Polynomial Graphical Lasso: Learning Edges From Gaussian Graph-Stationary Signals

Andrei Buciulea ⬤, *Member, IEEE*, Jiaxi Ying ⬤, *Member, IEEE*, Antonio G. Marques ⬤, *Senior Member, IEEE*, and Daniel P. Palomar ⬤, *Fellow, IEEE*

Abstract—**This paper introduces Polynomial Graphical Lasso (PGL), a new approach to learning graph structures from nodal signals. Our key contribution lies in modeling the signals as Gaussian and stationary on the graph, enabling the development of a graph learning formulation that combines the strengths of graphical lasso with a more encompassing model. Specifically, we assume that the precision matrix can take any polynomial form of the sought graph, allowing for increased flexibility in modeling nodal relationships. Given the inherent complexity and nonconvexity of the optimization problem, we (i) propose a low-complexity algorithm that alternates between estimating the graph and precision matrices, and (ii) characterize its convergence. We evaluate the performance of PGL through comprehensive numerical simulations using both synthetic and real data, demonstrating its superiority over several alternatives. Overall, this approach presents a significant advancement in graph learning and holds promise for various applications in graph-aware signal analysis and beyond.**

Index Terms—**Network-topology inference, graph learning, Gaussian signals, graph-stationary signals, covariance estimation.**

I. INTRODUCTION

MODERN datasets often exhibit irregular non-Euclidean support. In such scenarios, graphs have emerged as a pivotal tool, facilitating the generalization of classical information processing and structured learning techniques to irregular

Andrei Buciulea and Antonio G. Marques are with the Department of Signal Theory and Communications, King Juan Carlos University, 28933 Madrid, Spain (e-mail: andrei.buciulea@urjc.es; antonio.garcia.marques@urjc.es).

Jiaxi Ying is with the Department of Mathematics, Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong (e-mail: jx.ying@connect.ust.hk).

Daniel P. Palomar is with the Department of Electronic and Computer Engineering and the Department of Industrial Engineering and Decision Analytics, Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong (e-mail: palomar@ust.hk).

Digital Object Identifier 10.1109/TSP.2025.3544376

domains. Today, there is a wide range of applications that leverage graphs when processing, learning, and extracting knowledge from their associated datasets (see, e.g., problems in the context of electrical, communication, social, geographic, financial, genetic, and brain networks [2], [3], [4], [5], [6], to name a few). When using graphs to process structured non-Euclidean data, it is usually assumed that the underlying network topology is known. Unfortunately, this is not always the case. In many cases, the structure of the graph is not well defined, either because there is no underlying physical network or because the (best) metric to assess the level of association between the nodes is not known.

Since in most cases the existing relationships are not known beforehand, the standard approach is to infer the structure of the network from a set of available nodal observations/signals/features. To estimate the interactions between the existing nodes, the first step is to formally define the relationship between the topology of the graph and the properties of the signals defined on top of it. Early graph topology inference methods [7], [8] adopted a statistical approach, such as the correlation network [2, Ch. 7.3.1], partial correlations, or Gaussian Markov random fields (GMRF), with the latter leading to the celebrated graphical Lasso (GL) scheme [2], [9]. Partial correlation methods have been generalized to nonlinear settings [10]. Also in the nonlinear realm, less rigorous approaches simply postulate a similarity score, with links being drawn if their score exceeds a given threshold. In recent years, graph signal processing (GSP) based models [11], [12], [13], [14] have brought new ideas to the field, considering more complex relationships between the signals and the sought graph. These approaches have been generalized to deal with more complex scenarios that often arise in practice, such as the presence of hidden variables [15], [16], [17] or the simultaneous inference of multiple networks [18], [19].

The existing graph learning methods exhibit different advantages and limitations, with relevant tradeoffs including computational complexity, expressiveness, model accuracy, or sample complexity, among others. For instance, correlation networks need very few samples and can be run in parallel for each pair of nodes, but fail to capture intermediation nodal effects. On the other hand, GL (a maximum likelihood estimator for GMRF) can handle the intermediation effect while still requiring a relatively small number of samples compared to the size of the network. Some disadvantages of GL include assuming a

relatively simple signal model (failing to deal with, e.g., linear autoregressive network models) and forcing the learned graph structure to be a positive definite matrix. To overcome some of these issues, [13] proposed a more general model, Graph Stationary Recovery (GSR) approach, that assumed that the signals were stationary in the network or, equivalently, that the covariance matrix can be defined as a polynomial of the adjacency matrix of the graph [20]. Since GSR is a more general model, it is less restrictive in terms of the signals it can handle. However, it has the disadvantage of requiring a significantly larger number of observations than GL [13].

Our proposal is to combine the advantages of assuming Gaussianity, which implies solving a maximum likelihood problem that requires fewer node observations, with the larger generality of graph-stationary approaches. Our ultimate goal is to generalize the range of scenarios where GL can be used, while keeping the number of observations and computational complexity under control. To be more precise, we introduce Polynomial Graphical Lasso (PGL), a new scheme to learn graphs from signals that works under the assumption that the samples are Gaussian and graph-stationary, so that the covariance (precision) matrix of the observations can be written as a polynomial of a sparse graph. These assumptions give rise to a constrained log-likelihood minimization that is jointly optimized over the precision and adjacency matrices, with GL being a particular instance of our problem. The price to pay is that the postulated optimization, even after relaxing the sparsity constraints, is more challenging, leading to a biconvex problem. To mitigate this issue, we provide an efficient alternating algorithm with convergence guarantees.

Contributions. To summarize, our main contributions are:

- Introducing PGL, a novel graph learning scheme that, by assuming that the observations are Gaussian and graph-stationary, generalizes GL and is able to learn a meaningful graph structure in scenarios where the precision/covariance matrices are polynomials of the sparse matrix that represents the graph.
- Formulating the inference problem as a biconvex constrained optimization, with the variables to optimize being the precision matrix and the graph. While our focus is on learning graphs, note that this implies that PGL can also be used in the context of covariance estimation.
- Developing an efficient algorithm to solve the proposed optimization, together with convergence guarantees to a stationary point (block coordinatewise minimizer).
- Evaluating the performance of the proposed approach through comparisons with alternatives from the literature on synthetic and real-world datasets.

A preliminary version of this work was presented in a conference paper [1]. Compared to [1], this manuscript devotes more space to introducing the problem, clarifying the differences relative to the state of the art, and discussing pros and cons. More importantly, from a technical point of view, we propose a new approach to solve the graph learning optimization problem. The new algorithm is more efficient and requires less computational complexity, making it more suitable for handling graphs with a larger number of nodes. Equally relevant from

a contribution perspective, we provide a theoretical analysis of the algorithm's convergence. Additionally, the manuscript provides a more comprehensive numerical evaluation, incorporating both synthetic and real datasets, covering more scenarios, offering additional evaluation metrics, and comparing against more alternative approaches.

Outline. The remainder of the paper is organized as follows. Section II surveys the basic graph and GSP background, placing special emphasis on the concepts of graph stationarity and Gaussian graph signals. In Section III, the problem of learning (inferring) graphs from signals under different assumptions on the observations is formally stated. Section IV presents a computationally tractable relaxation of the graph learning problem, along with an efficient algorithm and its associated convergence guarantees. Section V quantifies and compares the recovery performance of the proposed approach with other methods from the literature using both synthetic and real data. Section VI closes the paper with concluding remarks.

II. GRAPHS AND RANDOM GRAPH SIGNALS

This section introduces the notation used in the paper and fundamental concepts of graphs and GSP that will help to explain the relationship between the available signals and the topology of the underlying graph.

Notation. We represent scalar, vectors and matrices using lowercase ($x \in \mathbb{R}$), bold lowercase ($\mathbf{x} \in \mathbb{R}^N$), and bold uppercase ($\mathbf{X} \in \mathbb{R}^{N \times R}$) letters, respectively. X_{ij} represents the values of the entry $[i, j]$ of matrix $\mathbf{X}$. $\mathbf{I}_N$ denotes the identity matrix of size $N \times N$. The expression $\text{diag}(\mathbf{x})$ represents a diagonal matrix with the values of $\mathbf{x}$ in the diagonal. The expressions $\| \cdot \|_0$ and $\| \cdot \|_1$ refer to the entry-wise (vectorized) ℓ_0 and ℓ_1 matrix norms, and $\| \cdot \|_F$ is the Frobenius norm.

Graphs. Let $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ be a weighted and undirected graph with N nodes, where $\mathcal{V}$ and $\mathcal{E}$ denote the vertex and edge set, respectively. The topology of the graph $\mathcal{G}$ is encoded in the sparse, weighted adjacency matrix $\mathbf{A} \in \mathbb{R}^{N \times N}$, where A_{ij} represents the weight of the edge between nodes j and i, and $A_{ij} = 0$ if i and j are not connected. A more general representation of the graph is the Graph Shift Operator (GSO), denoted by $\mathbf{S} \in \mathbb{R}^{N \times N}$, where $S_{ij} \neq 0$ if and only if $i = j$ or $(i, j) \in \mathcal{E}$. Common choices for the GSO are the adjacency matrix $\mathbf{A}$ [21], the combinatorial graph Laplacian $\mathbf{L} := \text{diag}(\mathbf{A1}) - \mathbf{A}$ [4], where $\text{diag}(\mathbf{A1})$ represents the degree matrix, containing the sum of edge weights connected to each node along its diagonal, and their degree-normalized variants. Since the graph is undirected, the GSO is symmetric and can be diagonalized as $\mathbf{S} = \mathbf{V} \mathbf{\Lambda} \mathbf{V}^\top$, where the orthogonal matrix $\mathbf{V} \in \mathbb{R}^{N \times N}$ contains the eigenvectors of the GSO, and the diagonal matrix $\mathbf{\Lambda}$ contains its eigenvalues.

Graph signals and filters. A graph signal is represented by a vector $\mathbf{x} \in \mathbb{R}^N$, where x_i denotes the signal value observed at node i. Alternatively, a graph signal can be understood as a $\mathcal{V} \to \mathbb{R}$ mapping, so that the signal values can represent features associated with the node at hand. Graph filters provide a flexible tool for either processing or modeling the relationship between a signal $\mathbf{x}$ and its underlying graph $\mathcal{G}$. Succinctly, a graph filter

is a linear operator $\mathbf{H} \in \mathbb{R}^{N \times N}$ that takes into account the topology of the graph and is defined as a polynomial of the GSO $\mathbf{S}$ of the form

$$\mathbf{H} = \sum_{l=0}^{L-1} h_l \mathbf{S}^l = \sum_{l=0}^{L-1} h_l \mathbf{V} \mathbf{\Lambda}^l \mathbf{V}^\top = \mathbf{V} \left(\sum_{l=0}^{L-1} h_l \mathbf{\Lambda}^l \right) \mathbf{V}^\top, \quad (1)$$

where $\{h_l\}_{l=0}^{L-1}$ represent the filter coefficients, and $\mathbf{V}$ and $\mathbf{\Lambda}$ are the eigenvectors and eigenvalues of the GSO respectively. Since $\mathbf{H}$ is a polynomial of $\mathbf{S}$, it readily follows that both matrices have the same eigenvectors $\mathbf{V}$.

