Scilight

*Applied Mathematics and Statistics*
https://www.sciltp.com/journals/ams

*Article*

Check for updates

# Inferring High-Dimensional Dynamic Networks Changing with Multiple Covariates

Louis Dijkstra, Arne Godt and Ronja Foraita *

Department of Statistics in Epidemiology, Leibniz Institute for Prevention Research and Epidemiology—BIPS, 28359 Bremen, Germany
* Correspondence: foraita@leibniz-bips.de

**How To Cite**: Dijkstra, L.; Godt, A.; Foraita, R. Inferring High-Dimensional Dynamic Networks Changing with Multiple Covariates. *Applied Mathematics and Statistics* **2026**, *3*(1), 4. https://doi.org/10.53941/ams.2026.100004

Received: 23 February 2026
Revised: 26 March 2026
Accepted: 2 April 2026
Published: 21 April 2026

**Abstract:** High-dimensional networks play a key role in understanding complex relationships. These relationships are often dynamic in nature and can change with multiple external factors (e.g., time and groups). Methods for estimating graphical models are often restricted to static graphs or graphs that can change with a single covariate (e.g., time). We propose a novel class of graphical models, the covariate-varying network (CVN), that can change with multiple external covariates. To introduce sparsity, we apply a $L_1$-penalty to the precision matrices of $m \geq 2$ graphs we estimate. These graphs often show a level of similarity. To model this smoothness, we introduce the concept of a 'meta-graph' where each node in the meta-graph corresponds to an individual graph within the CVN. The (weighted) adjacency matrix of the meta-graph represents the strength with which similarity is enforced between the $m$ graphs. The resulting optimization problem is solved by employing an alternating direction method of multipliers. We test our method using a simulation study and we show its applicability by applying it to two real-world data sets in childhood cancer and vaccine genomics. An implementation of the algorithm in R is publicly available under https://bips-hb.r-universe.dev/CVN (accessed on 22 February 2026).

**Keywords:** alternating direction method of multipliers; covariate-varying networks; Gaussian graphical model; graphical LASSO; sparsity

## 1. Introduction

In many scientific fields, network models are essential for describing and understanding complex relationships [1]. In genetic and molecular epidemiology, networks elucidate interactions between molecular regulators governing gene expression such as gene regulatory networks [2] and how microbial communities in the microbiome interact with each other and the host (e.g., [3,4]). In neuroscience, network models map intricate brain connections, aiding in understanding communication and coordination between different regions [5].

Extensive research has been devoted to inferring high-dimensional networks from data [6–8], primarily on static networks where edges remain fixed and unaffected by external influences. In this paper, we use the terms 'graph' and 'network' interchangeably. While these static networks provide valuable insights, it is crucial to also understanding how network structures evolve with respect to external covariates, such as time or group differences. For example, long-term exposure to ultraviolet (UV) radiation can alter gene networks, impairing DNA repair mechanisms and potentially leading to skin cancer [9]. Microbiome networks differ between healthy individuals and those with obesity [10]. In neuroepidemiology, changes in brain connectome interactions due to cholesterol levels and aging indicate Alzheimer's disease [5]. Observing how dynamic networks change with external factors—such as UV radiation, treatment regimes, or cholesterol levels—is fundamental to understanding the underlying biological processes.

Current literature provides methods for estimating networks that vary with a single discrete external covariate, such as time [11–13] or group membership [7,12,14].Wang and Gao [15] introduced a novel parametrization by

**Copyright:** © 2026 by the authors. This is an open access article under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).
**Publisher's Note:** Scilight stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

expressing the precision matrix dependent on covariates in a linear regression format, which allows to investigate high-dimensional data covariate-dependent Gaussian graphical models. In addition, graphical regression approaches have been proposed [16–18], which model the dependence of individual edges or nodes on covariates through local regressions rather than joint likelihoods. While these methods can capture associations with external factors, they do not estimate a coherent joint precision matrix and thus do not directly represent a valid global probabilistic model. Here, we propose to extend joint graphical modeling frameworks to handle multiple, potentially interrelated, discrete covariates simultaneously, enabling a more comprehensive characterization of how network structures change across complex conditions. We present a very general framework for defining a dynamical graphical model of this kind. We call this the covariate-varying network (CVN) model. We present an alternating direction method of multipliers (ADMM) algorithm that can be used to estimate the model on the basis of (potentially high-dimensional) data.

The paper is organized as follows: Section 2.1 introduces static Gaussian graphical models in which the (in)dependency structure is unaffected by external covariates; we simultaneously present much of the notation used throughout the paper. In Section 2.2, we formally define the CVN graphical model, including the concept of a meta-graph to represent the similarities between the various individual graphs. This section also discusses the estimator considered throughout the paper. Additionally, we highlight several known models from the literature that are special cases of the CVN framework.

We solve the estimation problem for the CVN using an ADMM algorithm that we describe in detail in Section 2.4. The computational complexity of the algorithm is determined in Section 2.5. We show how one could interpolate a graph given an estimated CVN in Section 2.6. The two tuning parameters of the model, one regulating sparsity and the other controling the level of similarity between the graphs, need to be chosen. In Section 2.7, we show how this could be done using either the Akaike or Bayesian information criterion (AIC and BIC).

We assess the performance of the method in a simulation study in which we explore to what extent the proposed method can reconstruct a CVN in Section 2.8. The measures used to assess the performance are presented in the latter section as well.

In addition, we apply the method to a real data set from the KiKme study [19], see Section 2.9. To provide a fully reproducible example, a second case study using an influenza vaccine data set with publicly available data is included in Appendix D. The results of both the simulation study and the main case study are presented in Section 3. We end with our conclusions and a discussion in Section 4.

## 2. Methods

### *2.1. Static Networks*

We represent a static network as a probabilistic undirected graphical model with the tuple $\{\mathbf{X}, f(\mathbf{X}), G = (V, E)\}$ where $\mathbf{X} = (X_1, X_2, \ldots, X_p)^\top$ is a real-valued $p$-dimensional random vector with joint density function $f(\mathbf{X})$. The nodes of the graph $G$, $V = \{1, 2, \ldots, p\}$, correspond to the variables $X_1, X_2, \ldots, X_p$. The edges $E \subseteq V \times V$ reflect the (in)dependence structure between the variables in accordance with the density function $f(\mathbf{X})$. The presence or absence of an edge represents whether the two corresponding variables are conditionally independent, i.e., the edge $\{s, t\}$ is in $E$ if $X_s \not\perp\!\!\!\perp X_t \mid \mathbf{X}_{V\setminus\{s,t\}}$, where $\mathbf{X}_{V\setminus\{s,t\}}$ are all variables in $\mathbf{X}$ except $X_s$ and $X_t$. Let $\mathbf{A} = (a_{st})_{p\times p}$ be the graph's adjacency matrix where $a_{st} = \mathbb{1}(\{s, t\} \in E)$ and $\mathbb{1}(\cdot)$ is the indicator function.

We assume the random vector $\mathbf{X}$ to follow a multivariate normal distribution with mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$. Under this assumption, the entries of the precision matrix $\boldsymbol{\Theta} = (\theta_{st})_{p\times p} = \boldsymbol{\Sigma}^{-1}$ are proportional to the partial correlations between the variables. Determining the network structure, i.e., the edge set $E$, is, therefore, equivalent to determining which entries in the precision matrix $\boldsymbol{\Theta}$ are zero and non-zero. In terms of the graph's adjacency matrix, $a_{st} = \mathbb{1}(\theta_{st} \neq 0)$ for all $\{s, t\} \in V \times V$ [20]. Without loss of generality, we assume that the data are centered around the mean $\boldsymbol{\mu} = \mathbf{0}$ for the remainder of the paper.

Suppose we observe $n$ realizations $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n$ of the random vector $\mathbf{X}$. The maximum likelihood estimator of the precision matrix is given by

$$\widehat{\boldsymbol{\Theta}} = \underset{\boldsymbol{\Theta}\succ 0}{\arg\min}\, \ell(\boldsymbol{\Theta}) = \underset{\boldsymbol{\Theta}\succ 0}{\arg\min}\, \frac{n}{2}\left[\operatorname{trace}\left(\widehat{\boldsymbol{\Sigma}}\boldsymbol{\Theta}\right) - \log\det\left(\boldsymbol{\Sigma}\right)\right], \tag{1}$$

where $\boldsymbol{\Theta} \succ 0$ denotes positive definiteness, $\ell(\boldsymbol{\Theta})$ is the log-likelihood function, 'trace' and 'det' denote the trace and determinant of a matrix, and $\widehat{\boldsymbol{\Sigma}} = n^{-1}\sum_{i=1}^{n} \boldsymbol{x}_i\boldsymbol{x}_i^\top$ is the empirical covariance matrix (see, for example [21]). However, this estimator does lead to dense graphs. Moreover, in high-dimensional settings, the empirical covariance matrix $\widehat{\boldsymbol{\Sigma}}$ is rank deficient and cannot be used to estimate the precision matrix $\widehat{\boldsymbol{\Theta}}$. One can estimate $\widehat{\boldsymbol{\Theta}}$ by imposing a sparse graph structure and, hence, a sparse precision matrix. A popular method to estimate sparse networks

is the Graphical Least Absolute Selection and Shrinkage Operator (GLASSO; Banjeree and Ghaoui [22] and Friedman et al. [8]) which places a $L_1$-norm penalty on the precision matrix in order to 'shrink' its off-diagonal entries to zero. It involves solving the following convex optimization problem:

$$\widehat{\boldsymbol{\Theta}}_{\text{GLASSO}} = \underset{\boldsymbol{\Theta} \succ 0}{\arg\min}\, \ell(\boldsymbol{\Theta}) + \lambda \left\|\boldsymbol{\Theta}\right\|_1^{\text{off}}. \tag{2}$$

Here, $\lambda > 0$ is the tuning parameter and $\left\|\boldsymbol{\Theta}\right\|_1^{\text{off}} = \sum_{s \neq t} |\theta_{st}|$ is the off-diagonal $L_1$-norm of the precision matrix.

### *2.2. The Covariate-Varying Network Model*

We represent a covariate-varying network (CVN) graphical model with the quintuple

$$\text{CVN} = \{\mathbf{X}, \mathbf{U}, \mathcal{U}, f(\mathbf{X} \mid \mathbf{U}), \{G(u) = (V, E(u))\}_{u \in \mathcal{U}}\}, \tag{3}$$

where $\mathbf{X} = (X_1, X_2 \ldots, X_p)^\top$ is a $p$-dimensional random vector and $\mathbf{U} = (U_1, U_2, \ldots, U_q)^\top$ is a random vector representing $q$ external discrete covariates, i.e., variables that are not included in the graph with $(K_1, \ldots, K_q)^\top$ categories. The vector $\mathbf{U}$ lies in the discrete space $\mathcal{U}$ with cardinality $m \leq \prod_{k=1}^{q} K_k$. The joint density function of $\mathbf{X}$ conditioned on $\mathbf{U}$ is $f(\mathbf{X} \mid \mathbf{U})$. The fifth element of the CVN is a set of $m$ graphs, one for each value $u$ in $\mathcal{U}$. The vertices of $G(u)$, $V = \{1, 2, \ldots, p\}$, correspond to the variables $X_1, X_2, \ldots, X_p$ and do not change with $\mathbf{U}$. The (in)dependence structure between the variables is captured by the edge set $E(u)$ which can change with $\mathbf{U}$. Specifically, the edge $\{s, t\}$ is in $E(u)$ if $X_s \not\perp\!\!\!\perp X_t \mid U = u$ and $\mathbf{X}_{V \setminus \{s,t\}}$. We denote the adjacency matrix of graph $G(u)$ as $\mathbf{A}(u) = (a_{st}(u))_{p \times p}$ where $a_{st}(u) = \mathbb{1}(\{s, t\} \in E(u))$. In this paper, we are interested in estimating the graphs $\{G(u) = (V, E(u))\}_{u \in \mathcal{U}}$.

We assume throughout that $\mathbf{X} \mid \mathbf{U} = u$ follows a multivariate normal distribution with mean $\boldsymbol{\mu}(u) = \mathbf{0}$ and covariance matrix $\boldsymbol{\Sigma}(u)$. Under the normality assumption, the entries of the precision matrices correspond with the edge sets of the different graphs. Let $\boldsymbol{\Theta}(u) = \boldsymbol{\Sigma}(u)^{-1} = (\theta_{st}(u))_{p \times p}$ be the precision matrix, then the edge $\{s, t\}$ is in $E(u)$ if $\theta_{st}(u) \neq 0$. Hence, estimating the structure of the graphs $\{G(u)\}_{u \in \mathcal{U}}$, corresponds to determining the zero and non-zero entries in the precision matrices $\{\boldsymbol{\Theta}(u)\}_{u \in \mathcal{U}}$.

We index each element in $\mathcal{U}$ by assigning them a unique element from the index set $\mathcal{I} = \{1, 2, \ldots, m\}$. If $i \in \mathcal{I}$ is the index of $u \in \mathcal{U}$, we write $G_i = G(u)$, $E_i = E(u)$, $\boldsymbol{\Sigma}_i = \boldsymbol{\Sigma}(u)$, $\boldsymbol{\Theta}_i = \boldsymbol{\Theta}(u)$ and $\mathbf{A}_i = \mathbf{A}(u)$. The $(s, t)$ entry in the $i$-th precision matrix is written as $\theta_{st}^{(i)}$. The collection of all precision matrices is denoted by $\boldsymbol{\Theta}_{\text{CVN}} = \{\boldsymbol{\Theta}_i\}_{i=1}^{m}$.

For each value of $u \in \mathcal{U}$, we obtain $n_i > 0$ observations: $\left(\boldsymbol{x}_1^{(i)}, \boldsymbol{x}_2^{(i)}, \ldots, \boldsymbol{x}_{n_i}^{(i)}\right)$ where $\boldsymbol{x}_j^{(i)} \in \mathbb{R}^p$ and $j = 1, \ldots, n_j$. Similarly to Equation (1), we can express the log-likelihood function for precision matrix $\boldsymbol{\Theta}_i$ as

$$\ell_i(\boldsymbol{\Theta}_i) = \frac{n_i}{2} \cdot \left[\text{trace}\left(\widehat{\boldsymbol{\Sigma}}_i \boldsymbol{\Theta}_i\right) - \log\det\left(\boldsymbol{\Sigma}_i\right)\right], \tag{4}$$

where $\widehat{\boldsymbol{\Sigma}}_i = {n_i}^{-1} \sum_{j=1}^{n_i} \boldsymbol{x}_j^{(i)} (\boldsymbol{x}_j^{(i)})^\top$ is the empirical covariance matrix. Overall, the log-likelihood for the CVN model itself is simply the sum over all graphs, i.e., $\sum_{i=1}^{m} \ell_i(\boldsymbol{\Theta}_i)$.

To introduce sparsity, we apply, just as for the GLASSO in Equation (2), the off-diagonal $L_1$-norm penalty for each individual graph:

$$\underset{\{\boldsymbol{\Theta}_i\}_{i=1}^{m}}{\arg\min} \sum_{i=1}^{m} \ell_i(\boldsymbol{\Theta}_i) + \lambda_1 \sum_{i=1}^{m} \left\|\boldsymbol{\Theta}_i\right\|_1^{\text{off}},$$

where $\lambda_1 > 0$ is the GLASSO tuning parameter. Larger values of $\lambda_1$ increase the penalty and result in a sparser estimate of the precision matrices.

In many applications, we expect some similarities between the graphs. For example, two networks observed on two consecutive time points tend to have similar edge sets, while networks observed on far removed time points show less similarity. We refer to enforcing similarity between certain graphs as *smoothing*. For introducing smoothness across the $m$ graphs, we first define a *meta-graph*. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E}, w)$ be a weighted, undirected graph where each node in the node set $\mathcal{V} = \{1, 2, \ldots, m\}$ corresponds to the graphs $G_1, G_2, \ldots, G_m$, $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$, and $w : \mathcal{E} \to [0, 1]$ is the weight function that assigns a value between 0 and 1 to each edge $e \in \mathcal{E}$. We denote the symmetric weighted adjacency matrix of the meta-graph by $\mathbf{W} = (w_{ij})_{m \times m}$ where $w_{ij} = w_{ji} = w(\{i, j\})$.

The weight matrix reflects some prior knowledge about the similarity of how the $m$ graphs might change with the external covariates. For a consistent structure estimation, we must impose an additional smoothness constraint

on $\{\boldsymbol{\Theta}_i\}_{i=1}^m$ to detect graph-wise changes in the underlying (in)dependence structure. We define the CVN estimator of $\boldsymbol{\Theta}_{\text{CVN}} = \{\boldsymbol{\Theta}_i\}_{i=1}^m$ as

$$\widehat{\boldsymbol{\Theta}}_{\text{CVN}} = \underset{\{\boldsymbol{\Theta}_i\}_{i=1}^m}{\arg\min} \sum_{i=1}^m \ell_i(\boldsymbol{\Theta}_i) + \lambda_1 \sum_{i=1}^m \|\boldsymbol{\Theta}_i\|_1^{\text{off}} + \lambda_2 \sum_{i<j} w_{ij} \cdot \|\boldsymbol{\Theta}_i - \boldsymbol{\Theta}_j\|_1^{\text{off}}, \tag{5}$$

where $\lambda_2 \geq 0$ is the *smoothing tuning parameter*. The term $w_{ij} \|\boldsymbol{\Theta}_i - \boldsymbol{\Theta}_j\|_1^{\text{off}}$ penalizes the differences between the edge sets of graph $G_i$ and graph $G_j$. The value $w_{ij}$ encourages a similar structure between graphs $G_i$ and $G_j$ whereas $\lambda_2$ regulates how strongly differences between all $\binom{m}{2}$ pairs of graphs are penalized. To provide intuition for the smoothing penalty, Figure A1 in Appendix C shows example pairs of graphs with increasing values of $\|\boldsymbol{\Theta}_i - \boldsymbol{\Theta}_j\|_1^{\text{off}}$.

