldsymbol{U}}_{2i.}^{(k)} + (d_i^{(k)} + U_{3i}^{(k)})\mathbf{1}_{p-1})$.

    **while** converged = FALSE **do**

      Iterate on the $\ell$th element of $\check{\boldsymbol{W}}_{i.} \in \mathbb{R}^{p-1}$ until convergence as

$$[\check{\boldsymbol{W}}_{i.}]_\ell \leftarrow [\check{\boldsymbol{W}}_{i.}]_\ell \frac{[\boldsymbol{b}^{(i)}]_\ell^+ + \delta}{[\boldsymbol{A}\boldsymbol{b}^{(i)}]_\ell + [\boldsymbol{b}^{(i)}]_\ell^- + \delta}$$

      with initialization $\check{\boldsymbol{W}}_{i.} = \check{\boldsymbol{W}}^{(k)}{}_{i.}$.

    **end while**

    Obtain $\boldsymbol{W}_{i.} \in \mathbb{R}^p$ from converged $\check{\boldsymbol{W}}_{i.} \in \mathbb{R}^{p-1}$ by inserting $W_{ii} = 0$, and set $\boldsymbol{W}_{i.}^{(k+1)} = \boldsymbol{W}_{i.}$

  **end for**

  **for** $i, j = 1, 2, \cdots, p$ **do**

    Set $G_{1ij} = \Omega_{ij}^{(k+1)} - U_{1ij}^{(k)}$ and $G_{2ij} = W_{ij}^{(k+1)} - U_{2ij}^{(k)}$. For $i = j$, set $V_{1ij}^{(k+1)} = G_{1ij}$ and $V_{2ij}^{(k+1)} = G_{2ij}$. Define thresholding operator $S(a, \kappa) := (1 - \kappa/|a|)_+ a$ where $(a)_+ := \max(0, a)$. For $i \neq j$, the $(i,j)$th element of $\boldsymbol{V}_m$, $m = 1, 2$, is updated as ($\tilde{\lambda} = \lambda_{ij}/\rho^{(k)}$)

$$V_{mij}^{(k+1)} = \begin{cases} S(G_{1ij}, \tilde{\lambda})\Big(1 - \frac{\lambda_{gij}/\rho^{(k)}}{\sqrt{S^2(G_{1ij}, \tilde{\lambda}) + G_{2ij}^2}}\Big) \\ \qquad \text{if } m = 1\,, \\ G_{2ij}\Big(1 - \frac{\lambda_{gij}/\rho^{(k)}}{\sqrt{S^2(G_{1ij}, \tilde{\lambda}) + G_{2ij}^2}}\Big) \\ \qquad \text{if } m = 2 \end{cases}$$

  **end for**

17: **for** $i = 1, 2, \cdots, p$ **do**

18: Set $g_i = \boldsymbol{W}_i^{(k+1)} \mathbf{1}_p - u_{3i}^{(k)}$. Update $d_i^{(k+1)} = \frac{1}{2}\left(g_i + \sqrt{g_i + (4\alpha/\rho)}\right)$.

19: **end for**

20: Dual updates

$$\begin{aligned} \boldsymbol{U}_1^{(k+1)} &= \boldsymbol{U}_1^{(k)} + \boldsymbol{V}_1^{(k+1)} - \boldsymbol{\Omega}^{(k+1)} \\ \boldsymbol{U}_2^{(k+1)} &\leftarrow \boldsymbol{U}_2^{(k)} + \boldsymbol{V}_2^{(k+1)} - \boldsymbol{W}^{(k+1)}, \\ \boldsymbol{u}_3^{(k+1)} &\leftarrow \boldsymbol{u}_3^{(k)} + \boldsymbol{d}^{(k+1)} - \boldsymbol{W}^{(k+1)} \mathbf{1}_p . \end{aligned}$$

21: Check convergence. Let

$$\begin{aligned} e_1 &= \sqrt{\| \left[ \boldsymbol{\Omega}^{(k+1)}, \boldsymbol{W}^{(k+1)}, \boldsymbol{W}^{(k+1)} \mathbf{1}_p \right] \|_F^2} \\ e_2 &= \sqrt{\| \left[ \boldsymbol{V}_1^{(k+1)}, \boldsymbol{V}_2^{(k+1)}, \boldsymbol{d}^{((k+1)} \right] \|_F^2} \\ e_3 &= \sqrt{\| \left[ \boldsymbol{U}_1^{(k+1)}, \boldsymbol{U}_2^{(k+1)}, (\sqrt{p-1})\, \boldsymbol{u}_3^{((k+1)} \right] \|_F^2}. \end{aligned}$$

