Dynamic Graph Topology Learning with Non-Convex Penalties

Reza Mirzaeifard[⋆], Vinay Chakravarthi Gogineni[⋆], Naveen K. D. Venkategowda[§], Stefan Werner[⋆]
[⋆]Dept. of Electronic Systems, Norwegian University of Science and Technology-NTNU, Norway
[§]Department of Science and Technology, Linköping University, Sweden
E-mails: {reza.mirzaeifard, vinay.gogineni, stefan.werner}@ntnu.no, naveen.venkategowda@liu.se

***Abstract*—This paper presents a majorization-minimization-based framework for learning time-varying graphs from spatial-temporal measurements with non-convex penalties. The proposed approach infers time-varying graphs by using the log-likelihood function in conjunction with two non-convex regularizers. Using the log-likelihood function under a total positivity constraint, we can construct the Laplacian matrix from the off-diagonal elements of the precision matrix. Furthermore, we employ non-convex regularizer functions to constrain the changes in graph topology and associated weight evolution to be sparse. The experimental results demonstrate that our proposed method outperforms the state-of-the-art methods in sparse and non-sparse situations.**

***Index Terms*—Graph topology learning, time-varying graphs, Laplacian constraints, non-smooth and non-convex penalties**

I. Introduction

The data in a variety of networked structures, including social networks [1], brain networks [2], traffic networks [3], electronic networks [4], and sensor networks [5], exhibit certain intrinsic relationships with the underlying network structure. The inherent network structure can be exploited to analyze data from large-scale networks in an efficient manner. A graph and graph signal are the most natural ways to represent the inherent structure and network data [6]. However, in many applications the underlying graph is unavailable [1]–[3]. In such scenarios one of the challenges in graph signal processing is to learn the graph topology.

Prior to recent developments, approaches to identify graph topology relied on learning static networks, i.e., networks whose structure does not change over time [6]. However, the increasing prevalence of networks with time-varying components has quickly necessitated the development of alternative methods for graph topology learning. For example, it is of interest to infer time-varying networks of functional components in the brain using electroencephalography (EEG) or functional magnetic resonance imaging (MRI) data [7], to identify relationships between companies based on historical stock prices [8], and to capture changes in the environment factors [9]. As a result, it would be impossible to take into account the dynamic nature of these structures using a static approach. A straightforward approach would be to group the observation time into non-overlapping windows and then estimate each graph individually using a static graph learning algorithm. However, such an approach discards the temporal relationship between graphs, which can improve estimation accuracy, and it may require a large sample size within each time window, which is not always feasible.

Several works in the literature have addressed the time-varying graph topology learning [10]–[13]. Time-varying graphs are often learned by splitting the data into separate windows and learning a graph for each separate window with assumptions about the relationships between the graphs. Therefore, a transition constraint must be used [10], [13]. The differences between the algorithms are a result of considering different objective and regulation functions, which are influenced by prior knowledge of graph data and transitions, such as sparsity, smoothness, and bandlimiting. In [10] an algorithm for time-varying graph learning is proposed considering the graph signal to be smooth and the transitions between graphs sparse. As a result, this algorithm considers Direchlet energy functions as the main objective function that is regulated by Frobenius norm to ensure graph sparsity, and fussed LASSO (FLasso) to enforce sparse temporal transitions. However, a Dirichlet energy objective function can be viewed as an approximation of a log-likelihood function for learning graphs [8], and in general using a log-likelihood function may lead to better estimation accuracy. Moreover, the Frobenius norm does not promote sparsity and zero elements as effectively as the l_1 norm, minimum concave penalty (MCP) [14] or smoothly clipped absolute deviation (SCAD) [15] penalties.

This paper proposes log-likelihood approaches with sparse penalties to enforce sparsity in graphs, and sparse penalties for transition in order to emphasize the similarity between successive graphs. Since the l_1 norm with log-likelihood cannot provide sparse solutions for graphs, we use SCAD to encourage sparsity in each time window [16]. For the temporal regularizer, both SCAD and FLasso are considered. In contrast to FLasso, SCAD is more capable of identifying preserved and changed edges across successive windows, which makes it a more efficient temporal regulator. We consider the total positivity constraint to obtain a Laplacian matrix from the off-diagonal elements of the precision matrix. The resulting optimization problem is solved using the majorization-minimization (MM) framework [17], where each minimization problem can be effectively solved with an alternating direction method of multipliers (ADMM) [18] based algorithm.

This work was supported by the Research Council of Norway.