### *2.3. Special Cases of Covariate-Varying Networks*

By choosing the space $\mathcal{U}$, the indexation $\mathcal{I}$ and the weighted adjacency matrix $\mathbf{W}$ of the meta-graph appropriately, we can express various models from the literature as special cases of the CVN graphical model.

#### 2.3.1. Time-Varying Graphical LASSO (TVGL)

Suppose we observe graphs at $T$ time points, i.e., $\mathcal{U} = \{t_1, t_2, \ldots, t_T\}$. We define the index set $\mathcal{I} = \{1, 2, \ldots, T\}$ where $i \in \mathcal{I}$ corresponds to time point $t_i$. Consequently, the meta-graph $\mathcal{G}_{\text{TVGL}}$ has $T$ nodes, each corresponding to a single graph, and only graphs on consecutive time-points are smoothed. We define the $(T \times T)$-dimensional adjacency matrix of $\mathcal{G}_{\text{TVGL}}$ as

$$\mathbf{W}_{\text{TVGL}} = \left(w_{ij}^{\text{TVGL}}\right)_{T \times T} \qquad \text{where } w_{ij}^{\text{TVGL}} = \begin{cases} 1 & \text{if } |i-j| = 1 \\ 0 & \text{otherwise.} \end{cases} \tag{6}$$

The time-varying graphical LASSO $\{\widehat{\boldsymbol{\Theta}}_t^{\text{TVGL}}\}_{t=1}^T$ can then be estimated by solving

$$\{\widehat{\boldsymbol{\Theta}}_t^{\text{TVGL}}\}_{t=1}^T = \underset{\{\boldsymbol{\Theta}_t\}_{t=1}^T}{\arg\min} \sum_{t=1}^T \ell_t(\boldsymbol{\Theta}_t) + \lambda_1 \sum_{t=1}^T \|\boldsymbol{\Theta}_t\|_1^{\text{off}} + \lambda_2 \sum_{t=2}^T \|\boldsymbol{\Theta}_t - \boldsymbol{\Theta}_{t-1}\|_1^{\text{off}} \tag{7}$$

which is equivalent to the definition in Hallac et al. [12] and Monti et al. [13]. Figure 1 shows an example of a TVGL in which each of the $T$ graphs has five nodes corresponding to $X_1, X_2, \ldots, X_5$.

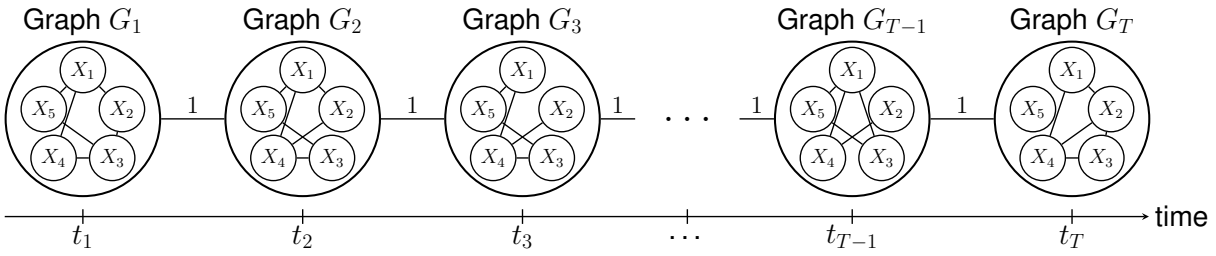



**Figure 1.** The Time-Varying Graphical LASSO (TVGL) as a special case of the CVN model, see Equations (6) and (7). A total of $T$ graphs $G_i$ with $p = 5$ variables $X_1, X_2, \ldots, X_5$ are observed at time points $t_1, t_2, \ldots, t_T$. The circle around each graph represents a node in the meta-graph $\mathcal{G}_{\text{TVGL}}$; the links between the graphs are the meta-graph's edges. Graphs on consecutive time points are smoothed with weight 1.

#### 2.3.2. Fused Graphical LASSO (FGL)

When there is only one external covariate, the CVN reduces in some cases to the fused graphical LASSO (FGL) from Danaher et al. [7]. The FGL takes advantage of structural similarities across different groups to estimate $m$ graphical models that share common edges. Let $m$ be the number of categories of the single external variable $U$, and for each category there are $n_1, n_2, \ldots, n_m > 0$ observations. As before, we choose the index set $\mathcal{I} = \{1, 2, \ldots, m\}$ where the graph of $i$-th category is denoted with $G_i$. The CVN reduces to the FGL with the fused graphical LASSO penalty term if the $(m \times m)$-dimensional weight matrix is defined as

$$\mathbf{W}_{\text{FGL}} = (w_{ij}^{\text{FGL}})_{m\times m} \qquad \text{where } w_{ij}^{\text{FGL}} = \begin{cases} 1 & \text{if } i \neq j \\ 0 & \text{otherwise.} \end{cases} \tag{8}$$

This leads to the following minimization problem:

$$\{\widehat{\boldsymbol{\Theta}}_i^{\text{FGL}}\}_{i=1}^m = \underset{\{\boldsymbol{\Theta}_i\}_{i=1}^m}{\arg\min} \sum_{i=1}^m \ell_i(\boldsymbol{\Theta}_i) + \lambda_1 \sum_{i=1}^m \|\boldsymbol{\Theta}_i\|_1^{\text{off}} + \lambda_2 \sum_{i<j} \|\boldsymbol{\Theta}_i - \boldsymbol{\Theta}_j\|_1^{\text{off}}, \tag{9}$$

which is the FGL [7]. Expressed as CVN this means that the meta-graph is fully connected with weights of 1. The FGL penalty enforces that every graph is smoothed with every other graph. See Figure 2 for a visual representation of this model.

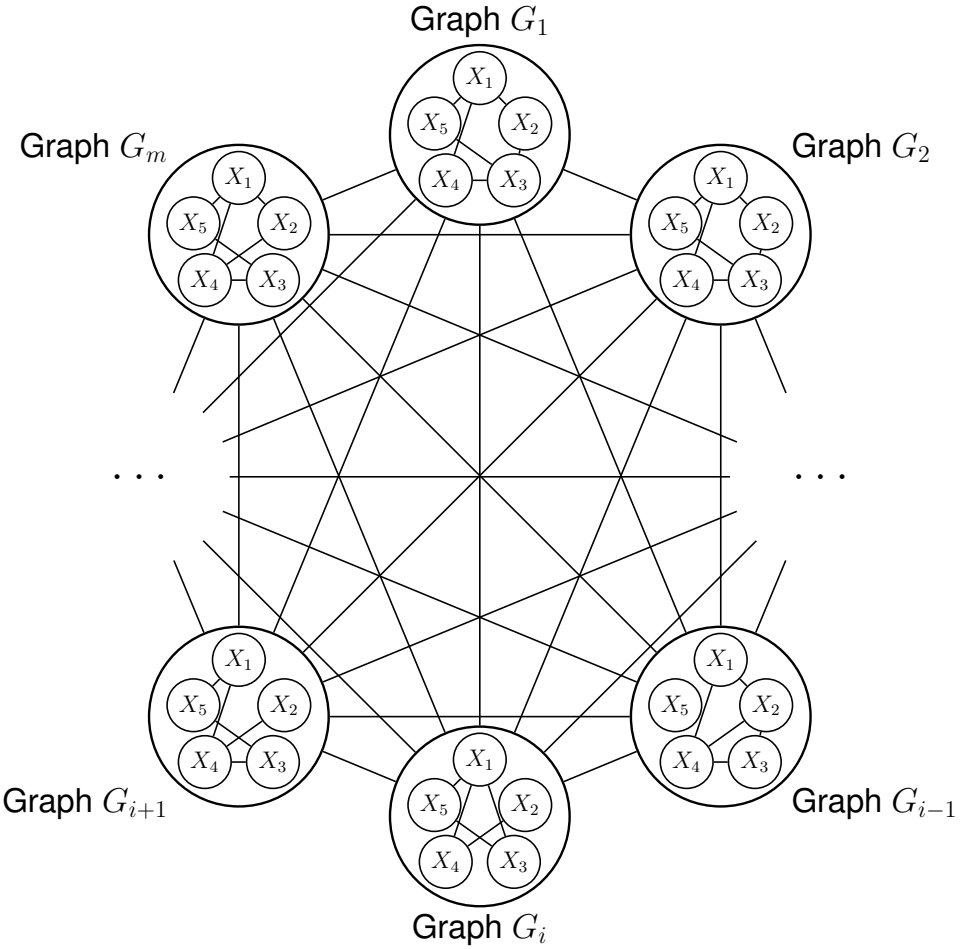



**Figure 2.** The joint graphical LASSO with the fused graphical LASSO (FGL) penalty term as a special case of the CVN model, see Equations (8) and (9). A total of $m$ graphs with $p = 5$ variables are observed. Just as in Figure 1, each graph represents a node in the meta-graph $\mathcal{G}_{\text{FGL}}$ and links between the graphs are the meta-graph's edges.

### 2.3.3. Multiple Static Graphical Models

A trivial case is when no structural constraints are imposed on the $m$ graphs. The weight matrix is hence defined as $\mathbf{W}_0 = \mathbf{0}$. The CVN reduces then to $m$ independent GLASSO problems as in Equation (2):

$$\{\widehat{\boldsymbol{\Theta}}_i^0\}_{i=1}^m = \underset{\{\boldsymbol{\Theta}_i\}_{i=1}^m}{\arg\min} \sum_{i=1}^m \ell_i(\boldsymbol{\Theta}_i) + \lambda_1 \sum_{i=1}^m \|\boldsymbol{\Theta}_i\|_1^{\text{off}} = \left\{ \underset{\boldsymbol{\Theta}_i}{\arg\min}\, \ell_i(\boldsymbol{\Theta}_i) + \lambda_1 \|\boldsymbol{\Theta}_i\|_1^{\text{off}} \right\}_{i=1}^m.$$

Figure 3 shows a visual representation.

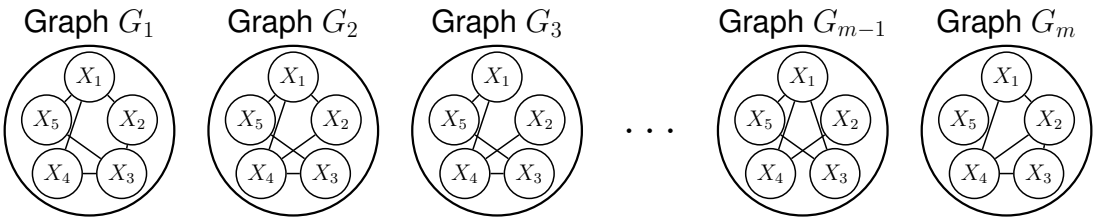



**Figure 3.** A trivial case with $\mathbf{W}_0 = \mathbf{0}$ without smoothing the $m$ graphs.

### *2.4. An Alternating Direction Method of Multipliers Algorithm for the Covariate-Varying Network Problem*

Estimating the CVN graphical model defined in Equation (3) requires us to solve the optimization problem in Equation (5). We propose to do this using an Alternating Direction Method of Multipliers (ADMM) algorithm [23]. By doing so, we can split the original problem into three subproblems, which in turn can be solved either analytically or numerically.

First, let us introduce a set of *consensus variables* $\mathbf{Z} = \{\mathbf{Z}_i\}_{i=1}^m$ where $\mathbf{Z}_i = \mathbf{\Theta}_i$ for all $i$. In addition, we write $\mathbf{\Theta} = \{\mathbf{\Theta}_i\}_{i=1}^m$. We can then express Equation (5) as the equality-constrained convex optimization problem:

$$\begin{gathered}\underset{\mathbf{\Theta},\mathbf{Z}}{\arg\min} \sum_{i=1}^{m} \ell_i(\mathbf{\Theta}_i) + \lambda_1 \sum_{i=1}^{m} \|\mathbf{Z}_i\|_1^{\text{off}} + \lambda_2 \sum_{i<j} w_{ij} \|\mathbf{Z}_i - \mathbf{Z}_j\|_1^{\text{off}} \\ \text{subject to } \mathbf{\Theta}_i - \mathbf{Z}_i = \mathbf{0} \text{ and } \mathbf{\Theta}_i, \mathbf{Z}_i \succ 0 \text{ for } i = 1, 2, \ldots, m.\end{gathered} \tag{10}$$

Thus, the problem does not have to be solved jointly but can be solved for $\mathbf{Z}$ and $\mathbf{\Theta}$ separately. The corresponding augmented Lagrangian is

$$\begin{aligned}\mathcal{L}_\rho(\mathbf{\Theta}, \mathbf{Z}, \mathbf{Y}) = &\sum_{i=1}^{m} \ell_i(\mathbf{\Theta}_i) + \lambda_1 \sum_{i=1}^{m} \|\mathbf{Z}_i\|_1^{\text{off}} + \lambda_2 \sum_{i<j} w_{ij} \|\mathbf{Z}_i - \mathbf{Z}_j\|_1^{\text{off}} \\ &+ (\rho/2) \sum_{i=1}^{m} \left[ \|\mathbf{\Theta}_i - \mathbf{Z}_i + \mathbf{Y}_i\|_F^2 - \|\mathbf{Y}_i\|_F^2 \right],\end{aligned} \tag{11}$$

where $\mathbf{Y} = \{\mathbf{Y}_i\}_{i=1}^m$ is a set of *scaled dual variables*, $\|\mathbf{A}\|_F^2 = \left(\sum_{s,t} a_{st}^2\right)^{1/2}$ is the Frobenius norm, and $\rho > 0$ is the ADMM's *dual update step length* [7,12,23].

Minimizing the function in Equation (11) is equivalent to minimizing the original problem in Equation (5). The advantage, however, of this alternative definition, is that Equation (10) can now be solved by iteratively updating $\mathbf{\Theta}$, $\mathbf{Z}$ and $\mathbf{Y}$ [23]. The update steps for the ADMM are

$$\text{Likelihood update: } \mathbf{\Theta}(k+1) = \underset{\mathbf{\Theta}}{\arg\min}\ \mathcal{L}_\rho(\mathbf{\Theta}, \mathbf{Z}(k), \mathbf{Y}(k)), \tag{12}$$

$$\text{Constraint update: } \mathbf{Z}(k+1) = \underset{\mathbf{Z}}{\arg\min}\ \mathcal{L}_\rho(\mathbf{\Theta}(k+1), \mathbf{Z}, \mathbf{Y}(k)) \text{ and} \tag{13}$$

$$\text{Dual update: } \mathbf{Y}(k+1) = \underset{\mathbf{Y}}{\arg\min}\ \mathcal{L}_\rho(\mathbf{\Theta}(k+1), \mathbf{Z}(k+1), \mathbf{Y}),$$

where $k$ is the iteration step count. We find that the update step for $\mathbf{Y}(k+1)$ has the analytical solution

$$\mathbf{Y}_i(k+1) = \mathbf{Y}_i(k) + [\mathbf{\Theta}_i(k+1) - \mathbf{Z}_i(k+1)] \quad \text{for } i = 1, 2, \ldots, m. \tag{14}$$

The update steps for $\mathbf{\Theta}(k+1)$ and $\mathbf{Z}(k+1)$ have to be solved numerically. The first has a known solution [7]. The latter is more involved and is discussed in Section 2.4.2.

The algorithm iterates until convergence is reached. To guarantee convergence to an optimal solution, the constraint $\mathbf{\Theta}_i - \mathbf{Z}_i = \mathbf{0}$ must be satisfied while the augmented Lagrangian forms (12) and (13) have to be minimized. In each iteration, the minimization constraint on the dual variable is satisfied, which ensures that the constraint update is satisfied by construction [13,24]. The likelihood update step holds asymptotically as $k \to \infty$ using the following stopping criterion [7,25]:

$$\sum_{i=1}^{m} \|\mathbf{\Theta}_i(k+1) - \mathbf{\Theta}_i(k)\|_1 \Big/ \sum_{i=1}^{m} \|\mathbf{\Theta}_i(k)\|_1 < \epsilon. \tag{15}$$

As a default, we choose $\epsilon = 10^{-5}$ as our convergence threshold and control the step size by setting $\rho = 1$.