Set tolerances

$$\begin{aligned} \tau_{pri} &= (\sqrt{2p^2 + p})\, \tau_{abs} + \tau_{rel} \max(e_1, e_2) \\ \tau_{dual} &= (\sqrt{2p^2 + p})\, \tau_{abs} + \tau_{rel}\, e_3 / \rho^{(k)} . \end{aligned}$$

Define primary and dual residuals $d_p$ and $d_d$

$$d_p = \sqrt{s_1}, \quad d_d = \rho^{(k)} \sqrt{s_2}$$

where

$$\begin{aligned} s_1 =& \|\boldsymbol{\Omega}^{(k+1)} - \boldsymbol{V}_1^{(k+1)}\|_F^2 + \|\boldsymbol{W}^{(k+1)} - \boldsymbol{V}_2^{(k+1)}\|_F^2 \\ &+ \|\boldsymbol{W}^{(k+1)} \mathbf{1}_p - \boldsymbol{d}^{((k+1)}\|_2^2 \\ s_2 =& \|\boldsymbol{V}_1^{(k+1)} - \boldsymbol{V}_1^{(k)}\|_F^2 + \|\boldsymbol{V}_2^{(k+1)} - \boldsymbol{V}_2^{(k)}\|_F^2 \\ &+ (p-1)\, \|\boldsymbol{d}^{(k+1)} - \boldsymbol{d}^{((k)}\|_2^2 . \end{aligned}$$

If $(d_p \le \tau_{pri})$ AND $(d_d \le \tau_{dual})$, set converged = TRUE .

22: Update penalty parameter $\rho$ :

$$\rho^{(k+1)} = \begin{cases} 2\rho^{(k)} & \text{if } d_p > \mu d_d \\ \rho^{(k)}/2 & \text{if } d_d > \mu d_p \\ \rho^{(k)} & \text{otherwise} . \end{cases}$$

We also need to set $\boldsymbol{U}_m^{(k+1)} = \boldsymbol{U}_m^{(k+1)}/2$, $m = 1, 2$, and $\boldsymbol{u}_3^{(k+1)} = \boldsymbol{u}_3^{(k+1)}/2$ for $d_p > \mu d_d$ and $\boldsymbol{U}_m^{(k+1)} = 2\boldsymbol{U}_m^{(k+1)}$, $m = 1, 2$, and $\boldsymbol{u}_3^{(k+1)} = 2\boldsymbol{u}_3^{(k+1)}$ for $d_d > \mu d_p$.

23: $k \leftarrow k + 1$

24: **end while**

25: Denote the converged values of weighted adjacency and precision matrices as $\boldsymbol{W}$ and $\boldsymbol{\Omega}$, respectively. For $i \neq j$, if both $|\Omega_{ij}| > 0$ and $W_{ij} > 0$, assign edge $\{i, j\} \in \hat{\mathcal{E}}$, else $\{i, j\} \notin \hat{\mathcal{E}}$. Weighted adjacency matrix estimate $\hat{\boldsymbol{W}}$ is defined as nonzero only on $\{i, j\} \in \hat{\mathcal{E}}$ such that $\hat{W}_{ij} = -\Omega_{ij}$ if $\Omega_{ij} \le 0$ and $\hat{W}_{ij} = W_{ij}$ if $\Omega_{ij} > 0$. Clearly $\hat{\boldsymbol{W}} \in \mathcal{W}_p$. Finally, combinatorial Laplacian estimate is $\hat{\boldsymbol{L}} = \hat{\boldsymbol{D}} - \hat{\boldsymbol{W}}$ where $\hat{\boldsymbol{D}}$ is the (diagonal) weighted degree matrix corresponding to $\hat{\boldsymbol{W}}$.