Stationary graph signals. A graph signal $\mathbf{x}$ is defined to be stationary on the graph $\mathcal{G}$ if it can be expressed as the output of a graph filter $\mathbf{H}$ excited by a zero-mean white signal $\mathbf{w} \in \mathbb{R}^N$ [20], [22], [23]. In other words, if $\mathbf{w}$ has a covariance of $\mathbb{E}[\mathbf{w}\mathbf{w}^\top] = \mathbf{I}_N$, and $\mathbf{x} = \mathbf{H}\mathbf{w}$, then the covariance of $\mathbf{x}$ is $\mathbf{\Sigma} = \mathbb{E}[\mathbf{x}\mathbf{x}^\top] = \mathbf{H}\mathbb{E}[\mathbf{w}\mathbf{w}^\top]\mathbf{H}^\top = \mathbf{H}\mathbf{H}^\top = \mathbf{H}^2$. This reveals that if $\mathbf{x}$ is stationary on $\mathcal{G}$, then the covariance matrix $\mathbf{\Sigma}$ can be written as a polynomial of the GSO $\mathbf{S}$, and that, as a result, $\mathbf{\Sigma}$ and $\mathbf{S}$ have the same eigenvectors $\mathbf{V}$ [cf. (1)]. Hence, graph stationarity implies that the matrices $\mathbf{S}$ and $\mathbf{\Sigma}$ commute, which is a significant property to be exploited later on.

Gaussian graph signals. A Gaussian graph signal is an N-dimensional *random* vector $\mathbf{x}$ whose entries are jointly Gaussian. If $\mathbf{x} \in \mathbb{R}^N$ follows a multivariate normal distribution $\mathcal{N}(\mathbf{0}, \mathbf{\Sigma})$, then $\mathbf{\Sigma}$ is the covariance matrix, and $\mathbf{\Theta} = \mathbf{\Sigma}^{-1}$ is the precision matrix. The probability density function (PDF) of $\mathbf{x}$ is given by:

$$f_{\mathbf{\Theta}}(\mathbf{x}) = (2\pi)^{-N/2} \cdot \det(\mathbf{\Theta})^{\frac{1}{2}} \cdot \exp\left(-\frac{1}{2} \mathbf{x}^T \mathbf{\Theta} \mathbf{x} \right). \quad (2)$$

Here, $\mathbf{\Theta} \in \mathbb{R}^{N \times N}$ captures the conditional dependency structure among the graph's variables. [7].

Suppose now that we have a collection of R Gaussian signals $\mathbf{X} = [\mathbf{x}_1, ..., \mathbf{x}_R] \in \mathbb{R}^{N \times R}$ each of them independently drawn from the distribution in (2). The log-likelihood associated with $\mathbf{X}$ is

$$L(\mathbf{X}|\mathbf{\Theta}) = \prod_{r=1}^{R} f_{\mathbf{\Theta}}(\mathbf{x}_r), \quad \mathcal{L}(\mathbf{X}|\mathbf{\Theta}) = \sum_{r=1}^{R} \log(f_{\mathbf{\Theta}}(\mathbf{x}_r)). \quad (3)$$

This expression will be exploited when formulating our proposed inference approach and establishing links with classical methods.

III. GRAPH LEARNING PROBLEM FORMULATION

This section begins with a formal definition of the graph learning problem, followed by an explanation of some common approaches used in the literature to tackle this problem. Afterwards, we proceed to formalize the learning problem we aim to address and cast it as an optimization problem. We then provide an overview of the key features of our formulation and conduct a comparative analysis with the two closest approaches available in the literature.

To formally state the graph learning problem, let us recall that we assume: i) $\mathcal{G}$ is an undirected graph with N nodes,

ii) there is a random process associated with $\mathcal{G}$, and iii) we denote by $\mathbf{X} = [\mathbf{x}_1, ..., \mathbf{x}_R] \in \mathbb{R}^{N \times R}$ a collection of R independent realizations of such a process. The goal in graph learning is to use a given set of nodal observations to find the (estimates of the) links/associations between the nodes in the graph (i.e., use $\mathbf{X}$ to estimate the $\mathbf{S}$ associated with $\mathcal{G}$).

This problem has been addressed under different approaches [20], [24], [25]. Differences among these models typically arise from the underlying assumptions that are made regarding i) the graph, which almost universally entails just considering that the graph is sparse (see, e.g., [26] for a recent exception), ii) the signals, which assume certain properties related to the nature of the signals such as smoothness [25] or Gaussianity, [24] among others, and iii) the relationship between the graph and the signal, such as the stationarity property [20].

The model we propose aims to incorporate several assumptions about both the graph and the graph signals. To that end, we propose an approach for which we assume that: i) the graph is sparse, ii) the signals are Gaussian and, iii) the signals are stationary in the underlying graph.

Having established these assumptions, we now proceed to formally state our graph learning problem as follows

Problem 1: Given a set of signals $\mathbf{X} \in \mathbb{R}^{N \times R}$, find the underlying sparse graph structure encoded in $\mathbf{S}$ under the assumptions:

(AS1): The graph signals $\mathbf{X}$ are i.i.d. realizations of $\mathcal{N}(\mathbf{0}, \mathbf{\Sigma})$.

(AS2): The graph signals $\mathbf{X}$ are stationary in $\mathbf{S}$.

Our approach is to recast Problem 1 as the following optimization

$$\underset{\mathbf{\Theta} \succeq 0, \mathbf{S} \in \mathcal{S}}{\text{minimize}} \ -\log(\det(\mathbf{\Theta})) + \text{tr}(\hat{\mathbf{\Sigma}}\mathbf{\Theta}) + \rho\|\mathbf{S}\|_0,$$

$$\text{subject to} \ \ \mathbf{\Theta}\mathbf{S} = \mathbf{S}\mathbf{\Theta}, \quad (4)$$

where ρ is a hyperparameter that controls the sparsity of the graph, $\mathcal{S}$ is a generic set representing additional constraints that $\mathbf{S}$ is known to satisfy (e.g., the GSO being symmetric, its entries being between zero and one, and every node having at least one neighbor). The minimization takes as input the sample covariance matrix $\hat{\mathbf{\Sigma}} = \frac{1}{R}\mathbf{X}\mathbf{X}^\top$ and generates as output the estimate for $\mathbf{S}$ and, as a byproduct, the estimate for $\mathbf{\Theta}$. Moreover, for the problem in (4), we require $\mathbf{\Theta}$ to be positive semidefinite. This constraint arises because $\mathbf{\Theta}$ is the inverse of $\hat{\mathbf{\Sigma}}$ which is symmetric and positive definite by construction. Consequently, $\mathbf{\Theta}$ inherits the properties of being symmetric and positive semidefinite.

Next, we explain the motivation for each term in (4) with special emphasis on the constraint $\mathbf{\Theta}\mathbf{S} = \mathbf{S}\mathbf{\Theta}$, which is a fundamental component of our approach.

- The first two terms in the objective function are due to (AS1) and arise from minimizing the negative log-likelihood expression in (3). Indeed, it is clear that substituting (2) into (3) yields

$$\sum_{r=1}^{R} \left(-\frac{N}{2}\log(2\pi) - \log(\det(\mathbf{\Theta})) + \text{tr}(\mathbf{x}_r^T\mathbf{\Theta}\mathbf{x}_r) \right).$$

Since constants are irrelevant for the optimization, we drop the first term and divide the other two by R, yielding

$$
-\log(\det(\boldsymbol{\Theta})) + \frac{1}{R}\sum_{r=1}^{R} \mathrm{tr}(\mathbf{x}_r \mathbf{x}_r^T \boldsymbol{\Theta})
$$
$$
= -\log(\det(\boldsymbol{\Theta})) + \mathrm{tr}(\hat{\boldsymbol{\Sigma}}\boldsymbol{\Theta}). \qquad (5)
$$

- The equality constraint serves to embody (AS2). It is important to highlight that the polynomial relationship between $\boldsymbol{\Sigma}$ and $\mathbf{S}$, as implied by (AS2), can be addressed in estimation and optimization problems through the following approaches: i) extracting the eigenvectors of $\boldsymbol{\Sigma}$ and enforcing them to align with the eigenvectors of $\mathbf{S}$ [13], a method that typically requires manifold-type optimization; ii) estimating higher-order polynomials in $\mathbf{S}$, which is inherently challenging due to the nonlinearity and nonconvexity introduced by the powers of $\mathbf{S}$; or iii) imposing the constraint $\boldsymbol{\Sigma}\mathbf{S} = \mathbf{S}\boldsymbol{\Sigma}$ [17]. In contrast, our approach encodes the polynomial relation implied by (AS2) by enforcing commutativity between $\boldsymbol{\Theta}$ and $\mathbf{S}$. Note that if $\boldsymbol{\Sigma}$ and $\mathbf{S}$ are full-rank matrices and commute, it follows that $\boldsymbol{\Theta}$ and $\mathbf{S}$ also commute. In other words, $\boldsymbol{\Theta} = \boldsymbol{\Sigma}^{-1}$ can be represented as a polynomial in $\mathbf{S}$, which can be verified by the Cayley-Hamilton theorem. Since the proposed commutativity constraint is bilinear, the problem exhibits a scaling ambiguity that can be trivially addressed by, e.g., normalizing one or several entries of the matrices involved (see the algorithmic section for details).
- Finally, the term $\rho\|\mathbf{S}\|_0$ accounts for the fact of $\mathbf{S}$ being a GSO (hence, sparse), with $\rho > 0$ being a regularization parameter that determines the desired level of sparsity in the graph among all feasible graphs (i.e., $\mathbf{S}$ satisfying the commutativity constraints and those in the set $\mathcal{S}$). For $\rho \gg 0$, the regularization term $\|\mathbf{S}\|_0$ dominates, yielding the sparsest graph allowed by the constraints (with those in $\mathcal{S}$ preventing the trivial all zero solution). For the extreme case of $\rho = 0$, no explicit sparsity is enforced, so that the optimal graph/s $\mathbf{S}$ would be fully characterized by the constraints. As will be apparent in the following sections, the role of ρ becomes particularly relevant when the commutativity constraint is relaxed and the problem considers a larger set of feasible graphs.

It is important to note that the assumption of stationarity may seem stringent since it implies commutativity between $\mathbf{S}$ and $\boldsymbol{\Theta}$. However, it provides more degrees of freedom than many existing methods [24], [25], [27]. For example, in partial correlation methods [2, Ch. 7.3.1], $\mathbf{S}$ is restricted to be $\mathbf{S} = \boldsymbol{\Theta}$, while in our case, assuming stationarity allows $\boldsymbol{\Theta}$ to be *any* polynomial in $\mathbf{S}$. This leads to a more general approach, including partial correlation as a particular case. To further illustrate this point, consider the sparse structural equation model $\mathbf{X} = \mathbf{A}\mathbf{X} + \mathbf{W}$, with $\mathbf{W}$ being white noise. GL will identify $\mathbf{S} = \boldsymbol{\Theta} = (\mathbf{I}_N - \mathbf{A})^2$, while PGL will identify $\mathbf{S} = \mathbf{A}$.