II. Preliminaries and Problem Formulation

Consider a weighted undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{W})$, where $\mathcal{V} = \{1, \cdots, P\}$ is the node set, $\mathcal{E} \subseteq \{1, \cdots, P\} \times \{1, \cdots, P\}$ denotes the edge set, and $\mathbf{W}$ is the symmetric weighted adjacency matrix, whose elements are non-negative edge weights. It is assumed that there no self-loops or multiple edges in the graph, which implies diag$(\mathbf{W}) = \mathbf{0}$. A graph signal is given by $\mathbf{x} = [x_1, \cdots, x_P]^{\mathrm{T}} \in \mathbb{R}^P$, where x_i is a measurement at node i. If $(i,j) \in \mathcal{E}$, x_i and x_j are mutually dependent or similar with the similarity proportional to $\mathbf{W}_{ij}$. The goal of graph topology learning is to estimate the weighted adjacency matrix $\mathbf{W}$, given the data matrix $\mathbf{X} = [\mathbf{x}^{(1)}, \cdots, \mathbf{x}^{(M)}]$ as input, where $\mathbf{x}^{(m)}$ is the mth data vector and M is the number of graph signals.

In graph theory, the graph Laplacian is defined by $\mathbf{L} = \mathbf{D} - \mathbf{W}$, where $\mathbf{D}$ is a diagonal weighted degree matrix with $D_{ii} = \sum_{j=1}^{P} D_{ij}, \forall i \in \{1, \cdots, P\}$. Equivalently, the Laplacian can be defined as $\mathbf{L} \in \mathcal{L}$ where $\mathcal{L}$ is

$$\mathcal{L} = \left\{ \begin{array}{c} \mathbf{L} \in \mathbb{R}^{N \times N} : \mathbf{L} = \mathbf{L}^{\mathrm{T}}, L_{ij} \leq 0, (i \neq j), \\ L_{ii} = -\sum_{i \neq j} L_{ij} \end{array} \right\}. \quad (1)$$

Since the Laplacian matrix is not a full rank matrix, the generalized Laplacian of a graph may be defined as $\mathbf{L}_g = \mathbf{L} + \mathbf{V}$, where $\mathbf{V}$ is a diagonal matrix. If all the diagonal elements of $\mathbf{V}$ are strictly positive, then $\mathbf{L}_g$ is positive-definite.

The graph signal $\mathbf{x}$ generation can be described as [19]

$$\mathbf{x} = \mathbf{H}\mathbf{x}_0 + \boldsymbol{\epsilon}, \quad (2)$$

where $\mathbf{x}_0$ is the latent variable, $\mathbf{H}$ represents the graph Fourier transformation matrix, and $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \sigma_\epsilon^2 \mathbf{I})$ is the additive noise. Assuming that $\mathbf{x}_0$ is drawn from a zero mean Gaussian distribution, it is easily shown that the graph signal can be modeled as follows [19]:

$$\mathbf{x}^{(m)} \sim \mathcal{N}(\mathbf{0}, \mathbf{L}^{\dagger} + \sigma_\epsilon^2 \mathbf{I}), \ \ \forall m \in \{1, \cdots, M\}, \quad (3)$$

where $\mathbf{A}^{\dagger}$ represents the Moore-Penrose pseudo-inverse of $\mathbf{A}$. Therefore, from (3) one can see the dependence of the graph signal on the Laplacian matrix $\mathbf{L}$ or generealized Laplacian matrix $\mathbf{L}_g = \mathbf{L} + \sigma_\epsilon^2 \mathbf{I}$. Consequently, it can be assumed that $\mathbf{x}$ is multivariate Gaussian and estimate the precision matrix $\boldsymbol{\Omega} = \boldsymbol{\Sigma}^{-1}$ of Gaussian graphical model based on the sample covariance matrix $\hat{\boldsymbol{\Sigma}} = \frac{1}{M}\mathbf{X}\mathbf{X}^{\mathrm{T}}$.

The log-likelihood is a well studied objective function for estimating precision matrices of Gaussian graphical models. The Laplacian matrix can b estimated by considering either $\boldsymbol{\Omega} = \mathbf{L}$ with the constraint $\boldsymbol{\Omega} \in \mathcal{L}$ [16], [20] or having $\boldsymbol{\Omega} = \mathbf{L}_g$ and $\boldsymbol{\Omega} \succ 0$ [21]. Furthermore, one can also impose the total positivity constraint $\Omega_{ij} \leq 0, \forall i \neq j$ and extract the weighted matrix from off-diagonal elements of the precision matrix [22]. The Laplacian matrix can be estimated by the weighted adjacency matrix. The problem of estimating the precision matrix with log-likelihood objective with penalties and total positivity constraint can be expressed as [22]

$$\begin{aligned} \hat{\boldsymbol{\Omega}} = & \underset{\{\boldsymbol{\Omega}\}}{\arg\min} \quad -\log(\det(\boldsymbol{\Omega})) + \mathrm{tr}(\boldsymbol{\Omega}\hat{\boldsymbol{\Sigma}}) + \omega P_{\mathrm{od}}(\boldsymbol{\Omega}), \quad (4) \\ & \text{subject to} \quad \boldsymbol{\Omega} \succ 0, \ \ [\boldsymbol{\Omega}]_{ij} \leq 0 \ \forall i \neq j, \end{aligned}$$

where $P_{\mathrm{od}}(\cdot)$ is the penalty function to incorporate prior knowledge about the off-diagonal elements of $\boldsymbol{\Omega}$. Several functions can be used to impose sparsity on a structure. For example, the LASSO penalty is a well-known for enforcing sparsity in graphs. However, it has been shown that the LASSO penalty is unable to recover the Laplacian structure when used with log-likelihood objective function; thus, resulting in estimate with fully connected graphs [16]. Therefore, in this paper, we investigate the SCAD function which is non-convex to provide sparsity in the estimated graph with a lower bias effect than LASSO [16], [23].