#### 2.4.1. Update Step for $\boldsymbol{\Theta}$

The update step for the precision matrices $\boldsymbol{\Theta}(k+1)$ in Equation (12) can be divided into $m$ distinct problems, one for each precision matrix:

$$\begin{aligned}\boldsymbol{\Theta}_i(k+1) &= \arg\min_{\boldsymbol{\Theta}_i} \; \mathcal{L}_\rho(\boldsymbol{\Theta}_i, \mathbf{Z}_i(k), \mathbf{Y}_i(k)) \\ &= \arg\min_{\boldsymbol{\Theta}_i} \; \frac{n_i}{2}\left[\text{trace}\left(\widehat{\boldsymbol{\Sigma}}_i\boldsymbol{\Theta}_i\right) - \log\det\left(\boldsymbol{\Sigma}_i\right)\right] + \\ &\qquad (\rho/2)\left\|\mathbf{Y}_i(k) + [\boldsymbol{\Theta}_i - \mathbf{Z}_i(k)]\right\|_F^2\end{aligned} \tag{16}$$

for $i = 1, 2, \ldots, m$. The precision matrix $\boldsymbol{\Theta}_i(k+1)$ can be determined numerically. Due to $\rho > 0$, one must choose $\boldsymbol{\Theta}_i$ in such a way that $\boldsymbol{\Theta}_i$ both minimizes the log-likelihood and is in the proximity of $\mathbf{Z}_i(k)$. The degree to which this must be enforced depends on both $\rho$ and $\mathbf{Y}_i(k)$. Others have shown that Equation (16) can be solved numerically by rescaling an eigenvalue decomposition [13, 26]. More precisely, let $\mathbf{Q}_i\boldsymbol{\Gamma}_i\mathbf{Q}_i^\top$ be the eigenvalue decomposition of $\widehat{\boldsymbol{\Sigma}}_i - (\rho/n_i)\left[\mathbf{Z}_i(k) - \mathbf{Y}_i(k)\right]$, where $\mathbf{Q}_i$ the columns are the eigenvectors and $\boldsymbol{\Gamma}_i = \text{diag}(\gamma_1, \gamma_2, \ldots, \gamma_p)$ is a diagonal matrix with the corresponding eigenvalues on the diagonal. We rescale this matrix such that $\widetilde{\boldsymbol{\Gamma}}_i = \text{diag}(\widetilde{\gamma}_{i1}, \widetilde{\gamma}_{i2}, \ldots, \widetilde{\gamma}_{ip})$ where $\widetilde{\gamma}_{ij} = \dfrac{n_i}{2\rho}\left(\left(\gamma_{ij}^2 + 4\rho/n_i\right)^{\frac{1}{2}} - \gamma_{ij}\right)$ for $j = 1, 2, \ldots, p$. The update step from Equation (16) then equals

$$\boldsymbol{\Theta}_i(k+1) = \mathbf{Q}_i\widetilde{\boldsymbol{\Gamma}}_i\mathbf{Q}_i^\top. \tag{17}$$

Note that each $\boldsymbol{\Theta}_i(k+1)$ is symmetric. Furthermore, all $\widetilde{\gamma}_i$ are strictly positive, ensuring that each $\boldsymbol{\Theta}_i(k+1)$ is positive definite.

The computationally most challenging part of the update step is the eigenvalue decomposition of a $(p \times p)$-matrix with a complexity of $\mathcal{O}(p^3)$. Since this is needed for each of the $m$ graphs, the $\boldsymbol{\Theta}$-update step's complexity is $\mathcal{O}(mp^3)$.

#### 2.4.2. Update Step for $\mathbf{Z}$

The minimization of the augmented Lagragian with respect to $\mathbf{Z}$ in Equation (13) is equal to

$$\begin{aligned}\mathbf{Z}(k+1) &= \arg\min_{\mathbf{Z}} \mathcal{L}_\rho(\boldsymbol{\Theta}(k+1), \mathbf{Z}, \mathbf{Y}(k+1)) \\ &= \arg\min_{\mathbf{Z}} \; (\rho/2)\sum_{i=1}^m \left\|\mathbf{Z}_i - \boldsymbol{\Theta}_i(k+1) - \mathbf{Y}_i(k)\right\|_F^2 + \lambda_1\sum_{i=1}^m \left\|\mathbf{Z}_i\right\|_1^{\text{off}} \\ &\qquad + \lambda_2\sum_{i<j} w_{ij}\left\|\mathbf{Z}_i - \mathbf{Z}_j\right\|_1^{\text{off}}.\end{aligned} \tag{18}$$

In contrast to $\boldsymbol{\Theta}(k+1)$, for which we could divide the update step into $m$ distinct problems, we need to solve the problem for $\mathbf{Z}(k+1)$ jointly due to the smoothness term $\|\mathbf{Z}_i - \mathbf{Z}_j\|_1^{\text{off}}$. However, we can divide the problem into multiple, independent optimization problems, one for each node pair $\{s, t\}$, corresponding to a potential edge in the graphs. Let $\{s, t\}$ be the potential edge between nodes $X_s$ and $X_t$. Equation (18) can then be written as

$$\begin{aligned}\mathbf{Z}(k+1) = \arg\min_{\mathbf{Z}} \; &(\rho/2)\sum_{i=1}^m\sum_{s,t}\left[z_{st}^{(i)} - \theta_{st}^{(i)}(k+1) - y_{st}^{(i)}(k)\right]^2 \\ &+ \lambda_1\sum_{i=1}^m\sum_{s\neq t}|z_{st}^{(i)}| + \lambda_2\sum_{i<j}\sum_{s\neq t} w_{ij}|z_{st}^{(i)} - z_{st}^{(j)}|,\end{aligned}$$

which is completely separable with respect to each node pairs $\{s,t\}$. Let

$$\begin{aligned}\boldsymbol{\theta}_{st} &= \left(\theta_{st}^{(1)}, \theta_{st}^{(2)}, \ldots, \theta_{st}^{(m)}\right)^\top, \\ \boldsymbol{z}_{st} &= \left(z_{st}^{(1)}, z_{st}^{(2)}, \ldots, z_{st}^{(m)}\right)^\top, \quad \text{and} \\ \boldsymbol{y}_{st} &= \left(y_{st}^{(1)}, y_{st}^{(2)}, \ldots, y_{st}^{(m)}\right)^\top\end{aligned}$$

be $m$-dimensional vectors with the $(s,t)$ entries of the matrices $\boldsymbol{\Theta}$, $\mathbf{Z}$ and $\mathbf{Y}$, respectively. There is an analytical solution for the diagonal elements of $\boldsymbol{Z}(k+1)$, specifically:

$$\boldsymbol{z}_{ss}(k+1) = \underset{\boldsymbol{z}_{ss}\in\mathbb{R}^m}{\arg\min}\ \|\boldsymbol{z}_{ss} - \boldsymbol{\theta}_{ss}(k+1) - \boldsymbol{y}_{ss}(k)\|_2^2 = \boldsymbol{\theta}_{ss}(k+1) + \boldsymbol{y}_{ss}(k).$$

For $s \neq t$, we can update each $(s,t)$ entry by solving

$$\begin{aligned}\boldsymbol{z}_{st}(k+1) = \underset{\boldsymbol{z}_{st}\in\mathbb{R}^m}{\arg\min}\ &(\rho/2)\,\|\boldsymbol{z}_{st} - \boldsymbol{\theta}_{st}(k+1) - \boldsymbol{y}_{st}(k)\|_2^2 \\ &+ \lambda_1\,\|\boldsymbol{z}_{st}\|_1 + \lambda_2 \sum_{i<j} w_{ij}|z_{st}^{(i)} - z_{st}^{(j)}|.\end{aligned}$$

We can express this problem as a weighted Fused LASSO Signal Approximator (wFLSA). For ease of notation, let us define $\boldsymbol{\beta} = \boldsymbol{z}_{st}$ and $\boldsymbol{y} = \boldsymbol{\theta}_{st}(k+1) + \boldsymbol{y}_{st}(k)$; we then find that our problem can be written as

$$\boldsymbol{\beta}(k+1) = \underset{\boldsymbol{\beta}\in\mathbb{R}^m}{\arg\min}\ \frac{1}{2}\,\|\boldsymbol{y} - \boldsymbol{\beta}\|_2^2 + \eta_1\,\|\boldsymbol{\beta}\|_1 + \eta_2 \sum_{i<j} w_{ij}|\beta_i - \beta_j| \tag{19}$$

where $\eta_1 = \lambda_1/\rho$ and $\eta_2 = \lambda_2/\rho$. The wFLSA can be solved using an additional ADMM algorithm with complexity in the order of $\mathcal{O}(m^2)$ (see Dijkstra et al. [27]).

#### 2.4.3. Algorithm for the CVN model

Combining the results presented in the previous section, leads us to the following procedure for estimating CVN models. A graphical representation can be found in Figure 4.

1. Compute the empirical covariance matrices $\widetilde{\boldsymbol{\Sigma}}_i = (\sigma_{st}^{(i)})_{p\times p} = n_i^{-1}\sum_{j=1}^{n_i} \boldsymbol{x}_j^{(i)}\left(\boldsymbol{x}_j^{(i)}\right)^\top$ with $i = 1,2,\ldots,m$.
2. Initialize $\boldsymbol{\Theta}_i(1) = \text{diag}(\widetilde{\boldsymbol{\Sigma}}_i)^{-1}$ and $\mathbf{Z}_i(1) = \mathbf{Y}_i = \mathbf{0}$. Alternatively, one can use a warmstart where $\boldsymbol{\Theta}_i(1) = \arg\min_{\boldsymbol{\Theta}_i} \ell_i(\boldsymbol{\Theta}_i) + \lambda_1\,\|\boldsymbol{\Theta}_i\|_1^{\text{off}}$ is the GLASSO estimate for the individual graphs $i = 1,2,\ldots,m$.
3. Update $\boldsymbol{\Theta}$. For each graph $i = 1,2,\ldots,m$, do:
   (a) Compute the eigenvalue decomposition $\mathbf{Q}_i\boldsymbol{\Gamma}_i\mathbf{Q_i}^\top$ of $\widetilde{\boldsymbol{\Sigma}}_i - (\rho/n_i)\,[\mathbf{Z}_i(k) - \mathbf{Y}_i(k)]$;
   (b) Let $\widetilde{\boldsymbol{\Gamma}}_i = \text{diag}(\widetilde{\gamma}_{i1}, \widetilde{\gamma}_{i2}, \ldots, \widetilde{\gamma}_{ip})$ where $\widetilde{\gamma}_{ij} = \frac{n_i}{2\rho}\left(\sqrt{\gamma_{ij}^2 + 4\rho/n_i} - \gamma_{ij}\right)$;
   (c) Then $\boldsymbol{\Theta}_i(k+1) = \mathbf{Q}_i\widetilde{\boldsymbol{\Gamma}}_i\mathbf{Q}_i^\top$.
4. Update $\mathbf{Z}$ by going over all potential edges $\{s,t\}$:
   (a) Let $\boldsymbol{\theta}_{st} = \left(\theta_{st}^{(1)}, \theta_{st}^{(2)}, \ldots, \theta_{st}^{(m)}\right)^\top$, $\boldsymbol{z}_{st} = \left(z_{st}^{(1)}, z_{st}^{(2)}, \ldots, z_{st}^{(m)}\right)^\top$ and $\boldsymbol{y}_{st} = \left(y_{st}^{(1)}, y_{st}^{(2)}, \ldots, y_{st}^{(m)}\right)^\top$;
   (b) Update the diagonal: $\boldsymbol{z}_{ss}(k+1) = \boldsymbol{\theta}_{ss}(k+1) + \boldsymbol{y}_{ss}(k)$ for $s = 1,2,\ldots,p$;
   (c) Update the off-diagonal entries $\boldsymbol{z}_{st}(k+1)$ and $\boldsymbol{z}_{ts}(k+1)$ for all $s < t$. Let $\boldsymbol{\beta} = \boldsymbol{z}_{st}(k+1)$ and $\boldsymbol{y} = \boldsymbol{\theta}_{st}(k+1) + \boldsymbol{y}_{st}(k)$, then

$$\boldsymbol{z}_{st}(k+1) = \underset{\boldsymbol{\beta}\in\mathbb{R}^m}{\arg\min}\ \frac{1}{2}\,\|\boldsymbol{y} - \boldsymbol{\beta}\|_2^2 + \eta_1\,\|\boldsymbol{\beta}\|_1 + \eta_2 \sum_{i<j} w_{ij}|\beta_i - \beta_j|$$

   is the corresponding wFLSA problem, where $\eta_1 = \lambda_1/\rho$ and $\eta_2 = \lambda_2/\rho$. Solve this using the algorithm presented in Dijkstra et al. [27].
5. Update $\mathbf{Y}$: $\mathbf{Y}_i(k+1) = \boldsymbol{\Theta}_i(k+1) - \mathbf{Z}_i(k+1) + \mathbf{Y}_i(k)$ for $i = 1,2,\ldots,m$.

6. In case the stopping criterion $\sum_{i=1}^{m} \|\boldsymbol{\Theta}_i(k+1) - \boldsymbol{\Theta}_i(k)\|_1 \,/\, \sum_{i=1}^{m} \|\boldsymbol{\Theta}_i(k)\|_1 < \epsilon$ has been met, return the estimate of the CVN: $\widehat{\boldsymbol{\Theta}}_{\text{CVN}} = \{\boldsymbol{\Theta}_i(k+1)\}_{i=1}^{m}$. Otherwise, repeat steps 3 to 5.

An implementation of this algorithm is publicly available as an `R` package at https://bips-hb.r-universe.dev/CVN.

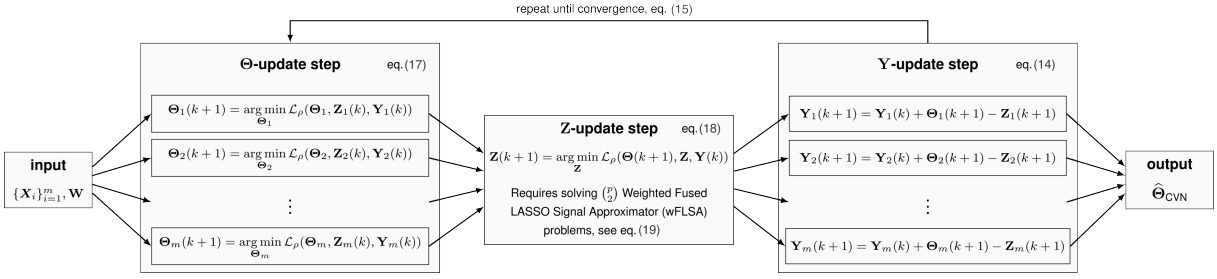



**Figure 4.** A graphical representation of the algorithm for estimating the CVN model, see Equation (5). The input is the raw data set and the weight matrix; the output is the estimated CVN model. The ADMM consists of three updates steps for $\boldsymbol{\Theta}$, $\mathbf{Z}$ and $\mathbf{Y}$ that are repeated till convergence. Update step $\boldsymbol{\Theta}$ and $\mathbf{Y}$ can be subdivided into $m$ independent step. During the $\mathbf{Z}$-update step, $\binom{p}{2}$ wFLSA problems need to be solved, one for each potential edge.

### *2.5. Algorithmic Complexity*

Here, we determine the computational complexity of the CVN algorithm presented in the previous section. We first consider each of the update steps separately and combine the results at the end.

The $\boldsymbol{\Theta}(k+1)$-update step can be decomposed into $m$ distinct steps, each associated with a specific graph $i = 1, 2, \ldots, m$, as illustrated in Figure 4. The most computationally demanding aspect of the $\boldsymbol{\Theta}_i(k+1)$ update step, see Equation (12), involves performing the eigendecomposition of a $(p \times p)$-dimensional matrix, a task known to be on the order of $\mathcal{O}(p^3)$. The entire update step requires, thus, in the order of $\mathcal{O}(mp^3)$ operations.

In the case of the $\mathbf{Y}(k+1)$ update step, see Equation 14, the task entails summing three $(p \times p)$-dimensional matrices for each graph $i = 1, 2, \ldots, m$. This operation, thus, requires $\mathcal{O}(mp^2)$ operations.

The $\mathbf{Z}(k+1)$-update step requires one to solve a wFLSA problem for each of the $\binom{p}{2}$ potential edges in the graph. We know that the complexity of solving a single wFLSA problem is $\mathcal{O}(rm^2)$, where $r$ is the (average) number of iterations needed for the algorithm to converge [27]. The overall complexity of the $\mathbf{Z}(k+1)$-update step is, thus, $\mathcal{O}\left(\binom{p}{2} \cdot rm^2\right) = \mathcal{O}(rm^2p^2)$.

Combining these results, yields the computational complexity of the ADMM algorithm for solving CVN models as

$$\mathcal{O}\Big(R \cdot \big( \underbrace{mp^3}_{\boldsymbol{\Theta}(k+1)\text{-update}} + \underbrace{mp^2}_{\mathbf{Y}(k+1)\text{-update}} + \underbrace{rm^2p^2}_{\mathbf{Z}(k+1)\text{-update}} \big)\Big) = \mathcal{O}\left(Rrm^2p^3\right),$$

where $R$ is the number of iterations needed for the ADMM to satisfy the stopping criterion in Equation (15), $r$ denotes the average iterations needed for the wFLSA algorithm to converge [27], $m$ is the number of graphs, and $p$ is the number of variables.