To better understand the features of PGL, let us briefly discuss the main differences relative to its two closest competitors: GSR and GL.

GSR handles Problem 1 without considering the Gaussian assumption in (AS1). As a result, the first two terms in (4), which are associated with the log-likelihood function, are not present. This reduces the problem to inferring a sparse graph under the stationarity constraint. The stationarity assumption is incorporated into (4) through the expression $\boldsymbol{\Theta}\mathbf{S} = \mathbf{S}\boldsymbol{\Theta}$, which is equivalent to $\boldsymbol{\Sigma}\mathbf{S} = \mathbf{S}\boldsymbol{\Sigma}$ if $\boldsymbol{\Sigma}$ is a full-rank matrix. This property enables learning the graph by solving the following optimization problem with the commutativity constraint between $\hat{\boldsymbol{\Sigma}}$ and $\mathbf{S}$:

$$
\underset{\mathbf{S} \in \mathcal{S}}{\text{minimize}} \quad \rho\|\mathbf{S}\|_0 \quad \text{subject to} \quad \hat{\boldsymbol{\Sigma}}\mathbf{S} = \mathbf{S}\hat{\boldsymbol{\Sigma}}, \qquad (6)
$$

where the constraint is typically relaxed as $\|\hat{\boldsymbol{\Sigma}}\mathbf{S} - \mathbf{S}\hat{\boldsymbol{\Sigma}}\|_F^2 \leq \epsilon$ to account for the fact that we have $\hat{\boldsymbol{\Sigma}} \approx \boldsymbol{\Sigma}$. By assuming stationarity, the formulation in (6) allows the sample covariance to be modeled as any polynomial in $\mathbf{S}$, making it more general than the formulation in (4). However, the absence of Gaussianity in (6) means that it is no longer a maximum likelihood estimation. As a result, correctly identifying the ground truth $\mathbf{S}$ requires very reliable estimates of $\hat{\boldsymbol{\Sigma}}$, which usually entails having access to a large number of signals to set ϵ close to zero. This is indeed a challenge, especially in setups with a large number of nodes.

For the second scenario, suppose we simplify (AS2) and instead of considering $\boldsymbol{\Theta}$ as any polynomial in $\mathbf{S}$, we restrict it to a particular case with the following structure: $\boldsymbol{\Theta} = \boldsymbol{\Sigma}^{-1} = \sigma\mathbf{I} + \delta\mathbf{S}$. Then, up to the diagonal values and scaling issues, the sparse matrix $\mathbf{S}$ to be estimated and $\boldsymbol{\Theta} = \boldsymbol{\Sigma}^{-1}$ are the same. Consequently, it suffices to optimize over one of them, leading to the well-known GL formulation:

$$
\underset{\boldsymbol{\Theta} \succeq 0, \boldsymbol{\Theta} \in \mathcal{S}_{\boldsymbol{\Theta}}}{\text{minimize}} \quad -\log(\det(\boldsymbol{\Theta})) + \mathrm{tr}(\hat{\boldsymbol{\Sigma}}\boldsymbol{\Theta}) + \rho\|\boldsymbol{\Theta}\|_0, \qquad (7)
$$

where $\mathcal{S}_{\boldsymbol{\Theta}}$ denotes the set of structural constraints imposed on the precision matrix $\boldsymbol{\Theta}$. In this context, $\mathcal{S}_{\boldsymbol{\Theta}}$ ensures that $\boldsymbol{\Theta}$ is symmetric and positive semidefinite, formally defined as $\mathcal{S}_{\boldsymbol{\Theta}} := \{\boldsymbol{\Theta} \in \mathbb{R}^{N \times N} | \boldsymbol{\Theta} = \boldsymbol{\Theta}^T; \ \mathbf{S} \succeq 0\}$. The main advantages of (7) relative to (4) are that the number of variables is smaller and the resulting problem (after relaxing the ℓ_0 norm) is convex. The main drawback is that by forcing the support of $\mathbf{S}$ and $\boldsymbol{\Theta}$ to be the same, the set of feasible graphs (and their topological properties) is more limited. Indeed, GL can only estimate graphs that are positive definite, while the problem in (4) can yield any symmetric matrix. Remarkably, when the model assumed in (7) holds true (i.e., data is Gaussian and $\boldsymbol{\Theta}$ is sparse), GL is able to find reliable estimates of $\mathbf{S}$ even when the number of samples R is fairly low. On the other hand, simulations will show that GL does a poor job estimating $\mathbf{S}$ when the relation between the precision matrix and $\mathcal{G}$ is more involved.

In conclusion, from a conceptual point of view, our formulation reaches a favorable balance between GL and graph-stationarity approaches. This leads to the following two main advantages i) a more general model than GL since our approach models $\boldsymbol{\Theta}$ as any polynomial in $\mathbf{S}$ and ii) a model with more structure than the graph-stationarity approaches due to the incorporation of (AS1). However, it is important to note that the optimization in (4), even if the ℓ_0 norm is relaxed,

lacks convexity due to the presence of a bilinear constraint that couples the optimization variables $\mathbf{\Theta}$ and $\mathbf{S}$. These challenges will be addressed in the subsequent section.

IV. BICONVEX RELATION AND ALGORITHM DESIGN

As explained in the previous section, the problem in (4) is not convex and this challenges designing an algorithm to find a good solution. This section reformulates (4), develops an iterative algorithm, referred to as PGL, to estimate $\mathbf{S}$ and $\mathbf{\Theta}$, and characterizes its convergence to a coordinate-wise minimum point. The proposed approach involves several modifications, with the two most important ones being: i) we replace the ℓ_0-norm with an elastic net regularizer, which is convex [28], [29]; and ii) we relax the commutativity constraint using an inequality instead of an equality. Next, we explain step by step the resulting formulation.

A. Biconvex Relaxation

The first modification to reformulate (4) is to relax the constraint that imposes commutativity between $\mathbf{S}$ and $\mathbf{\Theta}$. Such a constraint is stringent and significantly narrows the feasible solution set of (4), which may not be practical in real-world scenarios. Furthermore, considering that our access to the covariance (or precision) matrix is limited to its sampled estimates, enforcing exact commutativity is excessively restrictive. To mitigate this, we relax the original constraint by replacing the matrix equality $\mathbf{\Theta S} = \mathbf{S\Theta}$ with the scalar Frobenius norm-based inequality $\|\mathbf{\Theta S} - \mathbf{S\Theta}\|_F^2 \leq \delta$. This modification not only expands the feasible region but also endows the model with a greater degree of robustness.

The second modification addresses the non-convexity of the objective in (4), which originates from the use of the ℓ_0-norm. To alleviate this issue, we relax the problem using an elastic net regularizer. Specifically, we replace the $\rho\|\mathbf{S}\|_0$ penalty with $\rho\big(\|\mathbf{S}\|_1 + \frac{\eta}{2\rho}\|\mathbf{S}\|_F^2\big)$, where the parameter η controls the trade-off between the ℓ_1-norm and the Frobenius norm components, and is typically set to a very small value. Although elastic net regularizers have demonstrated practical effectiveness, alternative methods for relaxing the ℓ_0-norm exist (see, for example, [30], [31]), each offering distinct trade-offs in computational complexity, convergence speed, and theoretical underpinnings.

With the incorporation of these two modifications, we reformulate the original graph learning problem presented in (4) as follows

$$\underset{\mathbf{\Theta}\succeq 0, \mathbf{S}\in\mathcal{S}}{\text{minimize}} \quad -\log(\det(\mathbf{\Theta})) + \text{tr}(\hat{\mathbf{\Sigma}}\mathbf{\Theta}) + \rho\|\mathbf{S}\|_1 + \frac{\eta}{2}\|\mathbf{S}\|_F^2,$$
$$\text{subject to} \quad \|\mathbf{\Theta S} - \mathbf{S\Theta}\|_F \leq \delta. \tag{8}$$

In this setup, δ serves as a hyperparameter chosen according to the quality of the estimation of $\hat{\mathbf{\Sigma}}$ which affects the estimation of $\mathbf{\Theta}$. A smaller value of δ is appropriate when the quality of $\hat{\mathbf{\Sigma}}$ is high, which typically corresponds to having a sample size R that is substantially larger than the number of nodes. Similarly, ρ and η are chosen to control the trade-offs in the structure of $\mathbf{S}$ where ρ governs sparsity, with higher values enforcing fewer edges,

while η balances sparsity with control over the overall magnitude of $\mathbf{S}$ through its Frobenius norm. While the relaxation of the commutativity constraint enhances the robustness of our formulation to data quality and simplifies the optimization by reducing the number of Lagrange multipliers, the product of $\mathbf{\Theta}$ and $\mathbf{S}$ still introduces nonconvexity into the problem. The way we propose for dealing with the (updated) biconvex constraint is to solve (8) using an alternating optimization algorithm. This family of algorithms is widely used to approximate nonconvex problems by dividing the original problem into several convex subproblems and solving them with respect to each of the variables by fixing all the others. In our particular case, this methodology involves alternately solving for $\mathbf{\Theta}$ with $\mathbf{S}$ held fixed, and then updating $\mathbf{S}$ using the newly updated $\mathbf{\Theta}$, at each iteration.

In the subsequent two subsections, we delve into the detailed methodologies employed to solve each of the two subproblems. Following this, we outline the overall algorithm and discuss its convergence properties. To simplify exposition, in the remainder of the section, we will assume that $\mathbf{S}$ represents the *adjacency* matrix of an undirected graph. Consequently, the feasible solution set for $\mathbf{S}$ is defined as:

$$\mathcal{S} := \big\{\mathbf{S}\in\mathbb{R}^{N\times N}|\mathbf{S} = \mathbf{S}^T; \; \mathbf{S}\geq 0; \; \text{diag}(\mathbf{S}) = \mathbf{0}; \; \mathbf{S1}\geq \mathbf{1}\big\}, \tag{9}$$

where $\mathbf{S}$ is constrained to be symmetric with zero diagonal entries and non-negative off-diagonal elements. The additional condition $\mathbf{S1} \geq \mathbf{1}$ is imposed to preclude the trivial solution, i.e., $\mathbf{S} = \mathbf{0}$. Nonetheless, the techniques presented next can be readily extended to accommodate different forms of $\mathbf{S}$.