A time-varying weighted graph refers to a sequence of weighted graphs, i.e., $\{\mathcal{G}_t\}_{t=1}^{T} = (\mathcal{V}, \mathcal{E}_t, \mathbf{W}_t)_{t=1}^{T}$. All nodes in each graph remain the same, but the edges may change. Time-varying graph learning involves identifying the sequence of graph Laplacians $\{\mathbf{L}_1, \cdots \mathbf{L}_T\}$, associated with the underlying sequence $\{\mathbf{X}_1, \cdots, \mathbf{X}_T\}$. We can model time-varying graph learning by utilizing (3) and a graph evolution process as [10]

$$\begin{aligned} \mathbf{x}_t^{(m)} &\sim \mathcal{N}(0, \mathbf{L}_t^{\dagger} + \sigma_\epsilon^2 \mathbf{I}), \ \ \forall m \in \{1, \cdots, M\} \\ \mathbf{L}_t &= \begin{cases} \mathbf{L}_1, & t = 1 \\ \mathbf{L}_{t-1} + \Delta \mathbf{L}_t, & t > 1 \end{cases} \end{aligned} \quad (5)$$

where $\mathbf{x}_t^{(m)}$, $\mathbf{L}_t$, and $\Delta \mathbf{L}_t$ represent, respectively, a signal, graph Laplacian and graph variation at a given time. In order to enhance the estimation accuracy of time-varying graphs, one can utilize the *prior* information about the graph variation. Thus, the problem of learning time-varying graph topology can be formulated as an optimization given by

$$\begin{aligned} \hat{\boldsymbol{\Omega}} = \underset{\{\boldsymbol{\Omega}_1, \cdots, \boldsymbol{\Omega}_T\}}{\arg\min} & \sum_{t=1}^{T} -\log(\det(\boldsymbol{\Omega}_t)) + \mathrm{tr}(\boldsymbol{\Omega}_t \hat{\boldsymbol{\Sigma}}_t) + \omega P_{\mathrm{od}}(\boldsymbol{\Omega}_t) \\ & + \eta \sum_{t=2}^{T} \Psi(\boldsymbol{\Omega}_t - \boldsymbol{\Omega}_{t-1}), \\ \text{subject to} \quad & \boldsymbol{\Omega}_t \succ 0, \ \ [\boldsymbol{\Omega}_t]_{ij} \leq 0 \ \forall i \neq j, \forall t \in \{1, \cdots, T\}, \end{aligned} \quad (6)$$

where $\Psi(\cdot)$ is a temporal regulation function. In order to enforce sparse change between two successive time-windows, the FLasso regulation function can be used [24]. However, FLasso does not distinguish between situations where an edge changes or remains the same in two successive time windows. This leads to unexpected similarity between edges across successive windows and a high bias. Also, when an abrupt change occurs, FLasso leads to a smooth change. Several non-convex penalties, such as MCP and SCAD, can help to distinguish between the two situations, so they can both maximize similarity when needed and promote change when necessary with a low bias effect. Therefore, we study the possibility of using non-convex penalties, such as SCAD, as a temporal regulation function $\Psi(\cdot)$.

III. Proposed Majorization-Minimization based Learning Approach

In order to deal with the non-convexity of SCAD in optimization problem (6), we use the majorization-minimization

(MM) framework [17], which is a two-step iterative procedure as explained below. In the first step, one can construct a majorized function $f^{(l+1)}(\mathbf{\Omega})$ that locally approximates the objective function $F(\mathbf{\Omega})$ at $\hat{\mathbf{\Omega}}^{(l)}$, which at iteration $l+1$ is given by

$$f^{(l+1)}(\mathbf{\Omega}) = \sum_{t=1}^{T} -\log(\det(\mathbf{\Omega}_t)) + \mathrm{tr}(\mathbf{\Omega}_t \hat{\mathbf{\Sigma}}_t) + \sum_{i\neq j} \omega_{t,ij}^{(l+1)} |[\mathbf{\Omega}_t]_{ij}| + \sum_{t=2}^{T}\sum_{i=1}^{T}\sum_{j=1}^{T} \eta_{t,ij}^{(l+1)} |[\mathbf{\Omega}_t - \mathbf{\Omega}_{t-1}]_{ij}|, \tag{7}$$

where $\eta_{t,ij}^{(l+1)} = |\eta P'_{\text{od}}([\hat{\mathbf{\Omega}}_t^{(l)}]_{ij})|$ and $\omega_{t,ij}^{(l+1)} = |\omega \Psi'([\hat{\mathbf{\Omega}}_t^{(l)} - \hat{\mathbf{\Omega}}_{t-1}^{(l)}]_{ij})|$ and $P'_{\text{od}}(x)$ is the sub-derivative of function $P'_{\text{od}}(\cdot)$ at point x. The second step consists of minimizing the majorized function $f^{(l+1)}(\mathbf{\Omega})$ under the mentioned constraints. For this purpose, we propose an ADMM-based method to solve the optimization.