### *2.6. Interpolation*

In this section, we propose a method for interpolating a graph for which there are no observations based on an estimated CVN model with $m$ graphs. We denote the graph we want to interpolate by $G_{m+1}$. We are concerned solely with the existence of edges and not values of the entries in the precision matrix $\boldsymbol{\Theta}_{m+1}$. Approaches in the literature often rely on the so-called 'AND' and 'OR' rules (see, for example, Hallac et al. [12]). In case of the former, the interpolated graph has the edge $\{s, t\}$ if and only if *all* $m$ graphs in the CVN model have the edge $\{s, t\}$. In case of the 'OR' rule, the interpolated graph has the edge $\{s, t\}$ if at least one of the other estimated graphs has that edge. Note that with this approach, it is difficult to control the level of sparsity and take different levels of similarity between the graphs into account.

We interpolate graph $G_{m+1}$ given the estimated CVN model $\widehat{\boldsymbol{\Theta}}_{\text{CVN}}$ by solving

$$\widehat{\boldsymbol{\Theta}}_{m+1} = \underset{\boldsymbol{\Theta}_{m+1} \succ 0}{\arg\min}\, \ell_{m+1}(\boldsymbol{\Theta}_{m+1}) + \lambda_1 \|\boldsymbol{\Theta}_{m+1}\|_1^{\text{off}} + \lambda_2 \sum_{i=1}^{m} \omega_i \left\| \widehat{\boldsymbol{\Theta}}_i - \boldsymbol{\Theta}_{m+1} \right\|_1^{\text{off}}$$

where $\boldsymbol{\omega} = (\omega_1, \omega_2, \ldots, \omega_m)^\top$ are the *smoothing coefficients* which reflect the level of similarity between the individual graphs of the CVN and the interpolated graph. Note that, since there are no observations for the graph

$G_{m+1}$, the corresponding log-likelihood is equal to $\ell_{m+1}(\boldsymbol{\Theta}_{m+1}) = 0$, see Equation (4). Similar to the **Z**-update step, we can divide the problem into $\binom{p}{2}$ separate optimization problems, one for each potential edge $\{s,t\}$. In fact, we can write the function as

$$\begin{aligned}\widehat{\boldsymbol{\Theta}}_{m+1} &= \underset{\boldsymbol{\Theta}_{m+1}\succ 0}{\arg\min}\ \lambda_1 \sum_{s,t} |\theta_{st}^{(m+1)}| + \lambda_2 \sum_{i=1}^{m} \omega_i \sum_{s,t} |\widehat{\theta}_{st}^{(i)} - \theta_{st}^{(m+1)}| \\ &= \underset{\boldsymbol{\Theta}_{m+1}\succ 0}{\arg\min} \sum_{s,t} \left[ \lambda_1 |\theta_{st}^{(m+1)}| + \lambda_2 \sum_{i=1}^{m} w_i |\widehat{\theta}_{st}^{(i)} - \theta_{st}^{(m+1)}| \right].\end{aligned}$$

We, thus, need to solve for each potential edge $\{s,t\}$:

$$\widehat{\theta}_{st}^{(m+1)} = \underset{\theta_{st}^{(m+1)}\in\mathbb{R}}{\arg\min}\ \lambda_1 |\theta_{st}^{(m+1)}| + \lambda_2 \|\boldsymbol{\omega}^\top \widehat{\boldsymbol{\theta}}_{st} - \theta_{st}^{(m+1)}\|_1.$$

This function is convex (see Appendix A) and can be solved using any derivative-free search method. In this context, we employ Brent's algorithm [28]. The sign of $\widehat{\theta}_{st}^{(m+1)}$ is inconsequential, as our primary objective is to establish the presence or absence of the edge $\{s,t\}$ in the interpolated graph, i.e., $\widehat{\theta}_{st}^{(m+1)} \neq 0$. Note that the smoothing coefficients $\omega_1, \omega_2, \ldots, \omega_m$ need to be chosen a priori, which requires a certain level of prior knowledge.

### *2.7. Tuning Parameter Selection*

After fitting the CVN model, the next step involves selecting suitable values for the tuning parameters: $\lambda_1$, which governs the sparsity of the graphs, and $\lambda_2$, responsible for regulating the smoothness or similarity between the graphs. Since having prior knowledge about appropriate values for the tuning parameters is rare, one often opts for a data-driven approach. Numerous methods exist for tuning parameter selection, and in this context, we explore two frequently used measures: the Akaike Information Criterion (AIC) and Bayesian Information Criterion (BIC). However, it is important to note that in our scenario, these criteria can only be approximated [7].

Let $\widehat{\boldsymbol{\Theta}}_i(\lambda_1, \lambda_2)$ denote the estimator of the precision matrix for the $i$-th graph for specific values of $\lambda_1$ and $\lambda_2$. The AIC can be approximated as

$$\text{AIC}(\lambda_1, \lambda_2) = \sum_{i=1}^{m} \ell_i(\widehat{\boldsymbol{\Theta}}_i(\lambda_1, \lambda_2)) + 2\|\widehat{\boldsymbol{\Theta}}_i(\lambda_1, \lambda_2)\|_0, \tag{20}$$

where $\|\cdot\|_0$ denotes the $L_0$-norm, equivalent to the number of non-zero entries in the matrix [7]. On the other hand, the BIC, taking into account the number of observations for the graphs, tends to result in sparser models than the AIC:

$$\text{BIC}(\lambda_1, \lambda_2) = \sum_{i=1}^{m} \ell_i(\widehat{\boldsymbol{\Theta}}_i(\lambda_1, \lambda_2)) + 2\log(n_i)\|\widehat{\boldsymbol{\Theta}}_i(\lambda_1, \lambda_2)\|_0, \tag{21}$$

where $n_i$ is the number of observations for the $i$-th graph. The optimal values for $\lambda_1$ and $\lambda_2$ are those that minimize either the approximated AIC or BIC. In our simulation study, see Section 2.8, we consider both information criteria. Note that while AIC and BIC are commonly used, they are not the sole metrics for selecting the optimal tuning parameters based on the data. Other methods are cross-validation, bootstrap [29], and alternative information criteria like the extended BIC [30] and Predictive Information Criteria [31]. Here, we opt for AIC and BIC due to the computational efficiency, which is advantageous for a computationally intensive algorithm such as the ADMM propsed here.

### *2.8. Simulation Study*

In this simulation study, we consider a CVN model with two external covariates, $U_1$ and $U_2$, each of which takes values from the set $\{1,2,3\}$ featuring nine graphs. The graph corresponding to $U_1 = i$ and $U_2 = j$ is denoted as $G_{ij} = G(U_1 = i, U_2 = j)$, and its associated adjacency, precision, and covariance matrices are similarly indexed as $\mathbf{A}_{ij}$, $\boldsymbol{\Theta}_{ij}$ and $\boldsymbol{\Sigma}_{ij}$, respectively. Our simulation setup allows for varying the number of observations ($n$), the number of variables ($p$) and the the general graph structure, specifically Erdös-Rényi and Barabasi-Albert graphs [1] in our case. Additionally, we can control the extent to which the edges in the graph change with $U_1$ and $U_2$. Adjusting $n$ and $p$ is straightforward, as we will see later. Our focus is first on how to employ a specific general graph structure and how to regulate the extent with which the graphs change with changes in the two external covariates.

We can represent our case as a grid of graphs, as illustrated in Figure 5, with $U_1$ and $U_2$ positioned along the horizontal and vertical axes, respectively. The process how the various graphs were generated will be explained later. Graph $G_{11}$ is located in the bottom-left corner, followed by graph $G_{21}$ to its right, and so forth, concluding with graph $G_{33}$ in the upper-right corner. We designate graph $G_{11}$ as the *starting graph*; it is randomly generated according to a specific graph model. In this context, we consider two types:

1. The Erdös-Rényi (ER) graph, which has a single parameter $\pi$, representing the probability of the existence of any given edge. The expected density of the graph, defined as the number of edges divided by the total number of possible edges (i.e., $p(p-1)/2$) [1], is expressed as

$$\mathbb{E}_{\text{ER}}\left[|E|\right] = \pi.$$

   Here, $|E|$ denotes the cardinality of the edge set, and $\mathbb{E}_{\text{ER}}$ signifies the expectation under the ER model. The relevance of the expected density of the graph becomes apparent later.
2. The Barabasi-Albert (BA) graph, which is a scale-free graph. The degrees of the graph follow a power law distribution with an exponent of 3 [1]. The expected density [32] of this graph type is approximately given by

$$\mathbb{E}_{\text{BA}}\left[|E|\right] \approx \frac{p-1}{2}, \tag{22}$$

   where $\mathbb{E}_{\text{BA}}$ represents the expectation under the BA model.

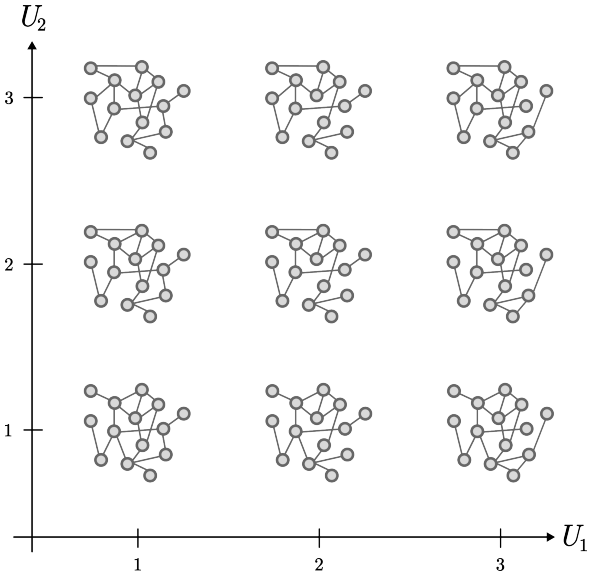



**Figure 5.** Illustration of the CNV model with two external covariates, $U_1$ and $U_2$, as used in the simulation study. The values of $U_1$ and $U_2$ are shown on the horizontal and vertical axes, respectively. The 'starting graph' $G_{11}$ is located in the lower-left corner, followed by graph $G_{21}$ to its right, and so on, ending with graph $G_{33}$ is in the upper-right corner. Figure 6 exemplify how these graphs were created.

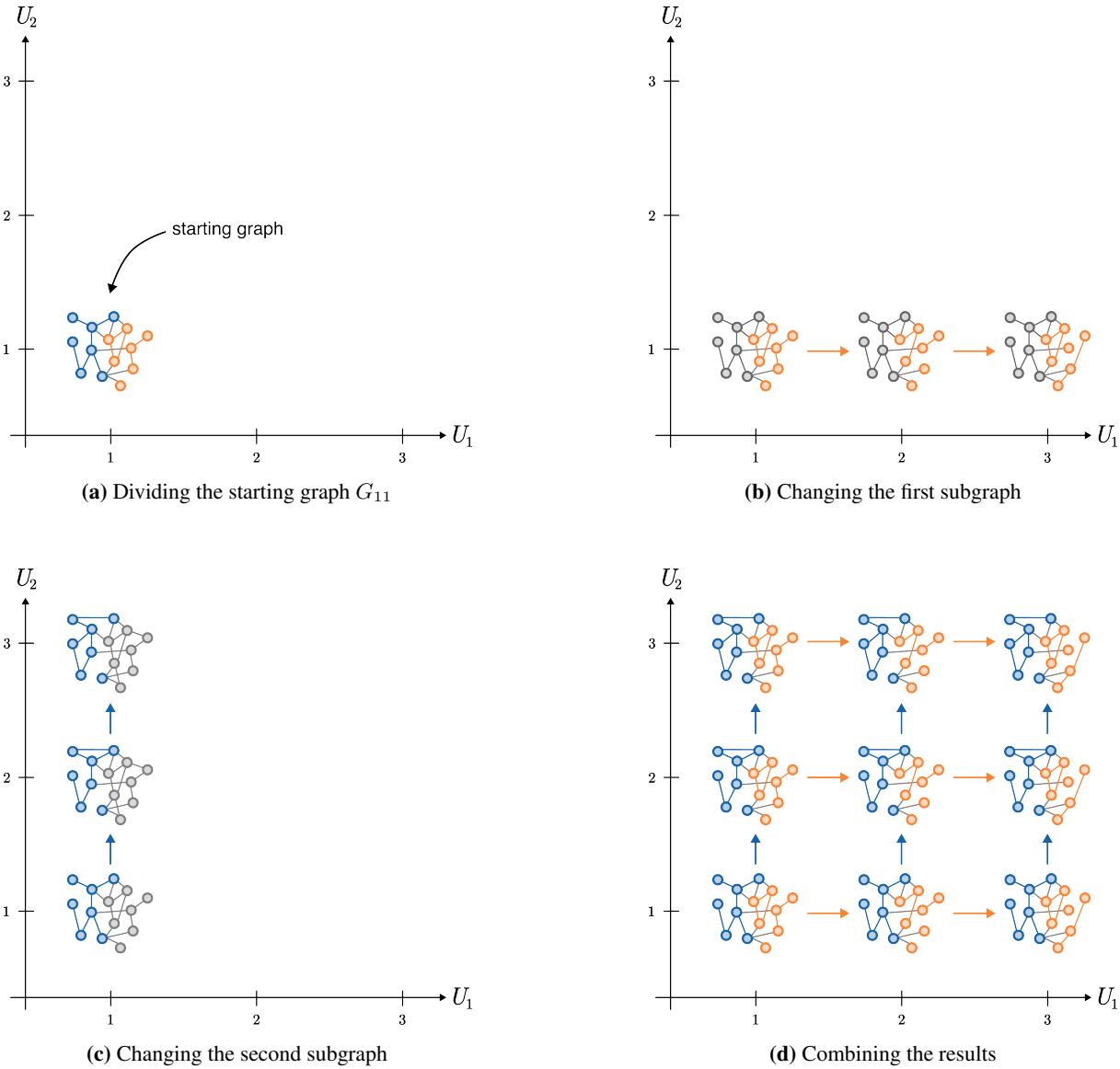
As previously mentioned, we aim to control the extent to which graphs change with $U_1$ and $U_2$. To achieve this, we partition the starting graph into two equal-sized subgraphs. See, for an example, Figure 6a, where the first subgraph is shown in orange and the second in blue. In cases where the number of variables $p$ is odd, the first and second subgraphs comprise $\lceil p/2 \rceil$ and $\lfloor p/2 \rfloor$ nodes, respectively, where $\lceil \cdot \rceil$ and $\lfloor \cdot \rfloor$ denote the ceiling and floor functions. The adjacency matrix of the starting graph can be expressed in terms of these subgraphs as follows:

$$\mathbf{A}_{11} = \begin{bmatrix} \mathbf{A}_1^1 & \mathbf{A} \\ \mathbf{A}^\top & \mathbf{A}_1^2 \end{bmatrix}. \tag{23}$$

Here, $\mathbf{A}_1^1$ represents the adjacency matrix for the first subgraph, and $\mathbf{A}_1^2$ is the matrix for the second subgraph. Similarly, we divide the remaining eight graphs into subgraphs using the same structure:

$$\mathbf{A}_{ij} = \begin{bmatrix} \mathbf{A}_i^1 & \mathbf{A} \\ \mathbf{A}^\top & \mathbf{A}_j^2 \end{bmatrix}$$

for $i = 1, 2, 3$ and $j = 1, 2, 3$.




**(a)** Dividing the starting graph $G_{11}$

**(b)** Changing the first subgraph

**(c)** Changing the second subgraph

**(d)** Combining the results

**Figure 6.** An example illustrating the generation of the nine graphs used in the simulation study. (**a**) The starting graph is divided into two subgraphs (orange and blue). (**b**) The edges in the orange subgraph change with respect to $U_1$. (**c**) The edges in the blue subgraph change with respect to $U_2$. (**d**) Combining both modifications in the orange and blue subgraphs yields the nine graphs. Note that the edges between the two subgraphs (depicted in gray) remain unaltered.

When varying $U_1$, only the subgraph corresponding to $\mathbf{A}_i^1$ undergoes changes, while the remainder of the graph remains unaltered. Similarly, when varying $U_2$, only the matrix $\mathbf{A}_j^2$ undergoes changes. It is important to note that the edges between the two subgraphs, represented here as $\mathbf{A}$, remain constant throughout and form the *core* of the network.

After sampling the starting graph $G_{11}$ and obtaining $\mathbf{A}_1^1$ and $\mathbf{A}_1^2$, we derive $\mathbf{A}_2^1$ by randomly adding and removing $b_1$ edges, where $b_1$ denotes the number of edges that are modified at each step along the $U_1$ direction. If there are up to $b_1$ edges in the subgraph, all existing edges change. Subsequently, we obtain $\mathbf{A}_3^1$ by altering $b_1$ edges in $\mathbf{A}_2^1$. A similar process is followed for the second subgraph: we obtain $\mathbf{A}_2^2$ by randomly adding and removing $b_2$ edges in the matrix $\mathbf{A}_1^2$. The matrix $\mathbf{A}_3^2$ is then created by modifying $b_2$ edges in the matrix $\mathbf{A}_2^2$.

We visually depict this process in Figure 6. Figure 6a displays the starting graph $G_{11}$ with the first subgraph depicted in orange and the second in blue. Changes occur exclusively in the edges of the orange subgraph as $U_1$ varies, as illustrated in Figure 6b. Similarly, only edges in the blue subgraph undergo changes with variations in $U_2$, as shown in Figure 6c. Combining the changes made in the orange and blue subgraphs results in the nine graphs depicted in Figure 6d.