B. Solving Subproblem for $\mathbf{S}$

We begin by addressing the subproblem with respect to $\mathbf{S}$, while holding $\mathbf{\Theta}$ fixed. The subproblem is formulated as:

$$\underset{\mathbf{S}\in\mathcal{S}}{\text{minimize}} \quad \rho\|\mathbf{S}\|_1 + \frac{\eta}{2}\|\mathbf{S}\|_F^2,$$
$$\text{subject to} \quad \|\mathbf{\Theta S} - \mathbf{S\Theta}\|_F \leq \delta. \tag{10}$$

To solve (10), we adopt a linearized ADMM approach [32], which introduces an auxiliary variable $\mathbf{P}$ and leads to the following equivalent formulation:

$$\underset{\mathbf{S}\in\mathcal{S}, \mathbf{P}}{\text{minimize}} \quad \rho\|\mathbf{S}\|_1 + \frac{\eta}{2}\|\mathbf{S}\|_F^2,$$
$$\text{subject to} \quad \mathbf{\Theta S} - \mathbf{S\Theta} = \mathbf{P}, \|\mathbf{P}\|_F \leq \delta. \tag{11}$$

The augmented Lagrangian associated with (11) is then given by

$$L(\mathbf{S}, \mathbf{P}, \mathbf{Z}) = \rho\|\mathbf{S}\|_1 + \frac{\eta}{2}\|\mathbf{S}\|_F^2 + \langle\mathbf{Z}, \mathbf{\Theta S} - \mathbf{S\Theta} - \mathbf{P}\rangle$$
$$+ \frac{\beta}{2}\|\mathbf{\Theta S} - \mathbf{S\Theta} - \mathbf{P}\|_F^2, \tag{12}$$

where $\mathbf{Z}$ is the Lagrange multiplier. Note that the term $\|\mathbf{P}\|_F \leq \delta$ was omitted in (12) because the optimization is performed over $\mathbf{S}$ first, for a fixed $\mathbf{P}$.

To update $\mathbf{S}$ at the t-th iteration, we address the following minimization problem:

$$\underset{\mathbf{S}\in\mathcal{S}}{\text{minimize}} \quad \rho\|\mathbf{S}\|_1 + \frac{\eta}{2}\|\mathbf{S}\|_F^2 + \frac{\beta}{2}\|\mathbf{\Theta}\mathbf{S} - \mathbf{S}\mathbf{\Theta} - \mathbf{P} + \frac{1}{\beta}\mathbf{Z}\|_F^2, \tag{13}$$

where, for the sake of simplicity, we omit the iteration subscript from $\mathbf{\Theta}^{(t)}$ and $\mathbf{Z}^{(t)}$. Problem (13) does not admit a closed-form solution due to the term $\frac{1}{2}\|\mathbf{\Theta}\mathbf{S} - \mathbf{S}\mathbf{\Theta} - \mathbf{P} + \frac{1}{\beta}\mathbf{Z}\|_F^2$. To deal conveniently with this problem we resort to the majorization-minimization (MM) algorithm [33]. We denote this term as $g(\mathbf{S})$ and proceed to majorize both $g(\mathbf{S})$ and $\frac{\eta}{2}\|\mathbf{S}\|_F^2$ at the point $\mathbf{S}^{(t)}$, resulting in the following problem:

$$\underset{\mathbf{S}\in\mathcal{S}}{\text{minimize}} \quad \langle\rho\mathbf{1}_{N\times N} + \eta\mathbf{S}^{(t)} + \beta\nabla g(\mathbf{S}^{(t)}), \mathbf{S} - \mathbf{S}^{(t)}\rangle$$
$$+ \frac{L_1}{2}\|\mathbf{S} - \mathbf{S}^{(t)}\|_F^2, \tag{14}$$

where $\mathbf{1}_{N\times N}$ represents an all ones square matrix of size N and $\nabla g(\mathbf{S})$ is the gradient of $g(\mathbf{S})$, detailed in the equation:

$$\nabla g(\mathbf{S}) = \mathbf{\Theta}\mathbf{\Theta}\mathbf{S} + \mathbf{S}\mathbf{\Theta}\mathbf{\Theta} - 2\mathbf{\Theta}\mathbf{S}\mathbf{\Theta} + \mathbf{P}\mathbf{\Theta}$$
$$- \mathbf{\Theta}\mathbf{P} + \frac{1}{\beta}(\mathbf{\Theta}\mathbf{Z} - \mathbf{Z}\mathbf{\Theta}). \tag{15}$$

Now, Problem (14) has a closed-form solution, allowing for the update of $\mathbf{S}^{(t+1)}$ as follows:

$$\mathbf{S}^{(t+1)} = \mathcal{P}_\mathcal{S}\Big(\mathbf{S}^{(t)} - \frac{1}{L_1}\big(\rho\mathbf{1}_{N\times N} + \eta\mathbf{S}^{(t)} + \beta\nabla g(\mathbf{S}^{(t)})\big)\Big), \tag{16}$$

where $\mathcal{P}_\mathcal{S}$ is the projection onto the set $\mathcal{S}$ with respect to the Frobenius norm, which can be computed efficiently by the Dykstra's projection algorithm [34]. More specifically, the set $\mathcal{S}$ can be written as the intersection of two closed convex sets as follows:

$$\mathcal{S} = \mathcal{S}_A \cap \mathcal{S}_B, \tag{17}$$

where $\mathcal{S}_A := \{\mathbf{S}\in\mathbb{R}^{N\times N} \,|\, \mathbf{S} = \mathbf{S}^T\}$ and $\mathcal{S}_B := \{\mathbf{S}\in\mathbb{R}^{N\times N} \,|\, \mathbf{S} \geq 0; \text{diag}(\mathbf{S}) = \mathbf{0}; \mathbf{S}\mathbf{1} \geq \mathbf{1}\}$. We employ Dykstra's projection algorithm [34] to compute the nearest point projection of a given point onto the intersection of sets $\mathcal{S}_A$ and $\mathcal{S}_B$. Dykstra's projection algorithm achieves this by alternately projecting the point onto $\mathcal{S}_A$ and $\mathcal{S}_B$ until the solution is reached. For a more comprehensive understanding of Dykstra's projection algorithm, the reader is directed to [34]. Detailed descriptions of the projection computations onto sets $\mathcal{S}_A$ and $\mathcal{S}_B$ are provided in Appendix A.

Returning to the augmented Lagrangian in (12), we update $\mathbf{P}$ at the t-th iteration by solving the following problem:

$$\underset{\mathbf{P}}{\text{minimize}} \quad \frac{\beta}{2}\|\mathbf{P} - \mathbf{\Theta}\mathbf{S} + \mathbf{S}\mathbf{\Theta} - \frac{1}{\beta}\mathbf{Z}\|_F^2,$$
$$\text{subject to} \quad \|\mathbf{P}\|_F \leq \delta, \tag{18}$$

where we have simplified the notation by omitting the iteration subscripts from $\mathbf{S}^{(t+1)}$ and $\mathbf{Z}^{(t)}$. Problem (18) has a closed-form solution. As a result, $\mathbf{P}^{(t+1)}$ can be updated by

$$\mathbf{P}^{(t+1)} = \mathcal{P}_\delta\Big(\mathbf{\Theta}\mathbf{S} - \mathbf{S}\mathbf{\Theta} + \frac{1}{\beta}\mathbf{Z}\Big), \tag{19}$$

Algorithm 1: Inner loop for S update.

Input: $\hat{\mathbf{\Theta}}^{(k)}$, $\hat{\mathbf{S}}^{(k)}$, $\hat{\mathbf{P}}^{(k)}$, $\hat{\mathbf{Z}}^{(k)}$, ρ, η, β, δ
Outputs: $\hat{\mathbf{S}}^{(k+1)}$, $\hat{\mathbf{P}}^{(k+1)}$, $\hat{\mathbf{Z}}^{(k+1)}$
1 Initialize $\mathbf{S}^{(0)} = \hat{\mathbf{S}}^{(k)}$, $\mathbf{P}^{(0)} = \mathbf{P}^{(k)}$, $\mathbf{Z}^{(0)} = \mathbf{Z}^{(k)}$
2 **for** $t = 0$ **to** $T - 1$ **do**
3 $\quad$ Update $\mathbf{S}^{(t+1)}$ by (16);
4 $\quad$ Update $\mathbf{P}^{(t+1)}$ by (19);
5 $\quad$ Update $\mathbf{Z}^{(t+1)}$ by (21);
6 **end**
7 $\hat{\mathbf{S}}^{(k+1)} = \mathbf{S}^{(T)}$, $\hat{\mathbf{P}}^{(k+1)} = \mathbf{P}^{(T)}$, $\hat{\mathbf{Z}}^{(k+1)} = \mathbf{Z}^{(T)}$.

---

where $\mathcal{P}_\delta$ denotes the projection defined by:

$$\mathcal{P}_\delta(\mathbf{A}) = \begin{cases} \frac{\delta}{\|\mathbf{A}\|_F}\mathbf{A} & \text{if } \|\mathbf{A}\|_F > \delta \\ \mathbf{A} & \text{otherwise.} \end{cases} \tag{20}$$

Finally, the dual variable $\mathbf{Z}$ is updated according to:

$$\mathbf{Z}^{(t+1)} = \mathbf{Z}^{(t)} + \beta(\mathbf{\Theta}\mathbf{S} - \mathbf{S}\mathbf{\Theta} - \mathbf{P}), \tag{21}$$

where the iteration subscripts from $\mathbf{S}^{(t+1)}$ and $\mathbf{P}^{(t+1)}$ have been omitted for simplicity. A pseudocode of the steps to be performed for the update of $\mathbf{S}$ is summarized in Algorithm 1.

If the parameter L_1 in (14) is larger than the Lipschitz constant of the gradient of $\beta g(\mathbf{S}) + \frac{\eta}{2}\|\mathbf{S}\|_F^2$, then the sequence $\{(\mathbf{S}^{(t)}, \mathbf{P}^{(t)})\}$ converges to the optimal solution of Problem (11), and $\{\mathbf{Z}^{(t)}\}$ converges to the optimal solution of the dual of problem (11), which follows from the existing convergence result of majorized ADMM [35]. To enhance empirical convergence rates, adopting a more proactive strategy for selecting the parameter L_1 is beneficial. For example, utilizing a back-tracking line search to determine the stepsize in (16) can help to accelerate convergence.

We note that the choice of the penalty parameter β can affect the convergence speed of the ADMM algorithm. A poorly chosen β may lead to very slow convergence in practice. Adaptive schemes that adjust β have been shown to often result in better practical performance. For example, we can adopt the adaptive update rule presented in [32]:

$$\beta^{(t+1)} = \begin{cases} \tau^{\text{inc}}\beta^{(t)}, & \text{if } \|\mathbf{r}^{(t)}\|_F > \mu\|\mathbf{s}^{(t)}\|_F, \\ \beta^{(t)}/\tau^{\text{dec}}, & \text{if } \|\mathbf{s}^{(t)}\|_F > \mu\|\mathbf{r}^{(t)}\|_F, \\ \beta^{(t)}, & \text{otherwise,} \end{cases} \tag{22}$$

where $\mu > 1$, $\tau^{\text{inc}} > 1$, and $\tau^{\text{dec}} > 1$ are predefined parameters. Here, $\mathbf{r}^{(t)}$ and $\mathbf{s}^{(t)}$ represent the primal and dual residuals at iteration t, respectively. They are defined as

$$\mathbf{r}^{(t)} = \mathbf{\Theta}\mathbf{S}^{(t)} - \mathbf{S}^{(t)}\mathbf{\Theta} - \mathbf{P}^{(t)},$$

and

$$\mathbf{s}^{(t)} = \beta^{(t)}\mathbf{\Theta}\big(\mathbf{P}^{(t)} - \mathbf{P}^{(t-1)}\big) - \beta^{(t)}\big(\mathbf{P}^{(t)} - \mathbf{P}^{(t-1)}\big)\mathbf{\Theta}.$$

Although it can be challenging to prove the convergence of ADMM when β varies by iteration, the convergence theory established for a fixed β remains applicable if one assumes that β becomes constant after a finite number of iterations.