To apply ADMM algorithm, one can rewrite the (6) as

$$\hat{\mathbf{\Omega}}^{(l+1)} = \underset{\{\mathbf{\Omega}_1,\cdots,\mathbf{\Omega}_T\}}{\arg\min} \sum_{t=1}^{T} \Big(-\log(\det(\mathbf{\Omega}_t)) + \mathrm{tr}(\mathbf{\Omega}_t \hat{\mathbf{\Sigma}}_t) + \sum_{i\neq j} \omega_{t,ij}^{(l+1)} |[\mathbf{Z}_{0,t}]_{ij}| \Big) + \sum_{t=2}^{T}\sum_{i=1}^{T}\sum_{j=1}^{T} \eta_{t,ij}^{(l+1)} |[\mathbf{Z}_{2,t} - \mathbf{Z}_{1,t-1}]_{ij}|,$$
$$\text{subject to} \quad \mathbf{Z}_{0,t} = \mathbf{\Omega}_t, \forall t \in \{1,\cdots,T\}$$
$$(\mathbf{Z}_{1,t-1}, \mathbf{Z}_{2,t}) = (\mathbf{\Omega}_{t-1}, \mathbf{\Omega}_t), \forall t \in \{2,\cdots,T\}$$
$$\mathbf{\Omega}_t \succ 0, \ \ [Z_{0,t}]_{ij} \leq 0, \ \forall i \neq j, \forall t \in \{1,\cdots,T\}, \tag{8}$$

where $\mathbf{Z} = \{\mathbf{Z}_0, \mathbf{Z}_1, \mathbf{Z}_2\} = \{[\mathbf{Z}_{0,1},\cdots,\mathbf{Z}_{0,T}], [\mathbf{Z}_{1,1},\cdots, \mathbf{Z}_{1,T-1}], [\mathbf{Z}_{2,2},\cdots,\mathbf{Z}_{2,T}]\}$ is a set of auxiliary variables. Consequently, the Lagrangian function can be written as

$$\mathcal{L}_\rho(\mathbf{\Omega}, \mathbf{Z}, \mathbf{U}) = \sum_{t=1}^{T} \Big(-\log(\det(\mathbf{\Omega}_t)) + \mathrm{tr}(\mathbf{\Omega}_t \hat{\mathbf{\Sigma}}_t) + \sum_{i\neq j} \omega_{t,ij}^{(l+1)} |[\mathbf{Z}_{0,t}]_{ij}| + \frac{\rho}{2}\|\mathbf{\Omega}_t - \mathbf{Z}_{0,t} + \mathbf{U}_{0,t}\|_F \Big) + \sum_{t=2}^{T} \Big(\sum_{i=1}^{T}\sum_{j=1}^{T} \eta_{t,ij}^{(l+1)} |[\mathbf{Z}_{2,t} - \mathbf{Z}_{1,t-1}]_{ij}| + \frac{\rho}{2}\|\mathbf{\Omega}_{t-1} - \mathbf{Z}_{1,t-1} + \mathbf{U}_{1,t-1}\|_F + \frac{\rho}{2}\|\mathbf{\Omega}_t - \mathbf{Z}_{2,t} + \mathbf{U}_{2,t}\|_F \Big), \tag{9}$$

where ρ denotes the ADMM penalty parameter, $\mathbf{U} = \{\mathbf{U}_0, \mathbf{U}_1, \mathbf{U}_2\} = \{[\mathbf{U}_{0,1},\cdots,\mathbf{U}_{0,T}], [\mathbf{U}_{1,1},\cdots,\mathbf{U}_{1,T-1}], [\mathbf{U}_{2,2},\cdots,\mathbf{U}_{2,T}]\}$ is the scaled dual variable, and $\mathbf{\Omega} = \{\mathbf{\Omega}_1,\cdots,\mathbf{\Omega}_P\}$. Since the constraints $\mathbf{\Omega}_t \succ \mathbf{0}$, and $\mathbf{Z}_{0,t} \in \mathcal{V}_p = \{V_{ij} \leq 0, \forall i \neq j\}$ can be involved in the minimization of the sub-problems, there is no need for dual variables associated with them. In this case, the problem can be solved with the classic two-block ADMM algorithm whose $(k+1)$th iteration can be written as