Focusing on the number of edges $b_1$ and $b_2$ that change with varying either $U_1$ or $U_2$ may be challenging to interpret, particularly for different values of $p$ or the number of edges. Therefore, we find it more meaningful to

think in terms of the percentage of edges in the graph that change with $U_1$ and $U_2$. These percentages are denoted as $\pi_1$ and $\pi_2$. The number of edges changing with $U_1$ and $U_2$ are then given by

$$b_1 = \text{round}\left(\pi_1 \mathbb{E}[|E|]\right) \qquad \text{and} \qquad b_2 = \text{round}\left(\pi_2 \mathbb{E}[|E|]\right), \tag{24}$$

where round$(\cdot)$ rounds to the nearest integer. In other words, $\pi_1$ and $\pi_2$ represent the percentage of edges the graph is expected to have ($\mathbb{E}[|E|]$) that change with $U_1$ and $U_2$, respectively.

We translate the obtained adjacency matrices into nine precision matrices. Subsequently, these precision matrices are used to generate data using a normal distribution. We adopt a methodology similar to the one outlined in Liu and Wang [33]. We define $\widetilde{\mathbf{\Theta}}_{ij} = v\mathbf{A}_{ij}$ as the adjacency matrix for $U_1 = i$ and $U_2 = j$, multiplied by the scalar $v > 0$. Let $\kappa_{ij}$ represent the smallest eigenvalue of the matrix $\widetilde{\mathbf{\Theta}}_{ij}$. To ensure the precision matrix is positive definite, we 'increase' the diagonal, specifically,

$$\mathbf{\Theta}_{ij} = \widetilde{\mathbf{\Theta}}_{ij} + \text{diag}\left(|\kappa_{ij}| + 0.1 + u\right), \tag{25}$$

where $u > 0$ is a scalar as well. Throughout our simulation, we maintain $v = 0.4$ and $u = 0.1$, consistent with the values employed by Liu and Wang [33]. The corresponding covariance matrix is simply the inverse of the precision matrix, i.e., $\mathbf{\Sigma}_{ij} = \mathbf{\Theta}_{ij}^{-1}$, and is determined numerically. The data for each graph, $\mathbf{X}_{ij}$, consists of $n$ independent samples from the $p$-dimensional normal distribution $\mathcal{N}(\mathbf{0}, \mathbf{\Sigma}_{ij})$.

In conclusion, our simulator follows the steps:

1. Specify the number of observations ($n$), parameters ($p$), initial graph type (ER or BA model), and the percentage of edges in the graphs changing in the $U_1$ and $U_2$ directions ($\pi_1$ and $\pi_2$);
2. Determine the number of changing edges $b_1$ and $b_2$ based on the percentages $\pi_1$ and $\pi_2$ using Equation (24);
3. Generate the starting graph $G_{11}$ with $p$ vertices following the selected model from step 1.
4. Divide the starting graph $G_{11}$ randomly into two subgraphs, see Equation (23) and Figure 6a;
5. Generate the adjacency matrix $A_2^1$ by randomly adding and removing $b_1$ edges from $A_1^1$, see Figure 6b;
6. Generate $A_3^1$ by adding and removing $b_1$ edges from $A_2^1$ obtained in the previous step;
7. Repeat the last two steps for the adjacency matrices $A_2^2$ and $A_3^2$ by changing $b_2$ edges, see Figure 6c;
8. Steps 3–7 result in nine adjacency matrices, see Figure 6d. Determine the corresponding nine precision matrices using Equation (25);
9. Numerically determine the covariance matrices (inverse of the precision matrices) from the previous step, and
10. Generate nine data sets $\mathbf{X}_{ij}$, where $\mathbf{X}_{ij}$ comprises $n$ independent draws from the $p$-dimensional normal distribution $\mathcal{N}(\mathbf{0}, \mathbf{\Sigma}_{ij})$.

#### 2.8.1. Simulation Setup

In our simulation study, we explore various parameter settings outlined in Table 1. We set the number of variables, $p$, to 100 and vary the number of observations, $n$, from 100 to 200. Both scenarios are high-dimensional, as the number of edges in each graph is $\binom{p}{2} = 4{,}950$. As previously mentioned, we consider both the ER and BA models. For the ER model, the density $\pi$ is set to $10\%$. The sparsity of the BA model is constant, as given in Equation (22). We partition the starting graph into two subgraphs with a probability of $0.5$ and introduce variability in the degree to which graphs change in the $U_1$ and $U_2$ directions by setting $(\pi_1, \pi_2)$ to $(0.1, 0.1)$, $(0.1, 0.2)$, and $(0.2, 0.2)$.

**Table 1.** Parameter settings for the simulation study

| Description | Parameter | Value |
|---|---|---|
| Number of variables | $p$ | 100 |
| Number of observations | $n$ | 100, 200 |
| Density ER graph | $\pi$ | 0.1 |
| Percentage edges changing with $U_1$ | $\pi_1$ | 0.1, 0.2 |
| Percentage edges changing with $U_2$ | $\pi_2$ | 0.1, 0.2 |

Using the CVN algorithm necessitates the specification of $\lambda_1$ and $\lambda_2$ along with the weight matrix $\mathbf{W}$. To improve interpretability and tuning stability, we reparameterize the penalties as follows:

$$\gamma_1 = 2\frac{\lambda_1}{mp(p-1)} \qquad \text{and} \qquad \gamma_2 = 4\frac{\lambda_2}{m(m-1)p(p-1)}.$$

While $\lambda_1$, $\lambda_2$ act on global sums of penalties, $\gamma_1$, $\gamma_2$ directly control the shrinkage of individual edges and edge pairs, making the penalty strength invariant to the number of potential edges. This distinction only becomes relevant when the number of nodes, $p$ or graphs, $m$ varies; since our simulation study uses a fixed graph size, both parameterizations are equivalent in practice (see Appendix B). We nonetheless adopt $\gamma_1$ and $\gamma_2$ for clarity and consistency with the proposed formulation. In our simulation, we opt for $(\gamma_1, \gamma_2) \in \Gamma \times \Gamma$, where $\Gamma = \{10^{-5}, 5 \cdot 10^{-5}, 10^{-4}, 5 \cdot 10^{-4}, 10^{-3}, 5 \cdot 10^{-3}\}$.

We explore three distinct weight matrices:

1. $\mathbf{W}_0 = \mathbf{0}_{m\times m}$, the zero matrix. This corresponds to applying the GLASSO to each of the nine graphs individually, i.e., there is no smoothing;
2. $\mathbf{W}_{\text{grid}}$, defined as

$$\mathbf{W}_{\text{grid}} = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix} \tag{26}$$

   representing the scenario where each graph is smoothed with its 'adjacent' graphs, and
3. $\mathbf{W}_{\text{full}} = \mathbf{1}_{m\times m} - \text{diag}(\mathbf{1})$, a matrix of ones with zeros on its diagonal. In this case, each graph is equally smoothed with every other graph.

See Figure 7 for a visual representation of the meta-graphs implied by these three weight matrices. Given the simulation process for the nine graphs, the second weight matrix, $\mathbf{W}_{\text{grid}}$, mirrors the true underlying structure closest. We repeat the simulation for each simulation setting and weight matrix 20 times.

### 2.8.2. Performance Measures

As performance measure, we consider the $F_1$-score which we define here. Recall from the previous section that the true adjacency matrix is denoted as $\mathbf{A}_{ij} = \left(a_{st}^{(ij)}\right)_{p\times p}$, and let $\widehat{\mathbf{A}}_{ij}(\gamma_1, \gamma_2) = \left(\widehat{a}_{st}^{(ij)}(\gamma_1, \gamma_2)\right)_{p\times p}$ be the estimated adjacency matrix for $U_1 = i$, $U_2 = j$ and tuning parameters $\gamma_1$ and $\gamma_2$. The number of true positives (TP), false positives (FP) and false negatives (FN) are

$$\begin{aligned} \text{TP}(\gamma_1, \gamma_2) &= \sum_{i=1}^{3}\sum_{j=1}^{3}\left(\sum_{s<t} a_{st}^{(ij)} \cdot \widehat{a}_{st}^{(ij)}(\gamma_1, \gamma_2)\right), \\ \text{FP}(\gamma_1, \gamma_2) &= \sum_{i=1}^{3}\sum_{j=1}^{3}\left(\sum_{s<t} \left(1 - a_{st}^{(ij)}\right) \cdot \widehat{a}_{st}^{(ij)}(\gamma_1, \gamma_2)\right) \quad \text{and} \\ \text{FN}(\gamma_1, \gamma_2) &= \sum_{i=1}^{3}\sum_{j=1}^{3}\left(\sum_{s<t} a_{st}^{(ij)} \cdot \left(1 - \widehat{a}_{st}^{(ij)}(\gamma_1, \gamma_2)\right)\right). \end{aligned}$$

Note that one needs to sum over all nine graphs. Using these definitions, we can express precision as follows:

$$\text{Precision}\,(\gamma_1, \gamma_2) = \frac{\text{TP}\,(\gamma_1, \gamma_2)}{\text{TP}\,(\gamma_1, \gamma_2) + \text{FP}\,(\gamma_1, \gamma_2)}.$$

And recall as:

$$\text{Recall}\,(\gamma_1, \gamma_2) = \frac{\text{TP}\,(\gamma_1, \gamma_2)}{\text{TP}\,(\gamma_1, \gamma_2) + \text{FN}\,(\gamma_1, \gamma_2)}.$$

The $F_1$-score is then simply the harmonic mean of the precision and recall:

$$F_1\,(\gamma_1, \gamma_2) = 2 \cdot \frac{\text{Precision}\,(\gamma_1, \gamma_2) \cdot \text{Recall}\,(\gamma_1, \gamma_2)}{\text{Precision}\,(\gamma_1, \gamma_2) + \text{Recall}\,(\gamma_1, \gamma_2)}.$$

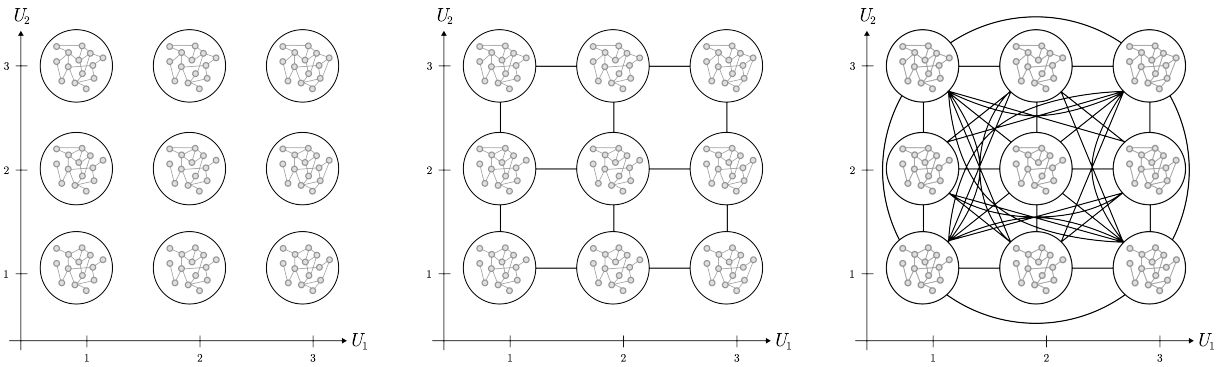
The $F_1$-score requires selecting the tuning parameters $\gamma_1$ and $\gamma_2$ (or equivalently, $\lambda_1$ and $\lambda_2$). We consider all values of $(\gamma_1, \gamma_2) \in \Gamma \times \Gamma$. If the selection is based on the AIC or the BIC, see Equations (20) and (21), respectively, the $F_1$-scores are given by

$$F_1^{\text{AIC}} = \min_{(\gamma_1,\gamma_2)\in\Gamma\times\Gamma} \text{AIC}\,(\gamma_1, \gamma_2) \quad \text{and} \quad F_1^{\text{BIC}} = \min_{(\gamma_1,\gamma_2)\in\Gamma\times\Gamma} \text{BIC}\,(\gamma_1, \gamma_2)\,. \tag{27}$$

i.e., the scores obtained for the $\gamma_1$ and $\gamma_2$ values that minimize the AIC or the BIC. Additionally, we consider the oracle $F_1$-score, defined as the highest $F_1$-score achievable by the CVN algorithm when the true underlying graphs are known. This value does not necessarily equal 1, as it is constrained by the model assumptions, regularization, and finite sample size, which may prevent perfect recovery even under optimal tuning. This score is given by

$$F_1^{\text{oracle}} = \max_{(\gamma_1,\gamma_2)\in\Gamma\times\Gamma} F_1\,(\gamma_1, \gamma_2)\,. \tag{28}$$

The oracle score serves as a benchmark to determine the best possible performance achievable when selecting the optimal tuning parameters.




**(a)** No smoothing, $\mathbf{W}_0$ **(b)** Grid, $\mathbf{W}_{\text{grid}}$ **(c)** Fully connected, $\mathbf{W}_{\text{full}}$

**Figure 7.** A visualization of the three meta-graphs corresponding to the three weight matrices considered in the simulation study.

### *2.9. Real Data Analysis*

High doses of ionizing radiation (2 Gray [Gy]) used in childhood cancer treatment are known risk factors for acute myeloid leukemia in children and for second primary neoplasms later in life. However, the effects of low doses of radiation ($\leq$0.05 Gy), typically used in medical diagnostics, on the incidence of childhood cancer are less understood.

Our case study uses data from the German 'Cancer in childhood and molecular-epidemiology' study (KiKme), which was collected between 2013 and 2019 from 591 former childhood cancer patients registered in the German Childhood Cancer Registry and cancer-free controls [19]. Within a sub-sample of carefully matched 156 participants, a radiation experiment was conducted where individual reactions of long non-coding ribonucleic acids (lncRNAs) to different radiation exposures in normal somatic cells of the same person were investigated. Primary fibroblasts were donated from long-term childhood cancer survivors who had a first primary cancer during childhood (N1, $n = 52$), with at least one second primary neoplasm (N2+, $n = 52$), and tumor-free controls (N0, $n = 52$) (see Grandt et al. [34]). Cultured fibroblasts of each donor were exposed to 0 Gy (sham-irradiation), 0.05 Gy, and 2 Gy X-rays to investigate cellular reaction to ionizing radiation.

We used $p = 191$ genes that are annotated with the GO-term 'response to ionizing radiation' (GO:0010212). All gene expressions were standardized with mean 0 and a standard deviation of 1 within each class. We estimated all nine graphs based on the various combinations of the levels of the external covariates *group* (N0, N1, N2+) and radiation (0, 0.05 and 2 Gy).

The tuning parameter to control the sparsity level was set to $\gamma_1 = 10^{-5}$ to achieve very sparse networks and $\gamma_2 = 10^{-6}$ to enforce smoothness. We assumed a regular grid $\mathbf{W}_{\text{grid}}$, see Equation (26), and a grid with weights accordingly to the radiation doses:

$$\mathbf{W}_{\text{rad}} = \begin{bmatrix} 0 & 0.025 & 0 & 0.5 & 0 & 0 & 0 & 0 & 0 \\ 0.025 & 0 & 0.975 & 0 & 0.5 & 0 & 0 & 0 & 0 \\ 0 & 0.975 & 0 & 0 & 0 & 0.5 & 0 & 0 & 0 \\ 0.5 & 0 & 0 & 0 & 0.025 & 0 & 0.5 & 0 & 0 \\ 0 & 0.5 & 0 & 0.025 & 0 & 0.975 & 0 & 0.5 & 0 \\ 0 & 0 & 0.5 & 0 & 0.975 & 0 & 0 & 0 & 0.5 \\ 0 & 0 & 0 & 0.5 & 0 & 0 & 0 & 0.025 & 0 \\ 0 & 0 & 0 & 0 & 0.5 & 0 & 0.025 & 0 & 0.975 \\ 0 & 0 & 0 & 0 & 0 & 0.5 & 0 & 0.975 & 0 \end{bmatrix}$$

Readers interested in the second case study, an influenza vaccination study, are referred to Appendix D.

## 3. Results

In this section, we present the results from both the simulation and the case study.

### *3.1. Simulation Results*

We consider 12 different parameter settings and repeat the simulation 20 times for each setting. We evaluate the $F_1$-score using the AIC and BIC criteria for selecting the tuning parameters $\gamma_1$ and $\gamma_2$, see Equation (27). Additionally, we assess the performance in an 'oracle' scenario where the true graphs are known, see Equation (28).

Figures 8–10 show results from the simulation study. These plots share a common structure: the top row displays results for the ER graphs, while the bottom row shows results for the BA graphs. Additionally, the left column corresponds to the setting with $n = 200$ observations, and the right column corresponds to the setting with $n = 100$ observations.