C. Solving Subproblem for $\boldsymbol{\Theta}$

Using the formulation from (8), we now turn our attention to the subproblem for $\boldsymbol{\Theta}$

$$\begin{aligned} \underset{\boldsymbol{\Theta} \succeq 0}{\text{minimize}} \quad & -\log(\det(\boldsymbol{\Theta})) + \text{tr}(\hat{\boldsymbol{\Sigma}}\boldsymbol{\Theta}), \\ \text{subject to} \quad & \|\mathbf{S}\boldsymbol{\Theta} - \boldsymbol{\Theta}\mathbf{S}\|_F \le \delta. \end{aligned} \tag{23}$$

Similarly to the approach taken for the $\mathbf{S}$ subproblem, we reformulate the subproblem (23) for $\boldsymbol{\Theta}$ as follows

$$\begin{aligned} \underset{\boldsymbol{\Theta} \succeq 0, \, \mathbf{Q}}{\text{minimize}} \quad & -\log(\det(\boldsymbol{\Theta})) + \text{tr}(\hat{\boldsymbol{\Sigma}}\boldsymbol{\Theta}), \\ \text{subject to} \quad & \mathbf{S}\boldsymbol{\Theta} - \boldsymbol{\Theta}\mathbf{S} = \mathbf{Q}, \; \|\mathbf{Q}\|_F \le \delta. \end{aligned} \tag{24}$$

The augmented Lagrangian associated with (24) is given by

$$\begin{aligned} L(\boldsymbol{\Theta}, \mathbf{Q}, \mathbf{Y}) = & -\log(\det(\boldsymbol{\Theta})) + \text{tr}(\hat{\boldsymbol{\Sigma}}\boldsymbol{\Theta}) + \langle \mathbf{Y}, \, \mathbf{S}\boldsymbol{\Theta} - \boldsymbol{\Theta}\mathbf{S} - \mathbf{Q} \rangle \\ & + \frac{\beta}{2} \|\mathbf{S}\boldsymbol{\Theta} - \boldsymbol{\Theta}\mathbf{S} - \mathbf{Q}\|_F^2 \, . \end{aligned} \tag{25}$$

To update $\boldsymbol{\Theta}$ at the t-th iteration, we solve the following optimization problem

$$\begin{aligned} \underset{\boldsymbol{\Theta} \succeq 0}{\text{minimize}} \quad & -\log(\det(\boldsymbol{\Theta})) + \text{tr}(\hat{\boldsymbol{\Sigma}}\boldsymbol{\Theta}) \\ & + \frac{\beta}{2} \|\mathbf{S}\boldsymbol{\Theta} - \boldsymbol{\Theta}\mathbf{S} - \mathbf{Q} + \frac{1}{\beta}\mathbf{Y}\|_F^2, \end{aligned} \tag{26}$$

where we have omitted the iteration subscripts from $\mathbf{Q}^{(t)}$ and $\mathbf{Y}^{(t)}$ for simplicity. Note that in (26), the terms $\langle \mathbf{Y}, \mathbf{S}\boldsymbol{\Theta} - \boldsymbol{\Theta}\mathbf{S} - \mathbf{Q} \rangle$ and $\frac{\beta}{2}\|\mathbf{S}\boldsymbol{\Theta} - \boldsymbol{\Theta}\mathbf{S} - \mathbf{Q}\|_F^2$ have been combined into a single quadratic term $\frac{\beta}{2}\|\mathbf{S}\boldsymbol{\Theta} - \boldsymbol{\Theta}\mathbf{S} - \mathbf{Q} + \frac{1}{\beta}\mathbf{Y}\|_F^2$, which is equivalent when minimizing with respect to $\boldsymbol{\Theta}$.

Let $f(\boldsymbol{\Theta}) = \frac{1}{2}\|\mathbf{S}\boldsymbol{\Theta} - \boldsymbol{\Theta}\mathbf{S} - \mathbf{Q} + \frac{1}{\beta}\mathbf{Y}\|_F^2$. We then construct the majorizer of the objective function in (26) at the point $\boldsymbol{\Theta}^{(t)}$ and obtain

$$\begin{aligned} \underset{\boldsymbol{\Theta} \succeq 0}{\text{minimize}} \quad & -\log(\det(\boldsymbol{\Theta})) + \langle \beta\nabla f(\boldsymbol{\Theta}^{(t)}) + \hat{\boldsymbol{\Sigma}}, \boldsymbol{\Theta} - \boldsymbol{\Theta}^{(t)} \rangle \\ & + \frac{L_2}{2} \|\boldsymbol{\Theta} - \boldsymbol{\Theta}^{(t)}\|_F^2, \end{aligned} \tag{27}$$

where $\nabla f(\boldsymbol{\Theta})$ denotes the gradient of $f(\boldsymbol{\Theta})$

$$\begin{aligned} \nabla f(\boldsymbol{\Theta}) = & \, \mathbf{S}\mathbf{S}\boldsymbol{\Theta} + \boldsymbol{\Theta}\mathbf{S}\mathbf{S} - 2\mathbf{S}\boldsymbol{\Theta}\mathbf{S} \\ & + \mathbf{Q}\mathbf{S} - \mathbf{S}\mathbf{Q} + \frac{1}{\beta}(\mathbf{S}\mathbf{Y} - \mathbf{Y}\mathbf{S}). \end{aligned} \tag{28}$$

Lemma 1: The optimal solution of problem (27) is [36]

$$\boldsymbol{\Theta}^{(t+1)} = \mathbf{U} \left(\frac{\boldsymbol{\Lambda} + \sqrt{\boldsymbol{\Lambda}^2 + \frac{4}{L_2}}}{2} \right) \mathbf{U}^\top, \tag{29}$$

where $\boldsymbol{\Lambda}$ and $\mathbf{U}$ contain the eigenvalues and eigenvectors of $\boldsymbol{\Theta}^{(t)} - \frac{1}{L_2}\left(\beta\nabla f(\boldsymbol{\Theta}^{(t)}) + \hat{\boldsymbol{\Sigma}}\right)$, respectively.

Then, $\mathbf{Q}$ is updated by addressing the following problem

$$\begin{aligned} \underset{\mathbf{Q}}{\text{minimize}} \quad & \frac{\beta}{2}\|\mathbf{Q} - \mathbf{S}\boldsymbol{\Theta} + \boldsymbol{\Theta}\mathbf{S} - \frac{1}{\beta}\mathbf{Y}\|_F^2, \\ \text{subject to} \quad & \|\mathbf{Q}\|_F \le \delta, \end{aligned} \tag{30}$$

Algorithm 2: Inner loop for $\boldsymbol{\Theta}$ update.

Input: $\boldsymbol{\Sigma}, \hat{\boldsymbol{\Theta}}^{(k)}, \hat{\mathbf{S}}^{(k+1)}, \hat{\mathbf{Q}}^{(k)}, \hat{\mathbf{Y}}^{(k)}, \beta, \delta$
Outputs: $\hat{\boldsymbol{\Theta}}^{(k+1)}, \hat{\mathbf{Q}}^{(k+1)}, \hat{\mathbf{Y}}^{(k+1)}$

1 Initialize $\boldsymbol{\Theta}^{(0)} = \hat{\boldsymbol{\Theta}}^{(k)}, \mathbf{Q}^{(0)} = \hat{\mathbf{Q}}^{(k)}, \mathbf{Y}^{(0)} = \hat{\mathbf{Y}}^{(k)}$
2 **for** $t = 0$ **to** $T - 1$ **do**
3   Update $\boldsymbol{\Theta}^{(t+1)}$ by (29);
4   Update $\mathbf{Q}^{(t+1)}$ by (31);
5   Update $\mathbf{Y}^{(t+1)}$ by (32);
6 **end**
7 $\hat{\boldsymbol{\Theta}}^{(k+1)} = \boldsymbol{\Theta}^{(T)}, \hat{\mathbf{Q}}^{(k+1)} = \mathbf{Q}^{(T)}, \hat{\mathbf{Y}}^{(k+1)} = \mathbf{Y}^{(T)}$.

Algorithm 3: Polynomial Graphical Lasso (PGL)

Input: $\hat{\boldsymbol{\Sigma}}, \rho, \eta, \beta,$ and δ
Outputs: $\hat{\boldsymbol{\Theta}}$ and $\hat{\mathbf{S}}$

1 Initialize $\hat{\boldsymbol{\Theta}}^{(0)} = \hat{\boldsymbol{\Sigma}}^{-1}$
2 Initialize $\hat{\mathbf{S}}^{(0)}$ by solving (6)
3 Initialize $\mathbf{P}^{(0)}, \mathbf{Q}^{(0)}, \mathbf{Y}^{(0)},$ and $\mathbf{Z}^{(0)}$ to zero
4 **for** $k = 0$ **to** $K - 1$ **do**
5   Update $\hat{\mathbf{S}}^{(k+1)}$ by running Algorithm 1;
6   Update $\hat{\boldsymbol{\Theta}}^{(k+1)}$ by running Algorithm 2;
7 **end**
8 $\hat{\boldsymbol{\Theta}} = \hat{\boldsymbol{\Theta}}^{(K)}, \hat{\mathbf{S}} = \hat{\mathbf{S}}^{(K)}$.

---

where the iteration subscripts from $\boldsymbol{\Theta}^{(t+1)}$ and $\mathbf{Y}^{(t)}$ have been omitted for simplicity. Similar to the case of updating $\mathbf{P}$, we update $\mathbf{Q}$ as

$$\mathbf{Q}^{(t+1)} = \mathcal{P}_\delta \left(\mathbf{S}\boldsymbol{\Theta} - \boldsymbol{\Theta}\mathbf{S} + \frac{1}{\beta}\mathbf{Y} \right). \tag{31}$$

Finally, the Lagrange multiplier $\mathbf{Y}$ is updated as follows

$$\mathbf{Y}^{(t+1)} = \mathbf{Y}^{(t)} + \beta(\mathbf{S}\boldsymbol{\Theta} - \boldsymbol{\Theta}\mathbf{S} - \mathbf{Q}), \tag{32}$$

where the iteration subscripts from $\boldsymbol{\Theta}^{(t+1)}$ and $\mathbf{Q}^{(t+1)}$ have been similarly omitted for clarity.