$$\mathbf{\Omega}^{(k+1)} = \underset{\mathbf{\Omega}\succ\mathbf{0}}{\arg\min}\, \mathcal{L}_\rho(\mathbf{\Omega}, \mathbf{Z}^{(k)}, \mathbf{U}^{(k)}), \tag{10}$$

$$\mathbf{Z}^{(k+1)} = \underset{\mathbf{Z}_0\in\mathcal{V}_p, \mathbf{Z}_1, \mathbf{Z}_2}{\arg\min}\, \mathcal{L}_\rho(\mathbf{\Omega}^{(k+1)}, \mathbf{Z}, \mathbf{U}^{(k)}), \tag{11}$$

$$\mathbf{U}^{(k+1)} = \mathbf{U}^{(k)} + (\mathbf{\Omega}^{(k+1)} - \mathbf{Z}^{(k+1)}). \tag{12}$$

Within the $(k+1)$th iteration, separate updates can be performed for each $\mathbf{\Omega}_t$ as follows:

$$\mathbf{\Omega}_t^{(k+1)} = \underset{\mathbf{\Omega}_t\succ\mathbf{0}}{\arg\min} -\log(\det(\mathbf{\Omega}_t)) + \mathrm{tr}(\mathbf{\Omega}_t \hat{\mathbf{\Sigma}}_t) + \frac{1}{2\mu}\|\mathbf{\Omega}_t - \mathbf{A}\|_F, \tag{13}$$

with $\mathbf{A} = \frac{\mathbf{Z}_{0,t}^{(k)} + \mathbf{Z}_{1,t}^{(k)} + \mathbf{Z}_{2,t}^{(k)} - \mathbf{U}_{0,t}^{(k)} - \mathbf{U}_{1,t}^{(k)} - \mathbf{Z}_{2,t}^{(k)}}{3}$, and $\mu = \frac{M_t}{3\rho}$. From [18], one can see that:

$$\mathbf{\Omega}_t^{(k+1)} := \frac{\mu}{2}\boldsymbol{\mathcal{Q}}(\mathbf{D} + \sqrt{\mathbf{D}^2 + 4\mu^{-1}\mathbf{I}})\boldsymbol{\mathcal{Q}}^{\mathrm{T}}, \tag{14}$$

where $\boldsymbol{\mathcal{Q}}\mathbf{D}\boldsymbol{\mathcal{Q}}^{\mathrm{T}}$ is the eigen value decomposition of $\frac{\mathbf{A}+\mathbf{A}^{\mathrm{T}}}{2\mu} - \hat{\mathbf{\Sigma}}_t$.

Updating $\mathbf{Z}_0$ can be accomplished by updating each $\mathbf{Z}_{0,t}$ separately as follows:

$$\mathbf{Z}_{0,t}^{(k+1)} = \underset{Z_{0,t}\in\mathcal{V}_p}{\arg\min} \sum_{i\neq j} \omega_{t,ij}^{(l+1)} |[\mathbf{Z}_{0,t}]_{ij}| + \frac{\rho}{2}\|\mathbf{\Omega}_t^{(k+1)} - \mathbf{Z}_{0,t} + \mathbf{U}_{0,t}^{(k)}\|_F. \tag{15}$$

The constraint can be treated as a simple mapping function to a non-positive domain for off-diagonal elements. Therefore, the solution comes as follows:

$$[\mathbf{Z}_{0,t}^{(k+1)}]_{ij} = \begin{cases} [\mathbf{\Omega}_t^{(k+1)} + \mathbf{U}_{0,t}^{(k)}]_{ij}, & i = j \\ \min(0, \mathrm{Shrink}([\mathbf{\Omega}_t^{(k+1)} + \mathbf{U}_{0,t}^{(k)}]_{ij}, \frac{\omega_{t,ij}}{\rho})), & i \neq j \end{cases} \tag{16}$$

where $\mathrm{Shrink}(u,\alpha) = \frac{u}{|u|}\max\{0, |u| - \alpha\}$.

For $\mathbf{Z}_1$ and $\mathbf{Z}_2$, the update steps can then be separated into $\mathbf{Z}_{1,t-1}$ and $\mathbf{Z}_{2,t}$ $\forall t \in \{2,\cdots,T\}$ as follows:

$$\begin{bmatrix} \mathbf{Z}_{1,t-1}^{(k+1)} \\ \mathbf{Z}_{2,t}^{(k+1)} \end{bmatrix} = \underset{\mathbf{Z}_{1,t-1}, \mathbf{Z}_{2,t}}{\arg\min} \sum_{i=1}^{T}\sum_{j=1}^{T} \eta_{t,ij}^{(l+1)} |[[\mathbf{Z}_{2,t} - \mathbf{Z}_{1,t-1}]_{ij}| + \|\mathbf{\Omega}_{t-1}^{(k+1)} - \mathbf{Z}_{1,t-1} + \mathbf{U}_{1,t-1}^{(k)}\|_F + \|\mathbf{\Omega}_t^{(k+1)} - \mathbf{Z}_{2,t} + \mathbf{U}_{2,t}^{(k)}\|_F. \tag{17}$$