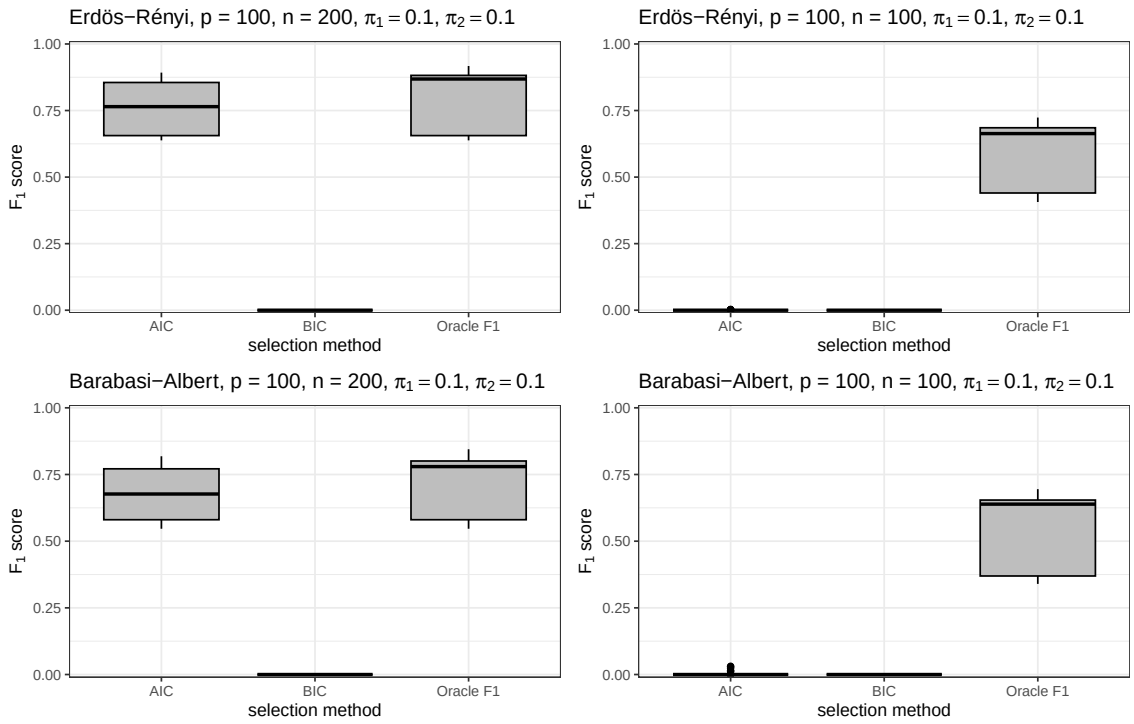



**Figure 8.** The $F_1$-scores when the tuning parameters are selected based on the AIC, BIC, or an oracle. The top row shows the results for the ER graphs, while the bottom row shows the results for the BA graphs. The left column corresponds to $n = 100$ observations, and the right column corresponds to $n = 200$ observations. The parameters $\pi_1 = \pi_2 = 0.1$ for all plots.

Figure 8 shows the $F_1$-scores obtained using three different methods for selecting the 'optimal' tuning parameters: AIC, BIC, and the oracle. These results pertain to the setting where $\pi_1 = \pi_2 = 0.1$; similar patterns are observed for other values of $\pi_1$ and $\pi_2$. The AIC performs reasonably well when $n = 200$, but extremely poorly when $n = 100$. The BIC performs poorly in both cases, since it tends to be too stringent, resulting in very sparse graphs. However, if the optimal values were known, one could achieve rather good $F_1$-scores. This is shown by the oracle which performs best since it considers the true underlying structure of the CVN. This at least shows that,

in principle, the CVN can be estimated rather well. Due to the poor performance of both information criteria, we consider the oracle $F_1$-score for the rest of this section.

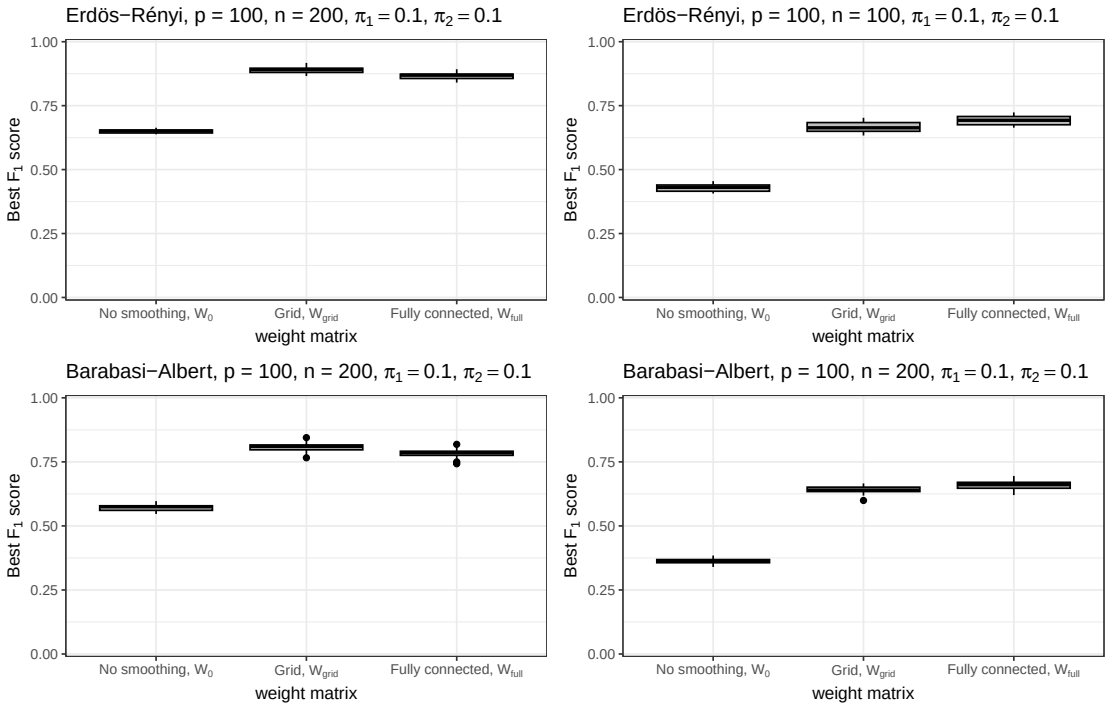



**Figure 9.** The $F_1$-scores when the tuning parameters $\gamma_1$ and $\gamma_2$ are selected using an oracle, with the CVN model using either $\mathbf{W}_0$ (no smoothing), $\mathbf{W}_{\text{grid}}$, or $\mathbf{W}_{\text{full}}$ as the weight matrix. The top row displays results for the ER graphs, and the bottom row shows results for the BA graphs. The left column corresponds to $n = 100$ observations, and the right column corresponds to $n = 200$ observations. The parameters $\pi_1 = \pi_2 = 0.1$ for all plots.

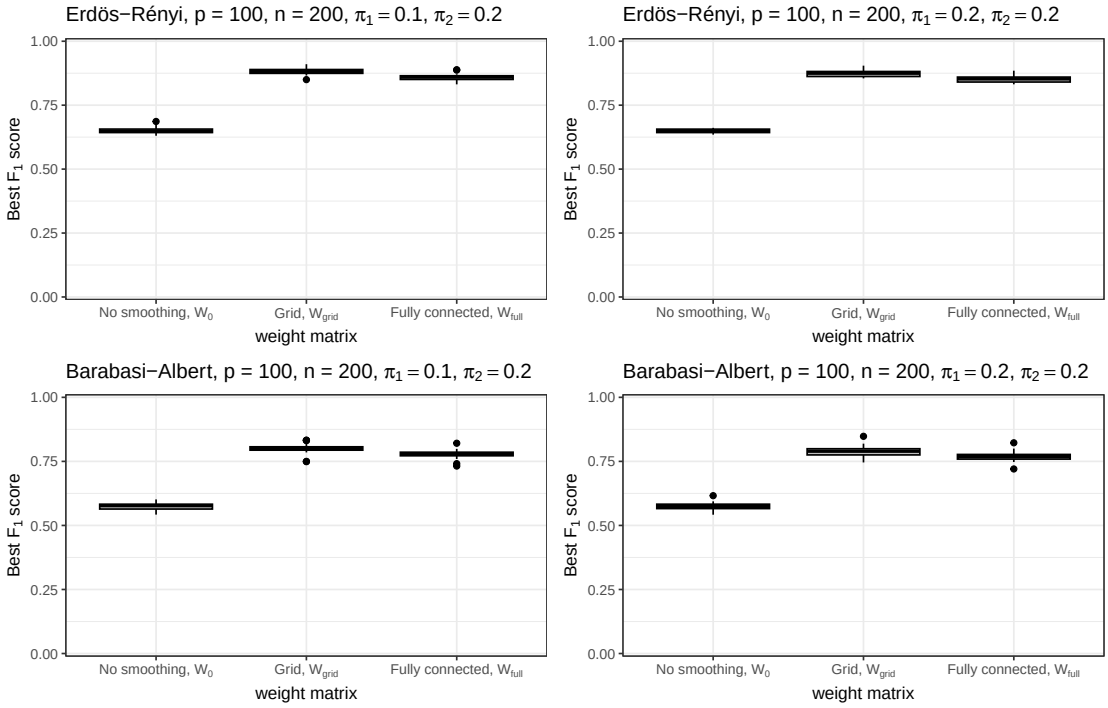



**Figure 10.** The $F_1$-scores when the tuning parameters $\gamma_1$ and $\gamma_2$ are selected using an oracle, with the CVN model using either $\mathbf{W}_0$ (no smoothing), $\mathbf{W}_{\text{grid}}$, or $\mathbf{W}_{\text{full}}$ as the weight matrix. The top row displays results for the ER graphs, and the bottom row shows results for the BA graphs. The number of observations is $n = 200$ for all plots. In the left column, $\pi_1 = 0.1$ and $\pi_2 = 0.2$, while in the right column, $\pi_1 = 0.2$ and $\pi_2 = 0.2$.

Figure 9 shows the best $F_1$-score achievable by selecting the 'optimal' $\gamma_1$ and $\gamma_2$ for different weight matrices. Recall that, as described in Section 2.8, the grid weight matrix, $\mathbf{W}_{\text{grid}}$, aligns best with the data-generating process.

As expected, using the grid weight matrix yields the best results when $n = 200$. However, with fewer observations (right column), the fully connected weight matrix, $\mathbf{W}_{\text{full}}$, performs slightly better. This might be due to the fact that smoothing all nine graphs simultaneously increases the 'evidence' for the existence of edges.

Figure 10 shows the best $F_1$-score when the extent to which the graphs change in the $U_1$ and $U_2$ directions varies. Whether 10% or 20% of the expected number of edges change in these directions has little impact on the algorithm's performance. This pattern is consistent across other simulation settings as well.

All the simulation results can be found and explored interactively at cvn.bips.eu.

### *3.2. Real Data Results*

Figure 11 shows the nine estimated graphs based on the KiKme dataset with $\gamma_1 = 5 \times 10^{-5}$ and $\gamma_2 = 5 \times 10^{-6}$ and the weight matrix $\mathbf{W}_{\text{rad}}$. The nodes, corresponding to the $p = 191$ gene expressions, are positioned on a circle, with lines indicating the edges. Red lines denote 42 edges present in *all* nine graphs, representing the 'core graph' unaffected by external covariates, while blue lines indicate edges present in only a subset of the graphs, i.e., the edges that are influenced by the external covariates.

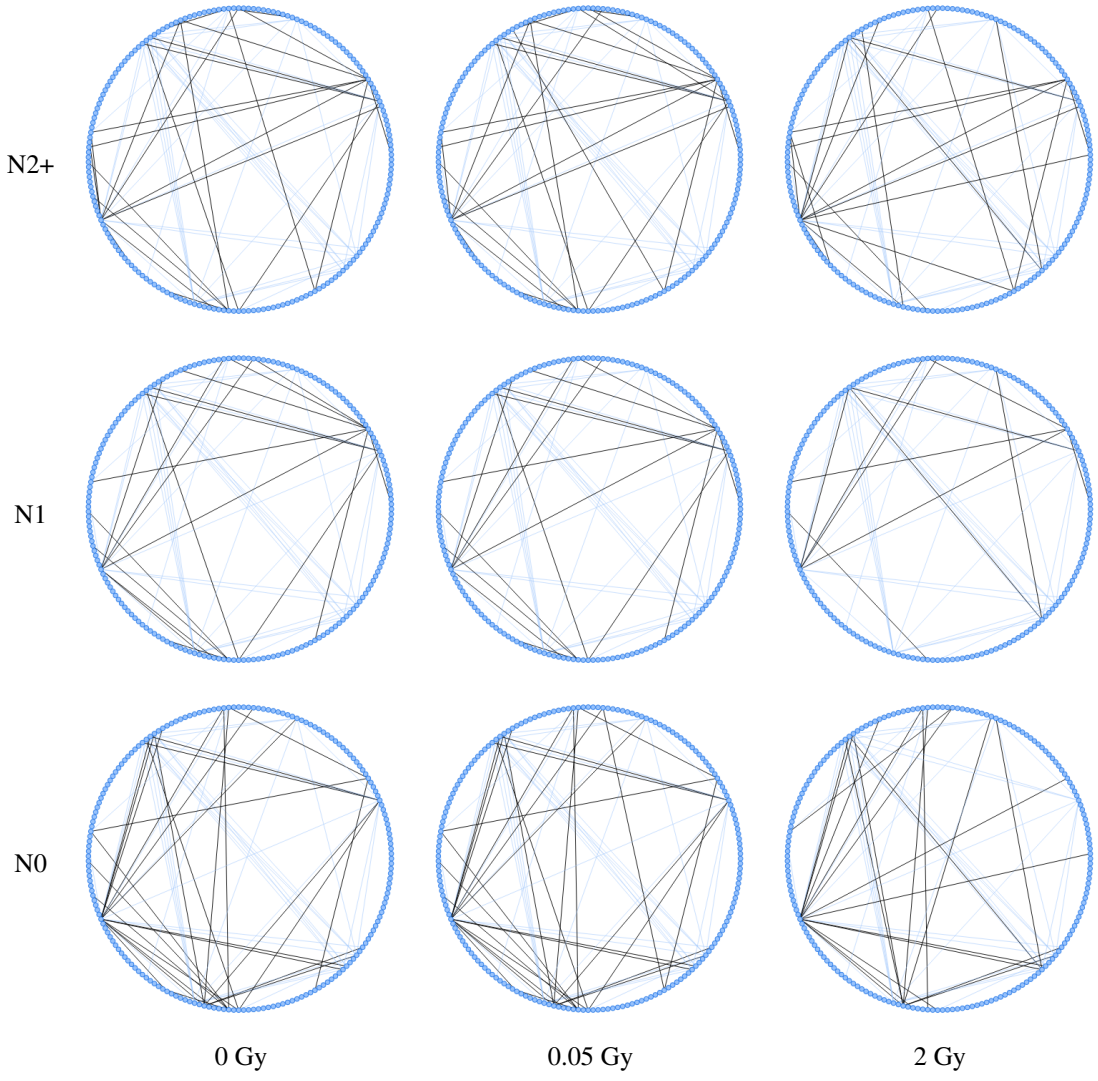



**Figure 11.** Comparison of graphs by group (rows) and radiation dose (columns). Estimated graphs based on $\mathbf{W}_{\text{rad}}$, $\gamma_1 = 5 \times 10^{-5}$ and $\gamma_2 = 5 \times 10^{-6}$. Blue lines denote 42 edges present in *all* nine graphs, black edges show differences between graphs.

Here, the focus is not just on the graphs themselves, but also on the level of (dis)similarity between them. Here, we consider the Hamming distances between graphs. Let $\widehat{\mathbf{A}}_{ij}(\gamma_1, \gamma_2)$ and $\widehat{\mathbf{A}}_{kl}(\gamma_1, \gamma_2)$ denote the estimated adjacency matrices when $(U_1 = i, U_2 = j)$ and $(U_1 = k, U_2 = l)$, respectively, and the tuning parameters are $\gamma_1$ and $\gamma_2$. The Hamming distance between these two graphs is defined as the number of edges that is present in one graph but absent in the other:

$$\text{Hamming distance}\,(\gamma_1,\gamma_2)=\sum_{s<t}\mathbb{1}\left(\widehat{a}_{st}^{(ij)}\,(\gamma_1,\gamma_2)\neq\widehat{a}_{st}^{(kl)}\,(\gamma_1,\gamma_2)\right).$$

This distance reflects the number of edges that must be added and/or removed to transform one graph into another.

Figure 12 shows a heatmap of the Hamming distances between the nine graphs. The first three rows/columns represent the N0 control group, the fourth through sixth rows/columns represent the N1 results, and the last three rows/columns represent the N2+ results. Within these groups, the first row/column corresponds to no radiation, the second to 0.05 Gy, and the third to the highest dose of 2 Gy. The color gradient represents the Hamming distances. Within a group, there is almost no difference between no radiation and low radiation. However, these networks are clearly different from those with the maximum radiation dose. The networks between cancer groups within an intensity of irradiation are clearly different, regardless of the radiation dose.