where, in our case, at $(k+1)$st iteration, the primal residual is given by the vectorized matrix

$$\begin{bmatrix} (\boldsymbol{\Omega}^{(k+1)} - \boldsymbol{V}_1^{(k+1)})^\top \\ (\boldsymbol{W}^{(k+1)} - \boldsymbol{V}_2^{(k+1)})^\top \\ (\boldsymbol{W}^{(k+1)} \mathbf{1}_p - \boldsymbol{d}^{((k+1)})^\top \end{bmatrix}$$

and the dual residual by the vectorized matrix

$$\rho^{(k)} \begin{bmatrix} (\boldsymbol{V}_1^{(k+1)} - \boldsymbol{V}_1^{(k)})^\top \\ (\boldsymbol{V}_2^{(k+1)} - \boldsymbol{V}_2^{(k)})^\top \\ \left((\boldsymbol{d}^{(k+1)} - \boldsymbol{d}^{(k)}) \mathbf{1}_{p-1}^\top\right)^\top \end{bmatrix} .$$

For all numerical results presented later, we used $\rho_0 = 2$, $\mu = 10$, and $\tau_{abs} = \tau_{rel} = 10^{-4}$.

### *E. Adaptive Lasso Augmentation*

Lasso and related approaches yield biased estimates [17]. To approximately debias, we will mimic the adaptive lasso approach of [17] to propose an adaptive lasso augmentation of the proposed approach. The basic idea in [17] is to replace $\lambda\Omega_{ij}$ in the penalty with $\lambda\Omega_{ij}/|\hat{\Omega}_{ij}|$ where $\hat{\Omega}_{ij}$ is any consistent estimate of $\Omega_{ij}$, typically a lasso estimate. Mimicking this approach, we propose to replace $\lambda\Omega_{ij}$ and $\lambda_g \sqrt{W_{ij}^2 + \Omega_{ij}^2}$ in (8) with $\lambda\Omega_{ij}/|\hat{\Omega}_{ij}|$ and $\lambda_g \sqrt{W_{ij}^2 + \Omega_{ij}^2}/\sqrt{\hat{W}_{ij}^2 + \hat{\Omega}_{ij}^2}$, respectively, where $\hat{\Omega}_{ij}$ and $\hat{W}_{ij}$ are the result of optimization of $L_{aug}(\boldsymbol{\Omega}, \boldsymbol{W})$ (the non-adaptive version). The modified objective function now is

$$\begin{aligned} \tilde{L}_{aug}(\boldsymbol{\Omega}, \boldsymbol{W}) \;=\; & \gamma \left( \mathrm{tr}(\boldsymbol{\Omega}\hat{\boldsymbol{\Sigma}}) - \ln(|\boldsymbol{\Omega}|) + \lambda \sum_{i \neq j}^{p} \frac{\Omega_{ij}}{|\hat{\Omega}_{ij}|} \right) \\ & + (1-\gamma) f_s(\boldsymbol{W}) + \lambda_g \sum_{i \neq j}^{p} \frac{\sqrt{W_{ij}^2 + \Omega_{ij}^2}}{\sqrt{\hat{W}_{ij}^2 + \hat{\Omega}_{ij}^2}} . \end{aligned} \tag{43}$$

Notice that we make no changes to $f_s(\boldsymbol{W})$ (defined in (4)). It has been shown in [9] that a lasso type penalty does not work for his objective function $f_s(\boldsymbol{W})$.

We optimize twice, first non-adaptive version of $L_{aug}(\boldsymbol{\Omega}, \boldsymbol{W})$, then the adaptive version $\tilde{L}_{aug}(\boldsymbol{\Omega}, \boldsymbol{W})$. The Algorithm 1 applies both times and the modifications required for the second optimization are specified therein.

### *F. Final Estimates*

Suppose we denote the converged values of the weighted adjacency and precision matrices resulting from our proposed approach, whether non-adaptive or adaptive version, as $\boldsymbol{W}$ and $\boldsymbol{\Omega}$, respectively. To calculate connected edges, we take $\{i, j\} \in \hat{\mathcal{E}}$ if both estimated $W_{ij} > 0$ and estimated $|\Omega_{ij}| > 0$, else $\{i, j\} \notin \hat{\mathcal{E}}$, where $\hat{\mathcal{E}}$ is the estimated edge-set. This reflects that fact that both weighted adjacency matrix in $f_s(\boldsymbol{W})$ and precision matrix in $f_L(\boldsymbol{\Omega})$ have nonzero entries if and only if corresponding edges are connected. As noted in [13], $\boldsymbol{W}$ estimated via [9] typically results in a biased estimate of the Laplacian $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{W}$. Ideally, we should have $\boldsymbol{L} = \boldsymbol{\Omega}$ implying that off-diagonal $\boldsymbol{\Omega}^- = -\boldsymbol{W}$. But estimated $\Omega$ is not