As it can be seen in [25], the above step can be simplified to:

$$\begin{bmatrix} \mathbf{Z}_{1,t-1}^{(k+1)} \\ \mathbf{Z}_{2,t}^{(k+1)} \end{bmatrix} = \frac{1}{2}\begin{bmatrix} \mathbf{\Omega}_t^{(k+1)} + \mathbf{U}_{2,t}^{(k)} + \mathbf{\Omega}_{t-1}^{(k+1)} + \mathbf{U}_{1,t-1}^{(k)} \\ \mathbf{\Omega}_t^{(k+1)} + \mathbf{U}_{2,t}^{(k)} + \mathbf{\Omega}_{t-1}^{(k+1)} + \mathbf{U}_{1,t-1}^{(k)} \end{bmatrix} + \frac{1}{2}\begin{bmatrix} -\mathbf{E} \\ \mathbf{E} \end{bmatrix}, \tag{18}$$

where $[\mathbf{E}]_{ij} = \mathrm{Shrink}([\mathbf{\Omega}_t^{(k+1)} - \mathbf{\Omega}_{t-1}^{(k+1)} + \mathbf{U}_{2,t}^{(k)} - \mathbf{U}_{1,t-1}^{(k)}]_{ij}, \frac{2\eta_{ij}}{\rho}), \forall (i,j) \in \{1,\cdots,P\} \times \{1,\cdots,P\}$.

Moreover, scaled dual variables can be updated as

$$\mathbf{U}_{0,t}^{(k+1)} = \mathbf{U}_{0,t}^{(k)} + (\mathbf{\Omega}^{(k+1)} - \mathbf{Z}_{0,t}^{(k+1)}), \forall t \in \{1, \cdots, T\} \quad (19)$$

$$\mathbf{U}_{1,t}^{(k+1)} = \mathbf{U}_{1,t}^{(k)} + (\mathbf{\Omega}^{(k+1)} - \mathbf{Z}_{1,t}^{(k+1)}), \forall t \in \{1, \cdots, T-1\} \quad (20)$$

$$\mathbf{U}_{2,t}^{(k+1)} = \mathbf{U}_{2,t}^{(k)} + (\mathbf{\Omega}^{(k+1)} - \mathbf{Z}_{2,t}^{(k+1)}). \forall t \in \{2, \cdots, T\} \quad (21)$$

Finally, a stopping criteria as in [18] can be adapted for the minimization step. It is worth noting that, the algorithm will be terminated in accordance with the stopping criteria specified in the MM framework [17].

IV. Simulation Results

In this section, we examine the efficacy of our proposed time-varying graph topology learning method using synthetic data. For comparative assessment, we utilized the time-varying graph learning (TVGL) method proposed in [10]. In all experiments, the optimal hyper-parameters values for each method were chosen through grid search.

A time-varying Erdős–Rényi graph is used as a data source for our experiments. As a starting point, we construct an Erdős–Rényi graph consisting 36 nodes with an edge probability $P_e = 0.06$. The edge weights are distributed uniformly in the interval $[2, 5]$. We assume that there are 100 time windows and the changes in the topology occur in time windows $11, 21, \cdots$, and 91. The changes are generated by re-sampling, and re-weighting 10% of the edges in $\mathbf{W}_{t-1}$. As a result, data point $\mathbf{x}_t^{(m)}$ is generated as follows: $\mathbf{x}_t^{(m)} \sim \mathcal{N}(0, \mathbf{L}_t^{\dagger} + \sigma_\epsilon^2 \mathbf{I})$, where $\sigma_\epsilon = 0.1$. For our experiments, we assume $M_t = 10$ data samples per time window. The results in Fig. 1 indicate that TGAM with SCAD as sparse and temporal penalties perform better than alternative methods for edge recovery, edge weight estimation, and capturing changes with a lower bias effect when transitions occur. Moreover, Fig. 1a demonstrates that SCAD is more effective in preserving similarity than FLasso. In Fig. 2, it is more evident that SCAD can act more selectively than l_1 to impose similarity between two time slots, and can alleviate the bias effect of Flasso.

In the second scenario, we compared the performance of algorithms against time-varying sparsity level. As a measure of performance, we considered the F-measure [10], which is a weighted average of precision and recall that reflects the accuracy of the estimated graph structure., and relative error [10], which indicates how accurate the edge weights of the estimated graph are. Under the same settings as in first scenario, we generated time-varying graphs for different $P_e \in \{0.06, 0.08, \cdots, 0.5\}$ representing various sparsity levels. The simulation results in terms of F-measure and relative error, obtained by averaging over 20 independent trails are plotted in Fig. 3 and Fig. 4. In terms of the F-measure, TVGM-SCAD-SCAD outperforms other methods both in sparse and non-sparse situations, as shown in Fig. 3. On the other hand, TGAM-SCAD-SCAD and TGAM-SCAD-FLasso provide better relative error than TVGL-FLasso in sparse enough situations, as shown in Fig. 4.