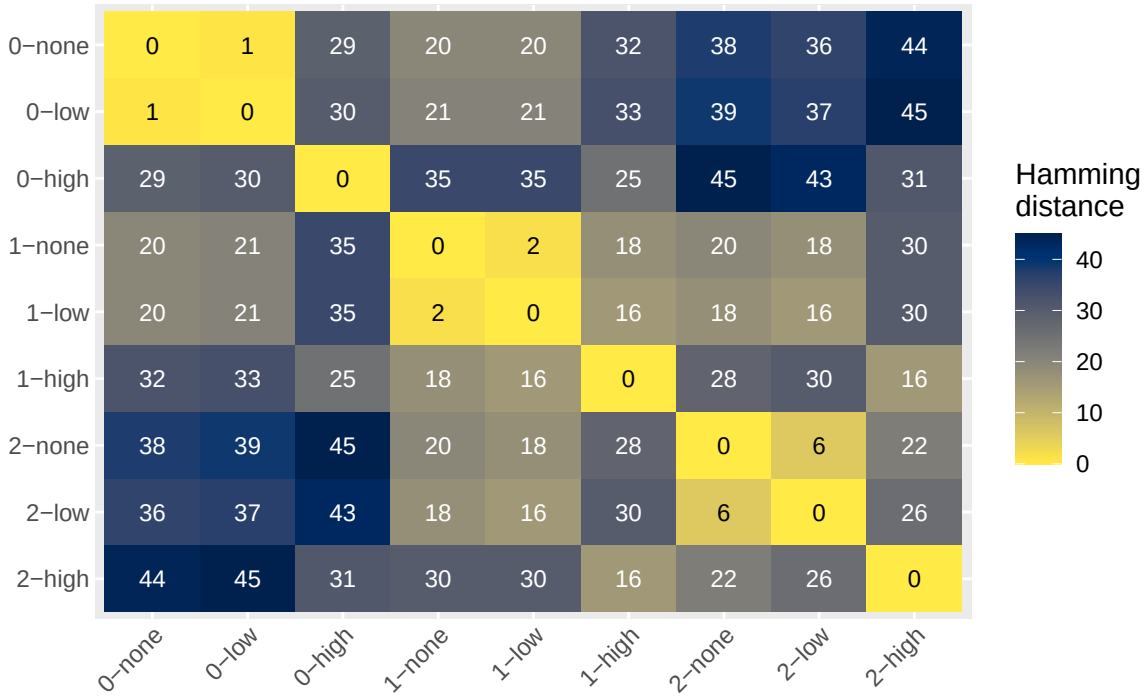



**Figure 12.** The Hamming distances between the nine graphs of a CVN with $\mathbf{W}_{\text{rad}}$ based on the KiKme data set when $\gamma_1 = 5 \times 10^{-5}$ and $\gamma_2 = 5 \times 10^{-6}$. The rows/columns are grouped together based on the cancer group the individuals belong to.

Table 2 demonstrates that within each CVN subgraph, only a subset of nodes are interconnected. There are only few differences within a group between radiation doses of 0 and 0.05 Gy, which may be due to random variation. Notably, the size of the subsets decreases with increasing radiation dose, whereas the density and the average node degree in each subgraph increases, implying a higher level of inter-connectivity among the remaining genes. The TP53 gene consistently exhibits the highest degree centrality, indicating its importance in the network regardless of the cancer group or radiation dosage. The TP53 gene encodes the tumor suppressor protein p53 that regulates DNA repair, apoptosis, and cell cycle control. It plays a critical role in the prevention of cancer, and mutations in this gene are common in human tumors.

Table 3 shows all adjacent genes of TP53 in each subgraph, nine of them being shared by all of them. Interestingly, TP53 is only associated with BAX in the control group for all radiation doses. The expression of BAX is upregulated in the presence of the tumor suppressor p53 following significant cellular damage, leading to the initiation of apoptotic (programmed) cell death. This is beneficial for cancer prevention. If there is a mutation in either TP53 or BAX, the process of TP53 initiating BAX may be disrupted and contribute to cancer development.

**Table 2.** Characteristics of the 9 estimated subgraphs. Nodes in each subgraph have at least one adjacent node. Singletons were removed from the original graph with $p = 191$ nodes.

| Group | Gy | Number of Nodes | Number of Edges | Graph Density | Avg. Shortest Path Length | Avg. Degree | Max. Degree and Nodes |
|---|---|---|---|---|---|---|---|
| **N0** | **0** | 31 | 75 | 0.16 | 2.38 | 4.84 | 21 |
| **N0** | **0.05** | 31 | 76 | 0.16 | 2.33 | 4.9 | 21 |
| **N0** | **2** | 27 | 68 | 0.19 | 2.15 | 5.04 | 21 |
| **N1** | **0** | 31 | 63 | 0.14 | 2.33 | 4.06 | 15 |
| **N1** | **0.05** | 30 | 61 | 0.14 | 2.34 | 4.07 | 14 |
| **N1** | **2** | 25 | 57 | 0.19 | 2.24 | 4.56 | 13 |
| **N2+** | **0** | 32 | 67 | 0.14 | 2.65 | 4.19 | 17 |
| **N2+** | **0.05** | 32 | 67 | 0.14 | 2.65 | 4.19 | 16 |
| **N2+** | **2** | 29 | 65 | 0.16 | 2.16 | 4.48 | 19 |

**Table 3.** Genes ajdacent to TP53 in the estimated CVN.

| Gene | Chr | Description | Found in | N00 | N01 | N02 | N10 | N11 | N12 | N20 | N21 | N22 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ATG4A | X | Autophagy Related 4a Cysteine Peptidase | N2+ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| RAD9B | 12 | Rad9 Checkpoint Clamp Component B | N2 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| STX6 | 1 | Syntaxin 6 | N2+ | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| TRIM13 | 13 | Tripartite Motif Containing 13 | N2+ | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| XRCC4 | 5 | X-Ray Repair Cross Complementing 4 | N2+ | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| CES2 | 16 | Carboxylesterase 2 | 2 Gy | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| SESN2 | 1 | Sestrin 2 | 2 Gy | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| FBXO22 | 15 | F-Box Protein 22 | in cancer group(s) and for N0 with 2 Gy | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 |
| XRCC6 | 22 | X-Ray Repair Cross Complementing 6 | in cancer group(s) and for N0 with 2 Gy | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| GRB2 | 17 | Growth Factor Receptor Bound Protein 2 | control group | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PRKAA1 | 5 | Protein Kinase Amp-Activated Catalytic Subunit Alpha 1 | control group | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| TAF3 | 10 | Tata-Box Binding Protein Associated Factor 3 | in N0 and N1 with <= 0.05 Gy | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| NOX4 | 11 | Nadph Oxidase 4 | control group | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| BAX | 19 | Bcl2 Associated X, Apoptosis Regulator | control group | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| ASCC3 | 6 | Activating Signal Cointegrator 1 Complex Subunit 3 | control group | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| TP53INP1 | 8 | Tumor Protein P53 Inducible Nuclear Protein 1 | control group | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| SPIDR | 8 | Scaffold Protein Involved In Dna Repair | 0-2 Gy in control group, unclear pattern for cancer groups | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 |
| DNMT3A | 2 | Dna Methyltransferase 3 Alpha | 0-2 Gy in control group, unclear pattern for cancer groups | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 |
| NABP1 | 2 | Nucleic Acid Binding Protein 1 | 0-2 Gy in control group, unclear pattern for cancer groups | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 |
| GADD45A | 1 | Growth Arrest And Dna Damage Inducible Alpha | 0-2 Gy in control group, unclear pattern for cancer groups | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 0 |
| RHNO1 | 12 | Rad9-Hus1-Rad1 Interacting Nuclear Orphan 1 | 0-2 Gy in control group, unclear pattern for cancer groups | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| RAD9A | 11 | Rad9 Checkpoint Clamp Component A | in all graphs | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| HMGA2 | 12 | High Mobility Group At-Hook 2 | in all graphs | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| BLM | 15 | Blm Recq Like Helicase | in all graphs | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| XRCC5 | 2 | X-Ray Repair Cross Complementing 5 | in all graphs | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| CASP3 | 4 | Caspase 3 | in all graphs | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| ANKRA2 | 5 | Ankyrin Repeat Family A Member 2 | in all graphs | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| SESN1 | 6 | Sestrin 1 | in all graphs | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| TNFRSF10B | 8 | Tnf Receptor Superfamily Member 10b | in all graphs | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| ACER2 | 9 | Alkaline Ceramidase 2 | in all graphs | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

## 4. Conclusions & Discussion

We introduced a new Gaussian graphical model called the covariate-varying network (CVN). This model allows the graph structure to change with multiple discrete external covariates, e.g., time and groups. To achieve this, we applied a LASSO penalty to the entries of the precision matrix to introduce sparsity. Additionally, we enforce similarities, or 'smoothness', between the different graphs by penalizing the absolute differences between them. We modeled smoothness using the concept of a meta graph with weighted edges, see Section 2.2. This approach is the first to address the problem of a graph changing with multiple external covariates. Many previously existing graphical models are special cases within the CVN graphical model class, see Section 2.3.

We determined that the complexity of our algorithm is cubic with respect to the number of variables ($p^3$) and squared with respect to the number of graphs ($m^2$), see Section 2.5. This is a downside, as the algorithm becomes computationally expensive and challenging to run for very large instances. The primary bottleneck is the second update step, where the wFLSA must be solved for each of the $\binom{p}{2}$ edges.

We proposed an alternative tuning parameterization (see Appendix B) that leads to more stable choices, as it is largely unaffected by the inclusion of additional variables or covariates. This approach can be easily adapted for use in related models, such as the graphical LASSO. Furthermore, we demonstrated how to interpolate a graph $G_{m+1}$, for which no direct observations are available, using an estimated CVN model with $m$ observed graphs.

We conducted a simulation study (see Section 2.8) using two types of graphs: Erdős–Rényi and Barabasi–Albert. These graphs were influenced by two external covariates, resulting in varied changes to the graph structure based on each covariate. We found that the estimation of the CVN was generally successful across different scenarios. However, selecting the tuning parameters poses significant challenges. The extent to which the graphs change with the external covariates did not appear to impact performance. The authors anticipate that if higher percentages were considered, one would see a decline in performance. It is important to note that comparing this method with others is difficult since, to the best of our knowledge, there are currently no other methods capable of handling the CVN graphical model.

To assess its applicability, we applied our method to a real dataset from the KiKme study (see Section 2.9). We showed that the method is able to recover differences and similarities of gene expression network structures that are specific to three different case/control groups and three different radiation dosages. This case study demonstrated that CVN is capable of estimating plausible networks in a high-dimensional setting where the overall number of samples ($n = 156$) is smaller than the number of nodes ($p = 191$). The number of samples for each subgraph was therefore much smaller. We provide a second, fully replicable case study in Appendix D, which further demonstrates that CVN can be readily applied to real data. The two data examples differ in the number of observations used to estimate the $m$ graphs. Due to the smoothing and weighting mechanisms in the CVN model, graphs borrow strength from one another: larger-sample networks help stabilize smaller-sample networks with which they are connected with in $\mathbf{W}$. Consequently, when there are substantial differences in sample sizes, networks with much larger samples tend to dominate the shared structure. In such cases, a larger $\lambda_2$ or a suitably weighted $\mathbf{W}$ may be needed to balance this effect. At the same time, excessively large $\lambda_2$ values can lead to oversmoothing, potentially masking genuine structural differences.

Moving forward, there are several avenues for future research. First, extending the method to handle continuous covariates would be a promising direction, broadening its applicability across different datasets. Currently, dealing with continuous external covariates necessitates data discretization. Second, based on our simulation study, we observed that the performance of AIC and BIC for tuning parameter selection is notably inadequate. Exploring alternative methods for parameter tuning could potentially improve performance. Lastly, due to the high algorithmic complexity of our method, exploring alternative optimization strategies, including heuristic approaches, could help reduce computational costs and enhance efficiency.

For practical applications, it is highly desirable to assess, after estimation, whether the inferred graphs differ significantly. Recent work has addressed this challenge. For example,Meng et al. [16] proposed a debiased estimator within a covariate-dependent Gaussian graphical regression framework, enabling valid statistical inference in high-dimensional settings. This approach allows one to identify which edges are significantly associated with a given covariate at a specified significance level. For smaller networks, Wang and Gao [15] established an asymptotic distribution for their estimator, which enables testing not only whether a specific edge is associated with the $i$-th covariate, but also whether the subgraphs differ statistically overall.

The algorithm has been implemented as an `R` package and is publicly accessible at https://bips-hb.r-universe.dev/CVN. Additionally, the CVN data simulator is available as an `R` package at https://github.com/bips-hb/CVNSim, and the codes for reproducing the simulation and case study are available at https://github.com/bips-hb/CVNStudy and https://github.com/bips-hb/cvn-case-study, respectively. The packages `CVNSim`, `CVNStudy`, and

`cvn-case-study` are also povided via Zenodo (DOI: https://zenodo.org/records/19232261). All results of the simulation study are hosted Shiny application at https://cvn.bips.eu.

### Author Contributions

Conceptualization and design: L.D., A.G. and R.F.; Methodology and formal analysis: L.D.; Software: L.D.; Interpretation of Results: L.D. and R.F.; Drafting of significant parts: L.D. and R.F.; Critical revision: L.D., A.G. and R.F.; Acquisition of research data: R.F.; All authors have read and agreed to the published version of the manuscript.

### Funding

The authors gratefully acknowledge the financial support of the German Research Foundation (DFG—Project FO 1045/2-1).

### Data Availability Statement

The Kikme data used in the case study was is based on Grandt et al. [34]. The study data were obtained under a data use agreement and contain sensitive information that cannot be publicly shared. Access to the data is therefore restricted. Any request to access the datasets should be directed to `vl-isibel@uni-mainz.de`. For the influenza vaccination study, we used the data provided by Wang and Gao [15].

### Acknowledgments

The first author thanks Professor Caroline Uhler and her team for the valuable and stimulating discussions. We sincerely thank the KiKME committee for generously providing the data used in our case study. We also thank the reviewers for their helpful comments.

### Conflicts of Interest

The authors declare no conflict of interest.

### Use of AI and AI-Assisted Technologies

During the preparation of this work, the authors used ChatGPT to perform language editing and improve readability. After using this tool/service, the authors reviewed and edited the content as needed and take full responsibility for the content of the published article.

### Appendix A. Interpolation

As outlined in Section 2.6, interpolating a graph $G_{m+1}$ using an estimated CVN model $\widehat{\boldsymbol{\Theta}}_{\text{CVN}}$ involves solving an optimization problem for each possible edge $\{s,t\}$ in the graph:

$$\widehat{\theta}_{st}^{(m+1)} = \underset{\theta_{st}^{(m+1)}\in\mathbb{R}}{\arg\min}\ \lambda_1|\theta_{st}^{(m+1)}| + \lambda_2\|\boldsymbol{\omega}^\top\widehat{\boldsymbol{\theta}}_{st} - \theta_{st}^{(m+1)}\|_1.$$

Here, $\widehat{\theta}_{st}^{(m+1)}$ represents the interpolated entry for the edge $\{s,t\}$ in the precision matrix. The vector $\boldsymbol{\omega} = (\omega_1,\omega_2,\ldots,\omega_m)^\top$ contains predetermined, non-negative weights, and $\widehat{\boldsymbol{\theta}}_{st} = \left(\widehat{\theta}_{st}^{(1)},\widehat{\theta}_{st}^{(2)},\ldots,\widehat{\theta}_{st}^{(m)}\right)^\top$ are the estimated values in the precision matrices for edge $\{s,t\}$ of the $m$ graphs. The tuning parameters are $\lambda_1 > 0$ and $\lambda_2 \geq 0$.

Although this optimization problem lacks an analytical solution, it is convex, allowing for solving it numerically using derivative-free search methods. We employ Brent's algorithm for this purpose [28]. For ease of notation, let $x = \theta_{st}^{(m-1)}$ and $\boldsymbol{y} = \widehat{\boldsymbol{\theta}}_{st}$. The goal is to solve:

$$\underset{x\in\mathbb{R}}{\arg\min}\, f(x;\boldsymbol{y},\boldsymbol{\omega}) = \underset{x\in\mathbb{R}}{\arg\min}\, \lambda_1|x| + \lambda_2\left\|\boldsymbol{\omega}^\top\boldsymbol{y} - x\right\|_1.$$

The convexity of the function $f(x;\boldsymbol{y},\boldsymbol{\omega})$ in $x$ is evident since

$$f(\alpha x_1 + (1-\alpha)x_2;\boldsymbol{y},\boldsymbol{\omega}) \leq \alpha f(x_1;\boldsymbol{y},\boldsymbol{\omega}) + (1-\alpha)f(x_2;\boldsymbol{y},\boldsymbol{\omega})$$

for all $x_1,x_2\in\mathbb{R}$, $\alpha\in[0,1]$, $\lambda_1 > 0$, $\lambda_2 \geq 0$, $\boldsymbol{\omega}\in\mathbb{R}_+^m$ and $\boldsymbol{y}\in\mathbb{R}^m$. This is easy to see for both

$$|\alpha x_1 + (1-\alpha)x_2| \leq \alpha|x_1| + (1-\alpha)|x_2|$$

and

$$\left\|\boldsymbol{\omega}^\top \boldsymbol{y} - (\alpha x_1 + (1-\alpha)x_2)\right\|_1 \leq \alpha \left\|\boldsymbol{\omega}^\top \boldsymbol{y} - x_1\right\|_1 + (1-\alpha)\left\|\boldsymbol{\omega}^\top \boldsymbol{y} - x_2\right\|_1 .$$

Note that the sign of $x$ (i.e., $\widehat{\theta}_{st}^{(m+1)}$) is irrelevant in this context, as our objective is solely to determine the existence of the edge $\{s,t\}$ in the interpolated graph, i.e., whether $\widehat{\theta}_{st}^{(m+1)} \neq 0$.