guaranteed to have non-positive off-diagonal entries. Based on these considerations, we estimate $\boldsymbol{W}$ as $\hat{\boldsymbol{W}}$ specified by

$$\hat{W}_{ij} = \begin{cases} -\Omega_{ij} & \text{if } \{i,j\} \in \hat{\mathcal{E}} \text{ and } \Omega_{ij} < 0 \\ W_{ij} & \text{if } \{i,j\} \in \hat{\mathcal{E}} \text{ and } \Omega_{ij} > 0 \\ 0 & \text{otherwise .} \end{cases} \tag{44}$$

Clearly $\hat{\boldsymbol{W}} \in \mathcal{W}_p$. Finally, combinatorial Laplacian estimate is $\hat{\boldsymbol{L}} = \hat{\boldsymbol{D}} - \hat{\boldsymbol{W}}$ where $\hat{\boldsymbol{D}}$ is the (diagonal) weighted degree matrix corresponding to $\hat{\boldsymbol{W}}$. By construction, $\hat{\boldsymbol{L}}$ is a combinatorial Laplacian matrix that is positive semidefinite with non-positive off-diagonal entries.

## IV. Simulation Example

We consider Gaussian graphical models based on an **Erdös-Rényi** graph where nodes are connected independently and randomly with probability $p_{er} = 0.03$. In the upper triangular $\Omega$ (inverse covariance), $\Omega_{ij} = 0$ if $\{i,j\} \notin \mathcal{E}$, and $\Omega_{ij}$ is uniformly distributed over $[-0.3, -0.1]$ if $\{i,j\} \in \mathcal{E}$. With $\Omega = \Omega^\top$, we take $\Omega_{ii} = -\sum_{j=1}^{p} \Omega_{ij}$ for every $i$, yielding the combinatorial Laplacian matrix $\boldsymbol{L} = \boldsymbol{\Omega}$. Now add $\kappa \boldsymbol{I}$ to $\boldsymbol{\Omega}$ with $\kappa$ picked to make minimum eigenvalue of $\boldsymbol{\Omega} + \kappa \boldsymbol{I}$ equal to 0.001, and with $\boldsymbol{\Phi}\boldsymbol{\Phi}^\top = (\boldsymbol{\Omega} + \kappa \boldsymbol{I})^{-1}$, we generate $\boldsymbol{x} = \boldsymbol{\Phi}\boldsymbol{w}$ with $\boldsymbol{w} \in \mathbb{R}^p$ as Gaussian $\boldsymbol{w} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_p)$, multivariate Gaussian distribution with zero mean and identity covariance matrix. We generate $n$ i.i.d. observations from $\boldsymbol{x}$ using $p = 100$. Addition of $\kappa \boldsymbol{I}$ yields a generalized Laplacian matrix $\boldsymbol{L} = \boldsymbol{\Omega} + \kappa \boldsymbol{I}$ [13].

We apply seven methods for estimating the true edge-set $\mathcal{E}_0$ and true (off-diagonal) inverse covariance $\boldsymbol{\Omega}_0^-$ (since it equals $-\boldsymbol{W}$ under the Laplacian assumptions, combinatorial or generalized):