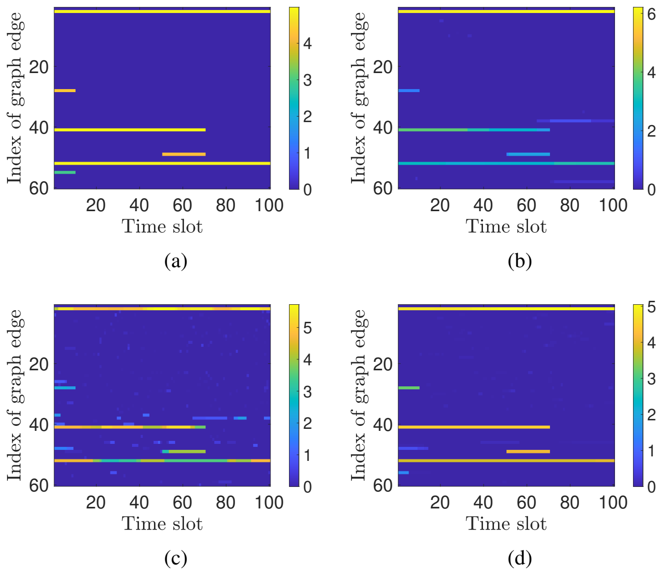

Fig. 1: (a) Ground truth, (b) TGL-FLasso, (c) TGAM-SCAD-FLasso, (d) TGAM-SCAD-SCAD.

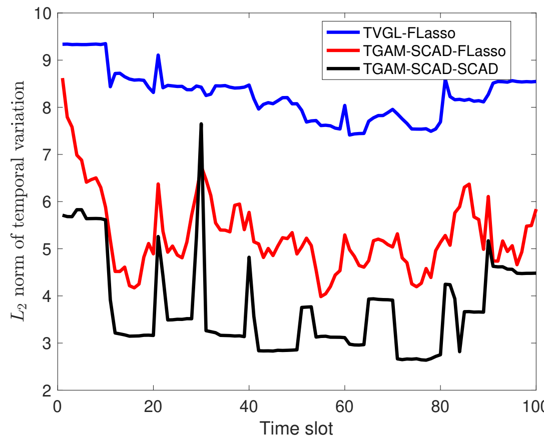

Fig. 2: There is a display of the temporal variation in the time-varying graph for TVGL-FLasso, TVGM-SCAD-FLasso and TVGM-SCAD-SCAD.

V. Conclusion

In this paper, we have presented a learning method for time-varying graph topology using log-likelihood formulations with non-convex penalties. To obtain Laplacian graphs, we imposed the total positivity constraint, and then extracted the Laplacian matrices from the diagonal of the precision matrices. For the first time, we adopted a non-convex and non-smooth penalty function, SCAD, as a temporal regulation function. SCAD penalty is demonstrated to be more effective in capturing abrupt changes and preserving similarity in time-varying graphs. Based on F-measure and relative error, our algorithm performed better than existing methods.

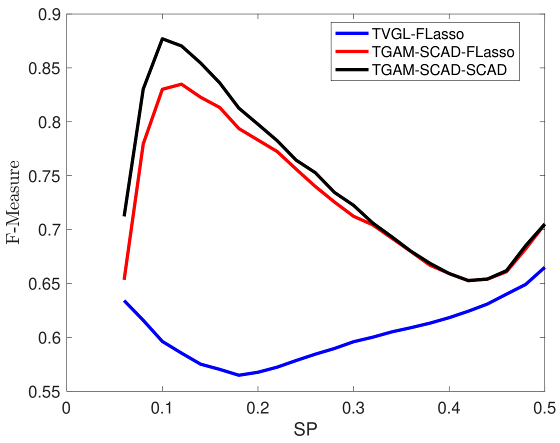

Fig. 3: F-measure of learning time-varying graphs for different levels of sparsity.

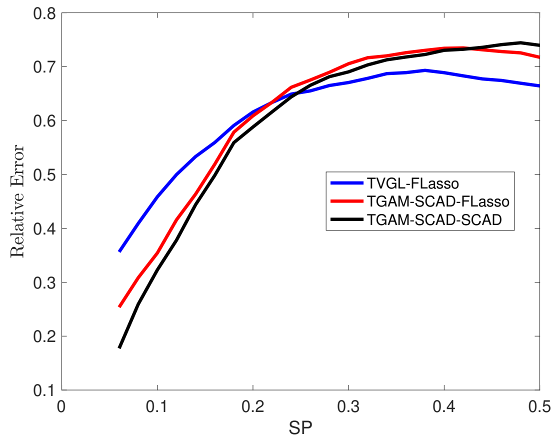

Fig. 4: Relative error of learning time-varying graphs for different levels of sparsity.