We can also use the adaptive strategy in (22) to adjust β during iterations, with $\mathbf{r}^{(t)}$ and $\mathbf{s}^{(t)}$ defined as

$$\mathbf{r}^{(t)} = \mathbf{S}\boldsymbol{\Theta}^{(t)} - \boldsymbol{\Theta}^{(t)}\mathbf{S} - \mathbf{Q}^{(t)},$$

and

$$\mathbf{s}^{(t)} = \beta^{(t)}\mathbf{S}\big(\mathbf{Q}^{(t)} - \mathbf{Q}^{(t-1)}\big) - \beta^{(t)}\big(\mathbf{Q}^{(t)} - \mathbf{Q}^{(t-1)}\big)\mathbf{S}.$$

A pseudocode of the steps to be performed for the update of $\boldsymbol{\Theta}$ is summarized in Algorithm 2.

D. Graph Learning Algorithm and Convergence Analysis

Leveraging the results presented in the previous two subsections, the steps to run our iterative scheme to find a solution $(\hat{\boldsymbol{\Theta}}, \hat{\mathbf{S}})$ to (8) are summarized in Algorithm 3.

Before presenting the associated theoretical analysis, several comments regarding the implementation of Algorithm 3 are in order:

- For simplicity, the algorithm considers a fixed number of iterations, but a prudent approach is to monitor the cost reduction at each iteration and implement an early exit approach if no meaningful improvement is achieved.
- The value of the hyperparameters ρ, η and δ is an input to the algorithm. We note that η is typically set to a small value to guarantee that the (sparsity promoting) ℓ_1 norm plays a more prominent role. Moreover, the value of δ should be chosen based on the quality of the estimate of the sample covariance matrix $\hat{\Sigma}$. The higher the number of observations R (hence, the better the quality of $\hat{\Sigma}$), the smaller the value of δ. Similarly, if the number of nodes N is high, the value of δ should be re-scaled accordingly, so that the constraint does not become too restrictive.
- The update of $\mathbf{S}$ poses the primary computational challenge, mainly due to the complex nature of its estimation. In contrast to Θ, which is primarily estimated from the data matrix $\hat{\Sigma}$, the estimation of $\mathbf{S}$ relies on its interplay with Θ through the relaxed commutativity constraint. This indirect relationship adds to the complexity of the estimation, as it does not directly benefit from data-driven insights, often necessitating greater precision in solving the corresponding subproblem. Furthermore, the constraints imposed on $\mathbf{S}$ are substantially more complex than those on Θ, thereby increasing the computational load to obtain a solution that lies within the feasible set. To alleviate the computational demand, we may employ a relatively loose stopping criterion for the Θ subproblem, which can expedite convergence without significantly affecting the quality of the solution.

To establish the theoretical convergence of Algorithm 3, we begin by introducing several definitions and (mild) assumptions pertinent to Problem (8).

Let $f(\Theta, \mathbf{S})$ denote the objective function and $\mathcal{X}$ the feasible set of Problem (8), respectively. We define $(\bar{\Theta}, \bar{\mathbf{S}})$ as a *block coordinatewise minimizer* of Problem (8) if:

$$f(\bar{\Theta}, \bar{\mathbf{S}}) \leq f(\Theta, \bar{\mathbf{S}}), \quad \forall \Theta \in \bar{\mathcal{X}}_{\Theta}, \tag{33}$$

and

$$f(\bar{\Theta}, \bar{\mathbf{S}}) \leq f(\bar{\Theta}, \mathbf{S}), \quad \forall \mathbf{S} \in \bar{\mathcal{X}}_{\mathbf{S}}, \tag{34}$$

where $\bar{\mathcal{X}}_{\Theta} = \{\Theta \in \mathbb{R}^{N \times N} \,|\, (\Theta, \bar{\mathbf{S}}) \in \mathcal{X}\}$, and $\bar{\mathcal{X}}_{\mathbf{S}} = \{\mathbf{S} \in \mathbb{R}^{N \times N} \,|\, (\bar{\Theta}, \mathbf{S}) \in \mathcal{X}\}$.

Furthermore, we introduce the following assumptions for our analysis:

Assumption 1: The parameter δ in Problem (8) is sufficiently large to ensure that the feasible set of the subproblem (10) is nonempty at every iteration.

We require Assumption 1 because the intersection $\mathcal{S} \cap \{\mathbf{S} \in \mathbb{R}^{N \times N} \,|, \|\Theta \mathbf{S} - \mathbf{S}\Theta\|_F \leq \delta\}$ may otherwise be empty, implying that the feasible set of subproblem (10) could be nonexistent. However, Assumption 1 is relatively mild, as we can always choose a sufficiently large δ to ensure that the feasible set of subproblem (10) remains nonempty at every iteration.

Given Assumption 1, our algorithm is guaranteed to find a minimizer of subproblem (10) throughout its iterations. Additionally, the feasibility of subproblem (23) is inherently assured.

Assumption 2: The matrix $\hat{\Sigma}$ in Problem (8) is positive definite.

Assumption 2, which requires all the eigenvalues to be nonzero, guarantees that subproblem (23) is well defined. Without this assumption, the objective function in (23) may fail to achieve a finite minimum value. In cases where Assumption 2 may not hold, incorporating a norm regularizer for Θ would bound the solution, thereby ensuring the existence of a minimizer.

Theorem 1: Let $\left\{\left(\hat{\Theta}^{(k)}, \hat{\mathbf{S}}^{(k)}\right)\right\}_{k \in \mathbb{N}}$ be a sequence generated by Algorithm 3. Under Assumptions 1 and 2, every limit point of $\left\{\left(\hat{\Theta}^{(k)}, \hat{\mathbf{S}}^{(k)}\right)\right\}_{k \in \mathbb{N}}$ is a block coordinatewise minimizer of Problem (8).

The proof of Theorem 1 is deferred to Appendix C. This theorem asserts the subsequence convergence of our algorithm to a *block coordinatewise minimizer* of Problem (8). When the *block coordinatewise minimizer* lies within the interior of the feasible set, it becomes a stationary point. The theorem's assertions are significant both theoretically and practically. As discussed earlier in the paper, GMRFs are a particular case of Gaussian graph-stationary processes. Taking this into account, we can always initialize our algorithm using the solution estimated by GL (which is optimal for GMRF) and then, run iteratively the updates over Θ and $\mathbf{S}$ in Algorithm 3, to get an (enhanced) coordinatewise minimum estimate. Other approaches to improve the coordinate-wise estimation involve initializing $\mathbf{S}$ with the solution provided by (6), as described in Algorithm 3, or using the $\mathbf{S}$ associated with the precision matrix $(\hat{\Sigma}^{-1})$, which may avoid the additional runtime complexity introduced by GSR initialization.

Regarding the computational complexity, it is important to note that solving the proposed graph inference problem using a standard convex solver, such as CVX, results in a computational complexity of $O(N^7)$. This complexity becomes infeasible in many practical scenarios when the number of nodes N is large. In contrast, our approach significantly reduces the computational costs and provides a more scalable solution by leveraging linearized ADMM to solve the optimization problems in (11) and (24). For (11), the computational cost per iteration is dominated by $O(N^3)$ matrix multiplications and the Dykstra projection, which has a cost of $O\left(N^2 \log(N) \log(1/\epsilon)\right)$. For (24), the key operations include matrix multiplications, eigenvalue decomposition, and updating the dual variable, with an overall per-iteration complexity of $O(N^3)$. Both problems share a worst-case convergence rate of $O(1/k)$ [37], meaning that achieving an ϵ-accurate solution requires $O(1/\epsilon)$ iterations. Consequently, the total computational complexity for solving (11) is $O\left(\left(N^3 + N^2 \log(N) \log(1/\epsilon)\right)\frac{1}{\epsilon}\right)$, while for (24), it is $O\left(N^3 \frac{1}{\epsilon}\right)$.

V. NUMERICAL EXPERIMENTS

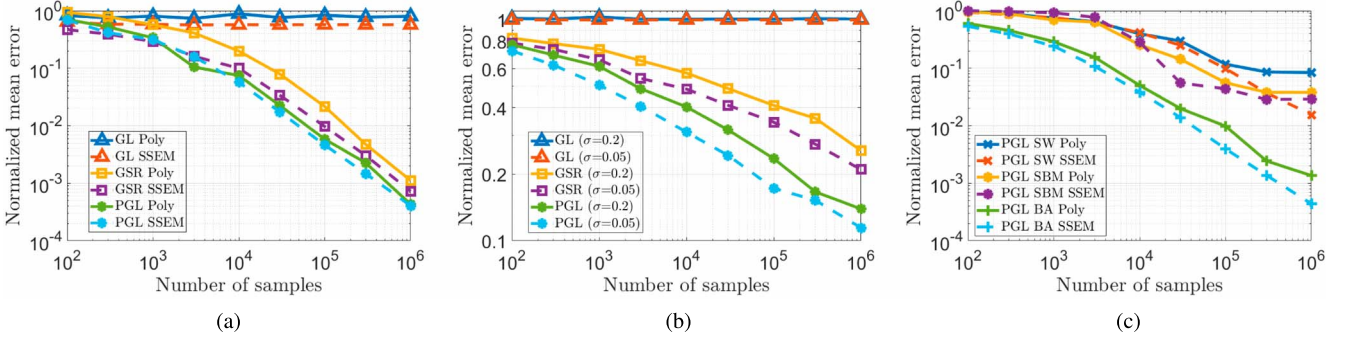This section evaluates quantitatively the performance of PGL. Since PGL can be understood as a generalization of the

Fig. 1. Graph estimation error nme($\mathbf{S}^*$, $\hat{\mathbf{S}}$) vs. number of samples R for different graph learning approaches (PGL, GL, and GSR) and simulations setups. Specifically, the six lines reported in each subplot correspond to the combination of (a) 3 different graph learning methods and 2 different covariance setups (SSEM and Poly) for a noise-free scenario; (b) 3 different graph learning schemes and 2 noise levels $\sigma \in \{0.05, 0.2\}$ for a Poly setup; and (c) 3 graph generation models and 2 covariance models (SSEM and Poly) for the PGL algorithm in a noise-free scenario.

widely adopted GL (with $\boldsymbol{\Theta}$ being any polynomial in $\mathbf{S}$), in most test cases, we will test both algorithms. Similarly, we also test the learning performance of GSR [13], which assumes stationarity but not Gaussianity. The performance results obtained from both synthetic and real-data experiments are summarized in Figs. 1–4. Unless otherwise stated, to assess the quality of the estimated GSO, we use the normalized mean error between the estimated and true $\mathbf{S}$. Mathematically, this entails computing[1]

$$\text{nme}(\mathbf{S}^*, \hat{\mathbf{S}}) = \frac{\|\mathbf{S}^* - \hat{\mathbf{S}}\|_F^2}{\|\mathbf{S}^*\|_F^2}, \tag{35}$$

where $\hat{\mathbf{S}}$ and $\mathbf{S}^*$ represent the estimated and true $\mathbf{S}$ respectively. Moreover, for the synthetic experiments we test the graph learning algorithms over 100 realizations of random graphs $\{\mathcal{G}_i\}_{i=1}^{100}$ and report the average normalized mean error $\frac{1}{100} \sum_{i=1}^{100} \text{nme}(\mathbf{S}_i^*, \hat{\mathbf{S}}_i)$.