(1) Lasso with cost (5), solved using the method of [22, Sec. 6.5] (also used in Sec. III), labeled "Lasso" in Table I.
(2) Smoothness-based graph learning [9] solved via the ADMM approach of [20], labeled "smooth (ADMM [20])" in Table I.
(3) Proposed approach described in Sec. III with $\gamma = .5$, $\lambda_g = 2\lambda$, labeled "L+Sm ($L_{aug}(\boldsymbol{\Omega}, \boldsymbol{W})$)" in Table I.
(4) Smoothness-based graph learning [9] solved via the forward-backward algorithm available in [19] (MATLAB function gsp_learn_graph_log_degrees.m), labeled "smooth [9]" in Table I. It requires one to set small values in estimated $\boldsymbol{W}$ to be set to zero; following [9], [19], all $\hat{W}_{ij} \leq 10^{-4}$ are set to zero.
(5) Generalized graph Laplacian (GGL) method of [13] described in Sec. II-C, using MATLAB function estimate_ggl.m from [24], labeled "GGL [13]" in Table I.
(6) Adaptive lasso, described in Sec. II-D, using the method of [22, Sec. 6.5] but with variable lasso penalty, labeled "Adap Lasso" in Table I.
(7) Adaptive lasso version of the proposed approach described in Sec. III-E, with $\gamma = .5$, $\lambda_g = 2\lambda$, labeled "L+Sm ($\tilde{L}_{aug}(\boldsymbol{\Omega}, \boldsymbol{W})$)" in Table I.

Table I shows the simulation results where the run time in seconds was calculated via MATLAB tic-toc functions on a Window 10 operating system with processor Intel(R) Core(TM) i5-6400T CPU @2.20 GHz with 12 GB RAM, and all ADMM approaches used variable penalty parameter $\rho$, as in Algorithm 1. For each of seven schemes, tuning parameters $\lambda$ and/or $\beta$ ($\alpha$=1, as in [9]) were picked for $n = 200$ via simulations to maximize the $F_1$-score (as in [9], [13]), then $\lambda$ was scaled as $\propto \sqrt{\ln(p)/n}$ [8], [13] while $\beta$ was kept fixed.

The performance measures are $F_1$-score for efficacy in edge detection, and normalized Frobenius error norm in estimating $\boldsymbol{\Omega}_0^-$ (off-diagonal true $\boldsymbol{\Omega}_0$, equaling $-\boldsymbol{W}$), defined as

$$\|c\hat{\boldsymbol{\Omega}}^- - \boldsymbol{\Omega}_0^-\|_F / \|\boldsymbol{\Omega}_0^-\|_F \tag{45}$$

where $c$ is selected as follows. We scale estimate $\hat{\boldsymbol{\Omega}}^-$ of $\boldsymbol{\Omega}_0^-$ (when only signal smoothing is used we take $\hat{\boldsymbol{\Omega}}_0^- = -\hat{\boldsymbol{W}}$), by a scalar $c$ chosen to minimize mean-square error $\|\boldsymbol{\Omega}_0^- - c\hat{\boldsymbol{\Omega}}_0^-\|_F^2$, resulting in $c = \text{tr}(\boldsymbol{\Omega}_0^- \hat{\boldsymbol{\Omega}}_0^-)/\text{tr}(\hat{\boldsymbol{\Omega}}_0^- \hat{\boldsymbol{\Omega}}_0^-)$. In practice, $\boldsymbol{\Omega}_0^-$ is unknown. The above scaling preserves relative weighting among $\Omega_{ij}$ which is what is relevant in applications and is available without knowing $\boldsymbol{\Omega}_0^-$. In applications such as those in [1], [9], [15], [16] that require an estimate of the graph Laplacian $\boldsymbol{L}$, the eigenvectors of $\boldsymbol{L}$ and the relative values of eigenvalues are exploited which do not depend on scaling $c$. As noted in Sec. III-F, the combinatorial Laplacian estimate is $\hat{\boldsymbol{L}} = \hat{\boldsymbol{D}} - \hat{\boldsymbol{W}} = \hat{\boldsymbol{D}} + \hat{\boldsymbol{\Omega}}_0^-$.

Comparing only "Lasso", "smooth (ADMM [20])" and proposed "L+Sm ($L_{aug}(\boldsymbol{\Omega}, \boldsymbol{W})$)", we see that the proposed method significantly improves upon "smooth (ADMM [20])" in estimation of inverse covariance without sacrificing the $F_1$-score, while it improves upon "Lasso" in both $F_1$-score and inverse covariance estimation. Adding adaptive lasso improves both "Lasso" and "L+Sm ($L_{aug}(\boldsymbol{\Omega}, \boldsymbol{W})$)". Indeed, "Adap lasso" outperforms other six approaches for both graphs, but along with "Lasso" it can not guarantee that off diagonal terms of the precision matrix are non-positive. Both "smooth [9]" and "GGL [13]" have poorer $F_1$-score performance, but are significantly faster. Approaches "smooth (ADMM [20])" and "L+Sm ($L_{aug}(\boldsymbol{\Omega}, \boldsymbol{W})$)" are significantly slower since the "inner loop" consisting of iterative solution (36) (line 10 in Algorithm 1) slows it down (we run it for fixed 40 iterations as a stopping criterion is not yet clear or effective), but it may account for superior performance of "smooth (ADMM [20])" over "smooth [9]". Finally, lasso (and adaptive lasso) can be made much faster by using fast algorithms such as [23].