References

[1] A. Ahmed and E. P. Xing, "Recovering time-varying networks of dependencies in social and biological studies," *Proc. National Academy Sci.*, vol. 106, no. 29, pp. 11 878–11 883, July 2009.

[2] J. Richiardi, S. Achard, H. Bunke, and D. Van De Ville, "Machine learning with brain graphs: Predictive modeling approaches for functional imaging in systems neuroscience," *IEEE Signal Process. Mag.*, vol. 30, no. 3, pp. 58–70, Apr. 2013.

[3] N. Hu, D. Zhang, K. Xie, W. Liang, and M.-Y. Hsieh, "Graph learning-based spatial-temporal graph convolutional neural networks for traffic forecasting," *Connection Sci.*, pp. 1–20, July 2021.

[4] F. R. Rasim, S. M. Sattler *et al.*, "Analysis of electronic circuits with the signal flow graph method," *Circuits Syst.*, vol. 8, no. 11, p. 261, Nov. 2017.

[5] I. Jabłoński, "Graph signal processing in applications to sensor networks, smart grids, and smart cities," *IEEE Sensors J.*, vol. 17, no. 23, pp. 7659–7666, Dec. 2017.

[6] X. Dong, D. Thanou, M. Rabbat, and P. Frossard, "Learning graphs from data: A signal representation perspective," *IEEE Signal Process. Magaz.*, vol. 36, no. 3, pp. 44–63, May 2019.

[7] M. G. Preti, T. A. Bolton, and D. Van De Ville, "The dynamic functional connectome: State-of-the-art and perspectives," *Neuroimage*, vol. 160, pp. 41–54, Oct. 2017.

[8] J. V. de Miranda Cardoso and D. P. Palomar, "Learning undirected graphs in financial markets," in *Proc. 54th IEEE Int. Conf. Signals, Sys., Comput.*, 2020, pp. 741–745.

[9] L. Zhang, G. Yang, and B. C. Stadie, "World model as a graph: Learninglatent landmarks for planning," in *Int. Conf. Mach. Learn.*, 2021, pp. 12 611–12 620.

[10] K. Yamada, Y. Tanaka, and A. Ortega, "Time-varying graph learning with constraints on graph temporal variation," *arXiv preprint arXiv:2001.03346*, Jan. 2020.

[11] A. Natali, E. Isufi, M. Coutino, and G. Leus, "Learning time-varying graphs from online data," *arXiv preprint arXiv:2110.11017*, Oct. 2021.

[12] K. Yamada, Y. Tanaka, and A. Ortega, "Time-varying graph learning based on sparseness of temporal variation," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2019, pp. 5411–5415.

[13] X. Zhang and Q. Wang, "Time-varying graph learning under structured temporal priors," *arXiv preprint arXiv:2110.05018*, Feb. 2022.

[14] C.-H. Zhang, "Nearly unbiased variable selection under minimax concave penalty," *Annals stat.*, vol. 38, no. 2, pp. 894–942, Apr. 2010.

[15] J. Fan and R. Li, "Variable selection via nonconcave penalized likelihood and its oracle properties," *J. Amer. stat. Assoc.*, vol. 96, no. 456, pp. 1348–1360, Dec. 2001.

[16] J. Ying, J. V. de Miranda Cardoso, and D. Palomar, "Nonconvex sparse graph learning under Laplacian constrained graphical model," *Adv. Neural Inf. Process. Syst.*, vol. 33, Dec. 2020.

[17] Y. Sun, P. Babu, and D. P. Palomar, "Majorization-minimization algorithms in signal processing, communications, and machine learning," *IEEE Trans. on Signal Process.*, vol. 65, no. 3, pp. 794–816, Feb 2016.

[18] S. Boyd, N. Parikh, and E. Chu, *Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers*. Now Publishers Inc, 2011.

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

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

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

[22] J. K. Tugnait, "Sparse graph learning under Laplacian-related constraints," *IEEE Access*, vol. 9, pp. 151 067–151 079, Nov. 2021.

[23] Y. Zhang, K.-C. Toh, and D. Sun, "Learning graph Laplacian with MCP," *arXiv preprint arXiv:2010.11559*, Oct. 2020.

[24] R. Tibshirani, M. Saunders, S. Rosset, J. Zhu, and K. Knight, "Sparsity and smoothness via the fused lasso," *J. the Royal Stat. Soc.: Series B (Stat. Method.)*, vol. 67, no. 1, pp. 91–108, Dec. 2005.

[25] D. Hallac, Y. Park, S. Boyd, and J. Leskovec, "Network inference via the time-varying graphical lasso," in *Proc. 23th Int. Conf. Knowl. Discovery and Data Mining*, 2017, pp. 205–213.