If not specified otherwise, in our synthetic experiments, we consider graphs with $N = 20$ nodes generated using the Erdös-Rényi (ER) model with a link probability of $p = 0.1$. Regarding the generation of the graph signals, three different setups for the covariance matrices have been studied. For the setup referred to as "Poly", the covariance matrix $\boldsymbol{\Sigma}$ is generated as a random polynomial of the GSO of the form $\boldsymbol{\Sigma} = (\sum_{l=0}^{L-1} h_l \mathbf{S}^l)^2$, where h_l are random coefficients drawn from a normalized zero-mean Gaussian distribution, and the square operator guarantees matrix $\boldsymbol{\Sigma}$ to be positive definite. The setup referred to as "SSEM" constructs the covariance matrices following the sparse structural equation model [38] for graph signal generation as $\boldsymbol{\Sigma} = (\mathbf{I} - \mathbf{S})^2$, where $\mathbf{S}$ is selected to ensure that $\boldsymbol{\Sigma}$ is positive definite. The setup referred to as "MRF" constructs the covariance matrices following the assumptions made by GL as $\boldsymbol{\Sigma} = (\mu \mathbf{I} + \nu \mathbf{S})^{-1}$, where μ is some positive number large

enough to assure that $\boldsymbol{\Sigma}^{-1}$ is positive definite and ν is some positive random number.

A. Test Case 1: Estimation Error Versus Number of Samples for Multiple Synthetic Scenarios

In this first test case, we employ synthetic scenarios for testing the performance of our approach in terms of nme($\mathbf{S}^*$, $\hat{\mathbf{S}}$) vs. R and also compare the results with other methods from the literature. The different scenarios considered are detailed below.

Error vs. number of samples for different graph learning methods and signal models. The results of the experiment depicted in Fig. 1(a), compare the nme($\mathbf{S}^*$, $\hat{\mathbf{S}}$) (y-axis) of various algorithms with respect to the number of available samples R (x-axis). Moreover, we consider SSEM and Poly setups for data generation. The results shown in Fig. 1(a) reveal that: i) PGL outperforms its competitors, ii) the error decreases as R increases, and iii) estimation is more accurate for SSEM than for Poly. Next, we discuss these three main findings in greater detail. All algorithms do a better job estimating the graph for the SSEM model. Since $\boldsymbol{\Theta}$ for SSEM is a specific second-order polynomial in $\mathbf{S}$ it can be seen as a particular case of Poly, and consequently, a scenario from which the graph structure is easier to estimate. Indeed, while GL fails to estimate the graph for the Poly model, it is able to estimate some of the links for the SSEM. However, the estimation error of GL is quite large and does not change with the number of samples R, demonstrating that the poor performance is due to a model mismatch. This will be further confirmed in Section V-B, where we simulate a GMRF data generation setup that aligns perfectly with the assumptions made by GL. Regarding PGL and GSR, we observe that the error decreases almost linearly with the number of samples R. Perhaps more importantly, we also observe that, as R increases, the gap between PGL and GSR diminishes. For example, while for the Poly case GSR needs 10 times more samples than PGL to achieve an error of 10^{-1}, GSR only needs 3 times more samples than PGL to achieve an error of 10^{-3}. This illustrates that, as expected, the gains associated with assuming Gaussianity are stronger when
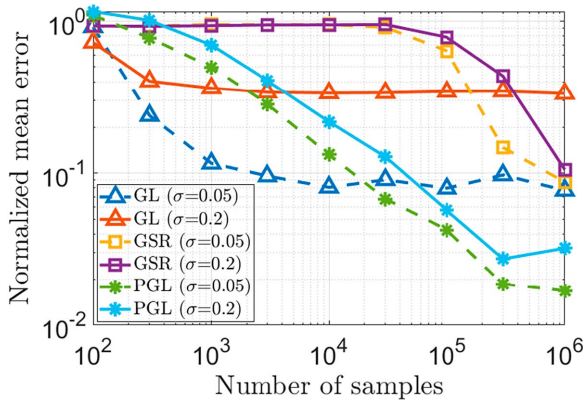
Fig. 2. Graph estimation error nme($\mathbf{S}^*, \hat{\mathbf{S}}$) vs. number of samples R for data generated according to a GMRF. We consider three different graph learning schemes (PGL, GL, and GSR) and two levels of additive white noise ($\sigma \in \{0.05, 0.2\}$), giving rise to the six lines in the figure.

the number of observations R is small, vanishing as R grows very large.

Error vs. number of samples for noisy observations. Next, we assess the performance of the graph learning algorithms in the presence of additive white Gaussian observation noise. The results are shown in Fig. 1(b). As in the previous test case, we report nme($\mathbf{S}^*, \hat{\mathbf{S}}$) vs R for PGL, GL and GSR. The difference here is that we consider only the more intricate signal generation model (Poly) and two normalized noise levels ($\sigma = 0.05$, $\sigma = 0.2$). The main observations in this case are: i) PGL outperforms GSR and GL when R is large, ii) the error for PGL and GSR decreases as R increases, while the one for GL is flat and close to 1, iii) the estimation performance for PGL and GSR worsens as the noise level σ increases, deteriorating noticeably with respect to the setup in Fig. 1(a), and iv) the gap between PGL and GSR grows. The findings in i) and ii) are consistent with those found in the previous test case. Finding iii) is expected and common in all graph learning approaches. Finally, the larger gap between PGL and GSR as R increases, as observed in finding iv), highlights the benefits of employing a more sophisticated model as PGL. By jointly considering (AS1) and (AS2), PGL is better equipped to handle challenging scenarios such as polynomial covariances and observation noise. While this increased sophistication introduces greater complexity to the problem, it also enables PGL to achieve significantly improved results compared to GSR, particularly when R is large.

Error vs. number of samples for different graph models. This test case considers network models other than ER. In particular, three different types of graphs are considered: 1) Small World (SW) graphs with mean node degree 4 and rewiring probability 0.15; 2) Stochastic block model (SBM) graphs with 4 clusters, and intra and inter-cluster edge probability of 0.8 and 0.05, respectively; and 3) Barabási-Albert (BA) graphs with 2 edges to attach at every step. As in the first test case, we consider two types of signal generation models: SSEM and Poly. Fig. 1(c) reports the error vs. the number of samples for the six combinations considered (3 types of graphs and 2 types

of signal generation models). The results show that there is a significant difference in performance between SW, SBM, and BA, which is part due to the sparsity level present in each graph. One of the assumptions codified in our model is that the graph is sparse, and, as a result, our algorithm does a better job estimating BA (the one with the lowest average degree) than SBM and SW (the one with the highest average degree). Finally, we also note that the estimation error achieved with SSEM consistently outperforms that of the Poly setup. These findings align with the results shown in Fig. 1(a) for ER graphs and support the theoretical discussion, which identified SSEM as a specific instance of Poly. Furthermore, the results obtained when comparing the different approaches across various graph types are consistent with those presented for ER graphs. Additional details about these results are available in the code repository.

B. Test Case 2: Noisy GMRF Graph Signals

In this test case, the goal is to assess the behavior of PGL for GMRF observations, which is the setup that motivated the development of the GL algorithm.

Estimation error considering noisy GMFR signals. In this experiment, we replicate the scenario from Fig. 1(b), utilizing a GMFR model to generate the signals. The performance of PGL, GL and GSR for two different noise levels, $\sigma \in \{0.05, 0.2\}$, is depicted in Fig. 2. The main observations are: i) across all considered approaches, increasing the noise level σ leads to a deterioration in terms of nme($\mathbf{S}^*, \hat{\mathbf{S}}$); ii) GSR always performs worse than PGL, providing very poor results when $R < 10^4$; and iii) GL outperforms PGL when the number of observations R is small. Findings i) and ii) are unsurprising and consistent with the behavior observed in the previous experiments, showcasing the benefits of considering the log-likelihood regularization in the optimization run by PGL. Regarding iii), GL outperforming PGL is expected, since the latter needs to "learn" the particular polynomial between $\boldsymbol{\Theta}$ and $\mathbf{S}$.

However, as the value of R increases, the error in PGL gradually decreases, while the error in GL remains relatively constant, leading to PGL outperforming GL for large values of R. This behavior is more surprising and can be attributed to the fact that GL focuses on learning the precision matrix $\boldsymbol{\Theta}$, while PGL balances the accuracy in terms of both the precision $\boldsymbol{\Theta}$ and the graph $\mathbf{S}$. Since the reported error focuses on the estimation $\mathbf{S}$, the values of the diagonal of $\boldsymbol{\Theta}$ are not relevant for nme($\mathbf{S}^*, \hat{\mathbf{S}}$) and this can explain that GL, which is a maximum likelihood estimate for $\boldsymbol{\Theta}$, does not yield the minimum nme($\mathbf{S}^*, \hat{\mathbf{S}}$).

C. Test Case 3: Computational Complexity

Here, we compare the runtime obtained by the efficient implementation of our PGL scheme provided in Algorithm 3 and a generic block-coordinate alternating minimization algorithm that uses CVX [39], the most common off-the-shelf solver for convex problems.

Computational complexity and estimation error. The objective of this experiment is to evaluate the running time and

TABLE I
VALUES OF THE RUNNING TIME AND nme($\mathbf{S}^*, \hat{\mathbf{S}}$) FOR TWO DIFFERENT IMPLEMENTATIONS OF PGL: ONE USING AN OFF-THE-SHELF CONVEX SOLVER (PGL-CVX) AND THE OTHER ONE USING THE EFFICIENT METHOD IN ALGORITHM 3 (PGL-ALG.3). THE RESULTS ARE SHOWN FOR DIFFERENT GRAPH SIZES N AND − INDICATES THAT THE RUNNING TIME WAS MORE THAN 2 HOURS

Alg. \ N	20	30	40	50	60	70	80	Metric
PGL-CVX	$2.37 \cdot 10^1$	$3.88 \cdot 10^1$	$1.29 \cdot 10^2$	$1.11 \cdot 10^3$	$3.13 \cdot 10^3$	—	—	**Time** (s)
PGL-Alg. 3	$2.49 \cdot 10^0$	$2.72 \cdot 10^0$	$2.88 \cdot 10^0$	$3.81 \cdot 10^0$	$4.45 \cdot 10^0$	$5.86 \cdot 10^0$	$6.75 \cdot 10^0$	
PGL-CVX	$7.98 \cdot 10^{-4}$	$3.12 \cdot 10^{-3}$	$1.35 \cdot 10^{-2}$	$2.59 \cdot 10^{-2}$	$9.16 \cdot 10^{-2}$	—	—	nme($\mathbf{S}^*, \hat{\mathbf{S}}$)
PGL-Alg. 3	$4.66 \cdot 10^{-4}$	$1.30 \cdot 10^{-3}$	$2.18 \cdot 10^{-3}$	$5.07 \cdot 10^{-3}$	$1.59 \cdot 10^{-2}$	$1.43 \cdot 10^{-1}$	$7.94 \cdot 10^{-1}$	

estimation error for different versions of our algorithm as the number of nodes increases. In particular, the experiment focuses on the Poly setup, utilizing $R = 10^6$ graph signals, and averages the results over 50 graph realizations. Table I lists the elapsed time required to obtain the graph and precision estimates for problems with different numbers of nodes N using two algorithms: 1) solving the optimization in (8) with a block coordinate approach where the minimization over $\mathbf{\Theta}$ given $\mathbf{S}$ and the minimization over $\mathbf{S}$ given $\mathbf{\Theta}$ are run using CVX (this algorithm is labeled as PGL-CVX) and 2) employing the efficient scheme outlined in Algorithm 3 (this choice is labeled as PGL-Alg. 3). To guarantee that the results are comparable, the nme($\mathbf{S}^*, \hat{\mathbf{S}}$) is also reported. Examining the listed running times, we observe that as the number of nodes increases, both solvers require more time to estimate the graph, with the number of variables scaling with N^2. More importantly, there exists a noticeable difference between PGL-CVX and PGL-Alg. 3. Not only is the latter faster for small graphs, but the performance gap grows significantly as N increases. Note that the results for graphs with more than $N = 60$ nodes are not reported for PGL-CVX, since the computation time exceeded two hours. In terms of nme($\mathbf{S}^*, \hat{\mathbf{S}}$), our approach yields results that are comparable to (and even slightly better than) those of PGL-CVX. These results align with the discussion in Section IV regarding the computational complexity of our method, confirming that, relative to a generic solver, our approach significantly reduces computational costs and provides a more scalable solution, especially for large graphs.