## V. Conclusions

Our objective was to estimate the structure of an undirected weighted graph underlying a set of signals, exploiting both smoothness of the signals as well as their statistics. Structure estimation of a weighted graphs entails estimation of the edge-set $\mathcal{E}$ and the weighted adjacency $\boldsymbol{W}$ (equivalently, the corresponding combinatorial Laplacian $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{W}$). We augmented the smoothness-based objective function of [9] with a penalized log-likelihood objective function with a lasso constraint to improve inverse covariance estimation performance of [9] without sacrificing the $F_1$-score.

| **Model**: | **Erdös-Rényi Graph**: number of nodes $p$=100 | | | | |
|---|---|---|---|---|---|
| | $F_1$ score ($\pm\sigma$) | | | | |
| Lasso | 0.3009 $\pm$0.0200 | 0.2859 $\pm$0.0417 | 0.2867 $\pm$0.0679 | 0.2951 $\pm$0.0458 | 0.3052 $\pm$0.0355 |
| smooth (ADMM [20]) | 0.5018 $\pm$0.1061 | 0.5881 $\pm$0.1037 | 0.6339 $\pm$0.1033 | 0.6339 $\pm$0.1154 | 0.6592 $\pm$0.1273 |
| L+Sm ($L_{aug}(\boldsymbol{\Omega}, \boldsymbol{W})$) | 0.5017 $\pm$0.0578 | 0.5497 $\pm$0.0700 | 0.6129 $\pm$0.0816 | 0.6285$\pm$0.0877 | 0.6965$\pm$0.0937 |
| smooth [9] | 0.1363 $\pm$0.0398 | 0.1420 $\pm$0.0397 | 0.1491 $\pm$0.0399 | 0.1666 $\pm$0.0495 | 0.1586 $\pm$0.0592 |
| GGL [13] | 0.4609 $\pm$0.0232 | 0.4901 $\pm$0.0232 | 0.4984 $\pm$0.0197 | 0.5032 $\pm$0.0175 | 0.4927 $\pm$0.0218 |
| Adap Lasso | 0.6566 $\pm$0.0462 | 0.8412 $\pm$0.0383 | 0.9429 $\pm$0.0196 | 0.9800 $\pm$0.0106 | 0.9987 $\pm$0.0025 |
| L+Sm ($\tilde{L}_{aug}(\boldsymbol{\Omega}, \boldsymbol{W})$) | 0.6657 $\pm$0.0496 | 0.8331 $\pm$0.0380 | 0.9309 $\pm$0.0200 | 0.9710 $\pm$0.0125 | 0.9836 $\pm$0.0125 |
| | Frobenius Error Norm ($\pm\sigma$) | | | | |
| Lasso | 0.9964 $\pm$0.0043 | 0.9898 $\pm$0.0098 | 0.6367 $\pm$0.0914 | 0.4743 $\pm$0.0378 | 0.3720 $\pm$0.0212 |
| smooth (ADMM [20]) | 0.7478 $\pm$0.0449 | 0.6773 $\pm$0.0294 | 0.6457 $\pm$0.0282 | 0.6312 $\pm$0.0272 | 0.6166 $\pm$0.0220 |
| L+Sm ($L_{aug}(\boldsymbol{\Omega}, \boldsymbol{W})$) | 0.6250 $\pm$0.0507 | 0.4550 $\pm$0.0383 | 0.3280 $\pm$0.0250 | 0.2402$\pm$0.0232 | 0.1458$\pm$0.0323 |
| smooth [9] | 0.7621 $\pm$0.0316 | 0.7028 $\pm$0.0236 | 0.6756 $\pm$0.0224 | 0.6562 $\pm$0.0200 | 0.6475 $\pm$0.0166 |
| GGL [13] | 0.6062 $\pm$0.0615 | 0.4204 $\pm$0.0492 | 0.2866 $\pm$0.0317 | 0.1963 $\pm$0.0234 | 0.0848 $\pm$0.0117 |
| Adap Lasso | 0.6808 $\pm$0.0541 | 0.4964 $\pm$0.0485 | 0.3561 $\pm$0.0390 | 0.2540 $\pm$0.0292 | 0.1096 $\pm$0.0189 |
| L+Sm ($\tilde{L}_{aug}(\boldsymbol{\Omega}, \boldsymbol{W})$) | 0.6627 $\pm$0.0573 | 0.4887 $\pm$0.0462 | 0.3550 $\pm$0.0367 | 0.2612 $\pm$0.0281 | 0.1490 $\pm$0.0320 |
| | Time(s) ($\pm\sigma$) | | | | |
| Lasso | 1.2664 $\pm$0.0412 | 1.2483 $\pm$0.0361 | 1.2671 $\pm$0.0393 | 1.2935 $\pm$0.0353 | 1.2923 $\pm$0.0368 |
| smooth (ADMM [20]) | 7.6652 $\pm$0.1276 | 7.6504 $\pm$0.0762 | 7.6550 $\pm$0.0872 | 7.5839 $\pm$0.5866 | 7.6422 $\pm$0.0917 |
| L+Sm ($L_{aug}(\boldsymbol{\Omega}, \boldsymbol{W})$) | 8.9153 $\pm$0.2261 | 8.9272 $\pm$0.4273 | 8.9606 $\pm$0.2111 | 10.6124 $\pm$2.4781 | 9.1852 $\pm$0.3722 |
| smooth [9] | 0.2587 $\pm$0.0068 | 0.2570 $\pm$0.0031 | 0.2568 $\pm$0.0023 | 0.2557 $\pm$0.0024 | 0.2582 $\pm$0.0021 |
| GGL [13] | 0.0652 $\pm$0.0047 | 0.0639 $\pm$0.0013 | 0.0637 $\pm$0.0012 | 0.0636 $\pm$0.0015 | 0.0633 $\pm$0.0014 |
| Adap Lasso | 1.8343 $\pm$0.2762 | 1.6987 $\pm$0.0429 | 1.6394 $\pm$0.0350 | 1.5736 $\pm$0.0416 | 1.4602 $\pm$0.0382 |
| L+Sm ($\tilde{L}_{aug}(\boldsymbol{\Omega}, \boldsymbol{W})$) | 17.8266 $\pm$0.8996 | 18.2939 $\pm$1.3875 | 17.8056 $\pm$0.3764 | 17.6605 $\pm$0.8972 | 17.9927 $\pm$0.5217 |