## Appendix B. An Alternative Tuning Parameterization

Due to the computational complexity of the algorithm exploring an extensive range of values for the tuning parameters $\lambda_1$ and $\lambda_2$ is impractical, and, as mentioned above, obtaining expert or prior knowledge about 'optimal' values for these parameters is challenging.

We introduce an alternative parameterization for the CVN model. Instead of using $\lambda_1$ and $\lambda_2$, we propose employing two different parameters, $\gamma_1$ and $\gamma_2$, which we define in this section. The benefit of this new parameterization is that the interpretation of these values do *not* heavily depend on the number of graphs ($m$) and/or the number of variables ($p$).

Suppose a model has been fitted and the best values for the tuning parameters $\lambda_1$ and $\lambda_2$ have already been selected based on some criterion, see the previous section. Now, consider a scenario where the same data set is used, but a number of variables are added or removed (thereby changing $p$), and/or other external covariates are added or removed (changing the number of graphs $m$). Due to the way the model is defined, see Equation (5), the interpretation of the penalty terms $\lambda_1$ and $\lambda_2$ change, i.e., the optimal values for the first model provide little information about the best values for the tuning parameters for the new model.

Note that this challenge is not unique to the CVN model; it extends to other penalized regression problems as well. However, this challenge is typically overlooked for two main reasons. Firstly, it is in other situations uncommon for the considered variables to change. Secondly, the associated algorithm is usually computationally light, making it straightforward to reapply to new datasets. Nevertheless, we would like to point out that the alternative tuning parameterization proposed here may be useful for other LASSO-related methods as well.

In the existing parameterization, penalties are imposed on the sum of the $L_1$-norm of the precision matrices, governed by $\lambda_1$, and the sum of the $L_1$-norm of the differences between the precision matrices, regulated by $\lambda_2$, see Equation (5). We propose a modification where we penalize each individual edge in the CVN model to regulate sparsity, and each pair of edges in the CVN model to control the level of smoothness. We denote the new parameters as $\gamma_1$, regulating sparsity, and $\gamma_2$, managing smoothness.

Recall that the total number of potential edges in the CVN model is $mp(p-1)$, and the total number of pairs of potential edges is $m(m-1)p(p-1)$. We define $\gamma_1$ and $\gamma_2$ as follows:

$$\gamma_1 = 2\frac{\lambda_1}{mp(p-1)} \qquad \text{and} \qquad \gamma_2 = 4\frac{\lambda_2}{m(m-1)p(p-1)}.$$

Consequently, the sparsity penalty term can be expressed as

$$\lambda_1 \sum_{i=1}^{m} \|\boldsymbol{\Theta}_i\|_1^{\text{off}} = \sum_{i=1}^{m} \sum_{s<t} \gamma_1 |\theta_{st}^{(i)}|.$$

In other words, $\gamma_1$ regulates how much an individual edge in a graph is penalized. Consequently, alterations in the number of edges have a relatively modest impact on this tuning parameter in comparison to $\lambda_1$. Similarly, the smoothness penalty term of the CVN model can be written in terms of $\gamma_2$:

$$\lambda_2 \sum_{i<j} \|\boldsymbol{\Theta}_i - \boldsymbol{\Theta}_j\|_1^{\text{off}} = \sum_{i<j} w_{ij} \sum_{s,t} \gamma_2 |\theta_{st}^{(i)} - \theta_{st}^{(j)}|.$$

Here, $\gamma_2$ is used to penalize a single pair of edges. Once more, changing the number of graphs and edges has little effect on the interpretation of this tuning parameter.

The idea that optimal values for $\gamma_1$ and $\gamma_2$ are consistent across various datasets hinges on the assumption that both the sparsity of the graphs and their smoothness will remain relatively stable across those datasets.

We use $\gamma_1$ and $\gamma_2$ in the simulation study (2.8). Given that we vary the number of variables, $p$, and the number of graphs, $m$, one would otherwise need to consider different values for the tuning parameters $\lambda_1$ and $\lambda_2$ for the different simulation setups. By using $\gamma_1$ and $\gamma_2$, we can consider the same tuning parameter range for all simulations.

## Appendix C. Illustrative Example of the Smoothing Term

To provide intuition for the smoothing parameter in CVN estimation, we illustrate how the off-diagonal $L_1$ norm

$$\|\boldsymbol{\Theta}_i - \boldsymbol{\Theta}_j\|_1^{\text{off}} = \sum_{s \neq t} |\theta_{i,st} - \theta_{j,st}|$$

relates to structural differences between graphs. We constructed a base precision matrix $\boldsymbol{\Theta}_1$ for a 5-node network and generated three variants with increasing differences by altering the values of specific entries corresponding to edges:

- Small difference ($\|\boldsymbol{\Theta}_1 - \boldsymbol{\Theta}_{small}\|_1^{\text{off}} = 0.1$): values for single 'edge' slightly altered.
- Medium difference ($\|\boldsymbol{\Theta}_1 - \boldsymbol{\Theta}_{medium}\|_1^{\text{off}} = 1.1$): values for a few 'edges' increased or decreased.
- Large difference ($\|\boldsymbol{\Theta}_1 - \boldsymbol{\Theta}_{large}\|_1^{\text{off}} = 3$): many values modified, reflecting multiple structural changes, including new connections and removed edges.

Figure A1 shows the resulting networks. Small values of the norm correspond to graphs that differ only slightly in their precision matrix entries, while larger values reflect substantial changes. This example provides visual intuition for how the smoothing penalty influences the similarity between estimated networks.

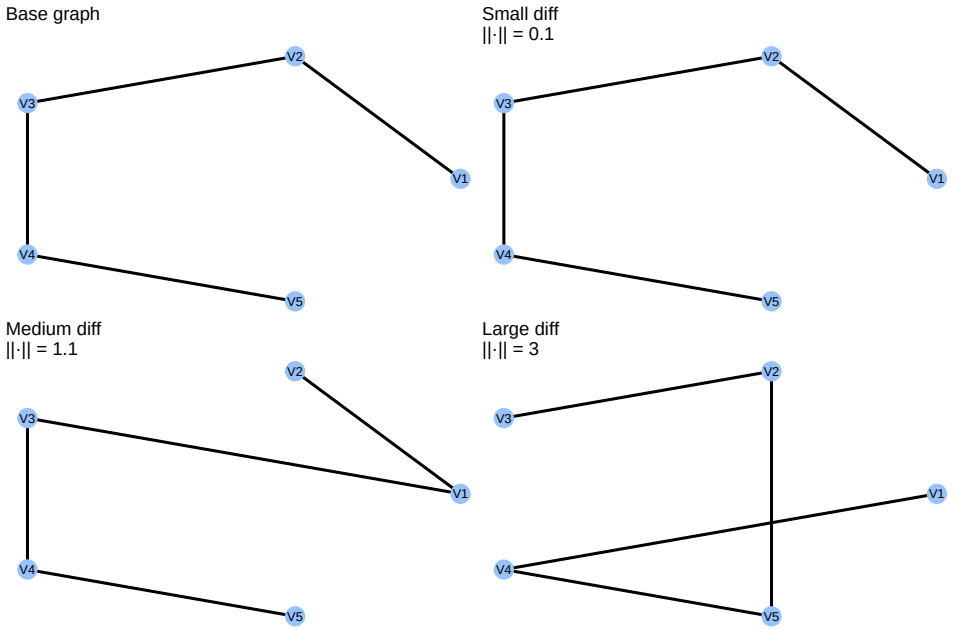



**Figure A1.** Illustrative networks showing increasing off-diagonal $L_1$ norm differences from the base graph. Small values correspond to minor changes in precision matrix entries, while larger values reflect more substantial differences.

## Appendix D. Influenza Vaccination Study

In addition to our simulation studies, we analyzed a real data set previously considered by Wang and Gao [15]. We used the publicly available data provided in their GitHub repository (https://github.com/XuranMeng/MTCDGGM), which is based on gene expression measurements from peripheral blood samples collected before and after trivalent influenza vaccination [35]. The dataset comprises expression profiles of 68 selected genes measured at four time points (Day 0, Day 1, Day 3, and Day 14), with sample sizes of 218, 217, 206, and 207 observations. These time points denote the day before vaccination and 1, 3, and 14 days post-vaccination. These genes were chosen based on evidence of both transcriptional response to vaccination and genetic regulation. Following Wang and Gao [15], this dataset enables the investigation of dynamic gene regulatory networks and their changes over time in response to vaccination. We applied our CVN method using a grid matrix $\mathbf{W}_{\text{grid}}$ with weights accordingly to the distance between days.

Influenza vaccination triggers a strong, coordinated immune response, which reshapes transcriptional networks. As shown in Figure A2, networks became sparser and several genes exhibited decreased connectivity in the expression networks, including ABCA7, LRRC37A4, and PJA2, which are implicated in immune-related processes, and SNHG5 and VASH1, which are involved in cellular stress responses. This suggests a reorganization of

co-expression patterns following vaccination. In contrast, fourteen days after influenza vaccination, NFXL1 exhibited increased connectivity, suggesting it may become more centrally integrated into the transcriptional network. NFXL1 encodes a transcription factor that may regulate multiple target genes in response to vaccination, suggesting a potential role in coordinating the post-vaccination transcriptional program.

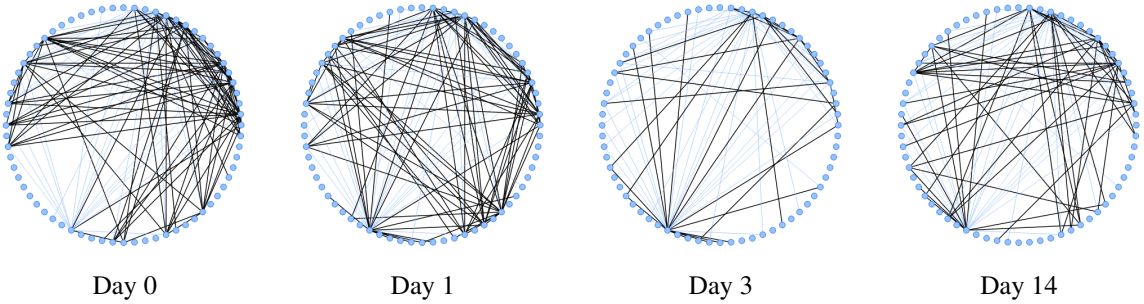



**Figure A2.** Covariate-varying gene networks inferred from a set of 68 genes in the influenza vaccination dataset studied by Wang and Gao [15]. Blue lines denote 45 edges present in *all* four graphs, black edges show differences between graphs. The number of edges in these graphs are 136, 130, 75, and 99, respectively.

## References

1. Newman, M. *Networks*; Oxford University Press: Oxford, UK, 2018.
2. Barabasi, A.L.; Oltvai, Z.N. Network biology: Understanding the cell's functional organization. *Nat. Rev. Genet.* **2004**, *5*, 101–113.
3. Curtis, H.; Blaser, M.J.; Dirk, G.; et al. Structure, function and diversity of the healthy human microbiome. *Nature* **2012**, *486*, 207–214.
4. de Vos, W.M.; de Vos, E.A. Role of the intestinal microbiome in health and disease: From correlation to causation. *Nutr. Rev.* **2012**, *70*, S45–S56.
5. Badhwar, A.; Tam, A.; Dansereau, C.; et al. Resting-state network dysfunction in Alzheimer's disease: A systematic review and meta-analysis. *Alzheimer's Dement. Diagn. Assess. Dis. Monit.* **2017**, *8*, 73–85.
6. Cai, T.; Liu, W.; Luo, X. A constrained $\ell_1$ minimization approach to sparse precision matrix estimation. *J. Am. Stat. Assoc.* **2011**, *106*, 594–607.
7. Danaher, P.; Wang, P.; Witten, D.M. The joint graphical LASSO for inverse covariance estimation across multiple classes. *J. R. Stat. Soc. Ser. B Stat. Methodol.* **2014**, *76*, 373–397.
8. Friedman, J.; Hastie, T.; Tibshirani, R. Sparse inverse covariance estimation with the graphical LASSO. *Biostatistics* **2008**, *9*, 432–441.
9. Brozyna, A.; Zbytek, B.; Granese, J.; et al. Mechanism of UV-related carcinogenesis and its contribution to nevi/melanoma. *Expert Rev. Dermatol.* **2007**, *2*, 451–469.
10. de la Cuesta-Zuluaga, J.; Corrales-Agudelo, V.; Velásquez-Mejía, E.P.; et al. Gut microbiota is associated with obesity and cardiometabolic disease in a population in the midst of Westernization. *Sci. Rep.* **2018**, *8*, 11356.
11. Gibberd, A.J.; Nelson, J.D. Regularized estimation of piecewise constant Gaussian graphical models: The group-fused graphical Lasso. *J. Comput. Graph. Stat.* **2017**, *26*, 623–634.
12. Hallac, D.; Park, Y.; Boyd, S.; et al. Network inference via the time-varying graphical Lasso. In Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, NS, Canada, 13–17 August 2017.
13. Monti, R.P.; Hellyer, P.; Sharp, D.; et al. Estimating time-varying brain connectivity networks from functional MRI time series. *Neuroimage* **2014**, *103*, 427–443.
14. Vinciotti, V.; Wit, E.C.; Richter, F. Random graphical model of microbiome interactions in related environments. *J. Agric. Biol. Environ. Stat.* **2024**, *31*, 46–59.
15. Wang, J.; Gao, X. High-dimensional covariate-dependent Gaussian graphical models. *J. Comput. Graph. Stat.* **2025**, *34*, 1642–1654.
16. Meng, X.; Zhang, J.; Li, Y. Statistical inference on high-dimensional covariate-dependent Gaussian graphical regressions. *Biometrics* **2025**, *81*, ujaf165.
17. Ni, Y.; Stingo, F.C.; Baladandayuthapani, V. Bayesian Covariate-Dependent Gaussian Graphical Models with Varying Structure. *J. Mach. Learn. Res.* **2022**, *23*, 1–29.
18. Zhang, J.; Li, Y. High-Dimensional Gaussian Graphical Regression Models with Covariates. *J. Am. Stat. Assoc.* **2023**, *118*, 2088–2100.
19. Marron, M.; Brackmann, L.K.; Schwarz, H.; et al. Identification of genetic predispositions related to ionizing radiation in primary human skin fibroblasts from survivors of childhood and second primary cancer as well as cancer-free controls: Protocol for the nested case-control study KiKme. *JMIR Res. Protoc.* **2021**, *10*, e32395.

20. Lauritzen, S. *Graphical Models*; Oxford Statistical Science Series; Clarendon Press: Oxford, UK, 1996.
21. Uhler, C. Gaussian graphical models. In *Handbook of Graphical Models*; CRC Press: Boca Raton, FL, USA, 2018; pp. 217–238.
22. Banjeree, O.; Ghaoui, L.E. Model selection through sparse maximum likelihood estimation for multivariate Gaussian or binary data. *J. Mach. Learn. Res.* **2008**, *9*, 485–516.
23. Boyd, S.; Vandenberghe, L. *Convex Optimization*; Cambridge University Press: Cambridge, UK, 2004.
24. Boyd, S.; Parikh, N.; Chu, E.; et al. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Found. Trends Mach. Learn.* **2010**, *3*, 1–122.
25. Wu, N.; Huang, J.; Zhang, X.F.; et al. Weighted fused pathway graphical Lasso for joint estimation of multiple gene networks. *Front. Genet.* **2019**, *10*, 623.
26. Witten, D.M.; Tibshirani, R. Covariance-regularized regression and classification for high dimensional problems. *J. R. Stat. Soc. Ser. B Stat. Methodol.* **2009**, *71*, 615–636.
27. Dijkstra, L.; Hanke, M.; Koenen, N.; Foraita, R. An alternating direction method of multipliers algorithm for the weighted fused LASSO signal approximator. *J. Stat. Comput. Simul.* **2025**, *95*, 2979–2997.
28. Brent, R.P. *Algorithms for Minimization without Derivatives*; Dover Publications: Mineola, NY, USA, 2013.
29. Hastie, T.; Tibshirani, R.; Friedman, J.H.; Friedman, J.H. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*; Springer: New York, USA, 2009; Volume 2.
30. Chen, J.; Chen, Z. Extended Bayesian information criteria for model selection with large model spaces. *Biometrika* **2008**, *95*, 759–771.
31. Ando, T. Bayesian predictive information criterion for the evaluation of hierarchical Bayesian and empirical Bayes models. *Biometrika* **2007**, *94*, 443–458.
32. Barabasi, A.L.; Pósfai, M. *Network Science*; Cambridge University Press: Cambridge, UK, 2016.
33. Liu, H.; Wang, L. TIGER: A tuning-insensitive approach for optimally estimating Gaussian graphical models. *Electron. J. Stat.* **2017**, *11*, 241–294.
34. Grandt, C.L.; Brackmann, L.K.; Poplawski, A.; et al. Identification of lncRNAs involved in response to ionizing radiation in fibroblasts of long-term survivors of childhood cancer and cancer-free controls. *Front. Oncol.* **2023**, *13*, 1158176.
35. Franco, L.M.; Bucasas, K.L.; Wells, J.M.; et al. Integrative genomic analysis of the human immune response to influenza vaccination. *eLife* **2013**, *2*, e00299.