TABLE I: *Simulation results for Erdös-Rényi graph based on 100 runs.*

## REFERENCES


[1] X. Dong, D. Thanou, M. Rabbat and P. Frossard, "Learning graphs from data," *IEEE Signal Process. Mag.*, pp. 44-63, May 2019.

[2] S.L. Lauritzen, *Graphical models*. Oxford, UK: Oxford Univ. Press, 1996.

[3] J. Whittaker, *Graphical Models in Applied Multivariate Statistics*. New York: Wiley, 1990.

[4] P. Danaher, P. Wang and D.M. Witten, "The joint graphical lasso for inverse covariance estimation across multiple classes," *J. Royal Statistical Society, Series B*, vol. 76, pp. 373-397, 2014.

[5] N. Meinshausen and P. Bühlmann, "High-dimensional graphs and variable selection with the Lasso," *Ann. Statist.*, vol. 34, no. 3, pp. 1436-1462, 2006.

[6] K. Mohan, P. London, M. Fazel, D. Witten and S.I. Lee, "Node-based learning of multiple Gaussian graphical models," *J. Machine Learning Research*, vol. 15, 2014.

[7] J. Friedman, T. Hastie and R. Tibshirani, "Sparse inverse covariance estimation with the graphical lasso," *Biostatistics*, vol. 9, no. 3, pp. 432-441, July 2008.

[8] O. Banerjee, L.E. Ghaoui and A. d'Aspremont, "Model selection through sparse maximum likelihood estimation for multivariate Gaussian or binary data," *J. Machine Learning Research*, vol. 9, pp. 485-516, 2008.

[9] V. Kalofolias, "How to learn a graph from smooth signals," in *Proc. 19th Intern. Conf. Artificial Intelligence & Statistics (AISTATS)*, Cadiz, Spain, 2016.

[10] V. Kalofolias and N. Perraudin, "Large scale graph learning from smooth signals," in *7th Intern. Conf. Learning Representations (ICLR 2019)*, New Orleans, LA, USA, May 6-9, 2019.

[11] X. Dong, D. Thanou, P. Frossard and P. Vandergheynst "Learning Laplacian matrix in smooth graph signal representations," *IEEE Trans. Signal Process.*, vol. 64, no. 23, pp. 6160-6173, Dec. 1, 2016.

[12] E. Pavez and A. Ortega, "Generalized Laplacian precision matrix estimation for graph signal processing," in *Proc. IEEE ICASSP 2016*, Shanghai, China, March 2016, pp. 6350-6354.

[13] H.E. Egilmez, E. Pavez and A. Ortega, "Graph learning from data under Laplacian and structural constraints," *IEEE J. Sel. Topics Signal Process.*, vol. 11, no. 6, pp. 825-841, Sept. 2017.

[14] E. Pavez, H.E. Egilmez and A. Ortega, "Learning graphs with monotone topology properties and multiple connected components," *IEEE Trans. Signal Process.*, vol. 66, no. 9, pp. 2399-2413, May 1, 2018.

[15] M. Belkin and P. Niyogi, "Laplacian eigenmaps and spectral techniques for embedding and clustering.," in *Proc. NIPS*, vol. 14, pp. 585-591, 2001.

[16] M. Belkin, P. Niyogi and V. Sindhwani "Manifold regularization: A geometric framework for learning from labeled and unlabeled examples," *J. Machine Learning Research*, vol. 7, pp. 2399-2434, 2006.

[17] H. Zou, "The adaptive lasso and it oracle properties," *J. American Statistical Assoc.*, vol. 101, pp. 1418-1429, 2006.

[18] N. Komodakis and J.C. Pesquet, "Playing with duality: An overview of recent primal-dual approaches for solving large-scale optimization problems," *IEEE Signal Processing Mag.*, vol. 32, pp. 31-54, 2015.

[19] N. Perraudin, J. Paratte, D. Shuman, V. Kalofolias, P. Vandergheynst and D.K. Hammond, "GSPBOX: A toolbox for signal processing on graphs," *arXiv:1408.5781v2 [cs.IT]*, 15 March 2016.

[20] J.K. Tugnait, "Graph learning from multi-attribute smooth signals," in *Proc. 2020 IEEE Intern. Workshop on Machine Learning for Signal Processing (MLSP 2020)*, Espoo, Finland, Sept. 21-24, 2020, pp. 1-6.

[21] R. Mazumder and T. Hastie, "The graphical lasso: New insights and alternatives," *Electronic J. Statistics*, vol. 6, 2012.

[22] S. Boyd, N. Parikh, E. Chu, B. Peleato and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends in Machine Learning*, vol. 3, no. 1, pp. 1-122, 2010.

[23] C.-J. Hsieh, M.A. Sustik, I.S. Dhillon and P. Ravikumar, "QUIC: quadratic approximation for sparse inverse covariance estimation, " *J. Mach. Learn. Res.*, vol. 15, pp. 2911-2947, 2014.

[24] H.E. Egilmez, E. Pavez and A. Ortega, "GLL: Graph Laplacian learning package, version 1.0," [Online]. Available: https://github.com/STACUSC/Graph_Learning, 2017.

[25] P. Ravikumar, M.J. Wainwright, G. Raskutti and B. Yu, "High-dimensional covariance estimation by minimizing $\ell_1$-penalized log-determinant divergence," *Electronic J. Statistics*, vol. 5, pp. 935-980, 2011.

[26] X. Xiao and D. Chen, "Multiplicative iteration for non-negative quadratic programming," *arXiv:1406.1008v1 [math.NA]*, 4 June 2014.

[27] J. Friedman, T. Hastie and R. Tibshirani, "A note on the group lasso and a sparse group lasso," *arXiv:1001.0736v1 [math.ST]*, 5 Jan 2010.