D. Test Case 4: Real Data Scenarios

Finally, we compare different graph learning algorithms (including PGL) in the context of two different graph-aware applications. Both applications are related to financial data, which was collected from Yahoo! Finance[2]. The details and results are provided next.

Stock graph-based clustering from returns. For this real-data experiment, we tested our graph learning algorithm on financial data and further performed a clustering task. Specifically, 40 companies from 4 different sectors of the S&P 500 were selected (10 companies from each sector) and the market data (log returns) of each company in the period 2010-2015 was retrieved. This gives rise to a data matrix of size $\mathbf{X} \in \mathbb{R}^{40 \times 1510}$. In this application, as in many others dealing with graph learning,
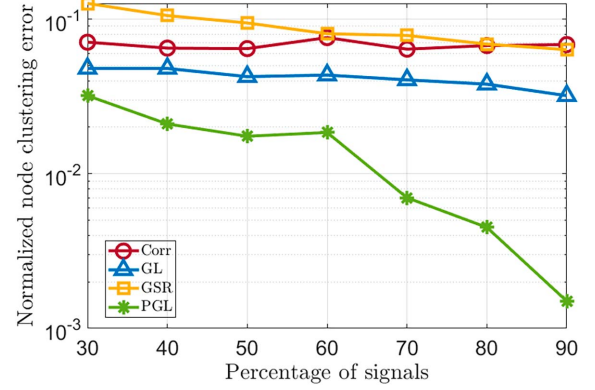
[2]https://finance.yahoo.com/

Fig. 3. The experiment considers 40 companies of the S & P 500 from 4 different sectors, using as nodal signals the daily returns in the period 2010-2015. Four different graph learning schemes are considered: correlation networks (Corr), PGL, GL and GSR. After learning the graph, a spectral clustering method is implemented. The figure shows the normalized node clustering error (fraction of wrongly clustering nodes) as the percentage of available signals to learn the graph increases.

we do not have access to the ground truth graph. Hence, we cannot quantify the quality of the estimated network directly by using a metric like nme or F1 score. As a result, we need to assess the quality of the estimated graph indirectly, using the graph as input for an ulterior task. In this experiment, we use the graph to estimate the community each company belongs to in an unsupervised manner. More specifically, we implement the following pipeline: 1) estimate several graphs from a subset of the available graph signals, 2) use spectral clustering to group the nodes into 4 communities (as many as sectors were selected), and 3) compute the ratio of incorrectly clustered nodes. To obtain more reliable results, we averaged the clustering errors over 50 realizations in which the subset of graph signals was chosen uniformly at random. The vertical axis of Fig. 3 represents the normalized clustering error $\frac{1}{50} \sum_{i=1}^{50} \frac{N_w^{(i)}}{N}$ which is computed as the average of the fraction between the number of wrongly clustered nodes N_w and the total number of nodes N over 50 graph realizations. The horizontal axis of Fig. 3 represents the percentages of graph signals used to estimate $\hat{\mathbf{S}}$. Results are provided for 4 graph learning approaches: PGL, GL, GSR, and "Corr", which estimates the graph as a thresholded version of the correlation matrix. The idea is that companies from the same sector have stronger ties among them and, as a result, when running a graph-based clustering method, the 4 sectors should arise. Based on the results presented in Fig. 3, it
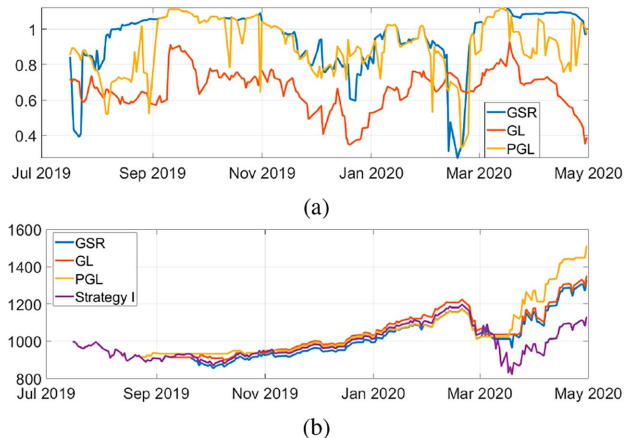
Fig. 4. This experiment learns the graph connecting the 7 FAAMUNG stocks using as signals the closing price from July 2019 to May 2020. Three different graph learning methods are used (PGL, GL, and GSR) and a different graph is learned for every day (using the signals of the previous 30 days). Subplot (a) shows the value of the algebraic connectivity (λ_2) associated with each one of the 200×3 estimated graphs. Subplot (b) shows the value of the portfolio for 4 different investment strategies, 3 of which are based on the algebraic connectivity estimated in the subplot (a).

can be observed that PGL outperforms the other alternatives and additionally, as the number of available signals increases, the clustering error for PGL drops significantly. The superiority of PGL may indicate that considering more complex relationships among stocks (beyond the simple correlations considered in "Corr" or the partial correlations considered in GL) is a better model to understand the dependencies between log-returns in the stock market. On the other hand, GSR shows high clustering error with a limited number of samples, but it improves as the number of available samples increases. This observation aligns with our earlier discussion in Section III, where we highlighted that this particular model offers greater generality but necessitates a substantial number of samples to accurately estimate the graph.

Learning sequential graphs for investing. This experiment deals with a different real-world problem and dataset. We still look at stocks, but consider now the close price of the 7 FAA-MUNG[3] companies from Jul 2019 to May 2020. The goal is to design an investment strategy to maximize the benefits using as input a graph describing the relationships among the companies. Inspired by the approach in [40], we first build a graph, analyze its connectivity and then, invest (or not) in a stock according to the graph connectivity. To be more specific, we use the close price to estimate multiple 7×7 adjacency matrices. We estimate a total of 200 matrices, where each adjacency matrix (graph) is estimated in a rolling window fashion. The window consists of 30 consecutive days and for each graph estimation, we shift the window one day at a time. These graph estimations help to visualize how the graph topology changes during this time period. The finding in [40] is that big changes in the graph connectivity indicate opportunities to invest. To that end, we

keep track of the algebraic connectivity value, which is the second smallest eigenvalue (λ_2) of the estimated Laplacian matrix. The lower the value, the less connected the graph is and, as a result, the easier breaking the graph into multiple components is. Fig. 4(a) shows the value of λ_2 for each of the 200 considered graphs (each associated with a 30-day period). Then, using the approach in [40] we invest only if λ_2 is below a fixed threshold. We learn the graph using 3 methods (GL, GSR, and PGL) and, for each of them, we select the best possible threshold (the one that maximizes the benefits). Correlation was not used here due to its poor performance. The results of applying the graph-connectivity-based investment strategy to the graphs estimated with each of the algorithms are shown in Fig. 4(b). The purple line labeled as "Strategy I" is used as a baseline and involves investing the initial amount evenly across the stocks of seven companies. All holdings are sold daily, and the full amount is reinvested in the same manner. In contrast, the graph-based strategies operate by investing the initial amount, selling all holdings whenever λ_2 exceeds a predefined threshold, and reinvesting the money when λ_2 falls below the threshold again. By analyzing the results obtained, we can observe that: i) the graph-based strategies outperform (gain more money than) the baseline; ii) the strategies based on GL and GSR provide similar gains; and iii) the strategy based on PGL yields the highest gains. This provides additional validation for the graph learning methodology proposed in this paper.

VI. CONCLUSION

This paper has introduced PGL, a novel scheme for learning a graph from nodal signals, with our key contribution being the modeling of the signals as Gaussian and stationary on the graph. This approach opens the door to a graph learning formulation that leverages the advantages of GL (needing a relatively small number of signals to get a good estimation of the graph structure) while encompassing a more comprehensive model (because it handles cases where the precision matrix can be any polynomial form of the sought graph). Given the increased complexity and nonconvex nature of the resulting optimization problem, we have developed a low-complexity algorithm that alternates between estimating the graph and precision matrices and have characterized its convergence to a block coordinate-wise minimum. To assess its efficacy, we have conducted numerical simulations comparing PGL with various alternatives, using both synthetic and real data. The results have showcased the benefits of our approach, motivating the adoption and further investigation of the proposed graph learning methodology.

APPENDIX A
COMPUTATIONS OF PROJECTIONS

We present the details about how to compute the projections $\mathcal{P}_{\mathcal{S}_A}$ and $\mathcal{P}_{\mathcal{S}_B}$.

The computation of $\mathcal{P}_{\mathcal{S}_A}$ is straightforward as follows:

$$\mathcal{P}_{\mathcal{S}_A}(\mathbf{A}) = (\mathbf{A} + \mathbf{A}^\top)/2. \tag{36}$$

---

[3]Facebook, Amazon, Apple, Microsoft, Uber, Netflix, and Google.

The projection $\mathcal{P}_{\mathcal{S}_B}$ is defined as the minimizer of the optimization problem as follows:

$$\mathcal{P}_{\mathcal{S}_B}(\mathbf{A}) := \arg\min_{\mathbf{X} \in \mathcal{S}_B} \frac{1}{2}\|\mathbf{X} - \mathbf{A}\|_F^2. \quad (37)$$

To compute the projection $\mathcal{P}_{\mathcal{S}_B}$, we solve Problem (37) row by row. For the j-th row, we solve the following problem,

$$\begin{aligned} \underset{\mathbf{x} \in \mathbb{R}^N}{\text{minimize}} \quad & \frac{1}{2}\|\mathbf{x} - \mathbf{a}\|^2, \\ \text{subject to} \quad & \mathbf{x}^\top \mathbf{1} \geq 1, \ x_j = 0, \ \mathbf{x}_{\setminus j} \geq \mathbf{0}, \end{aligned} \quad (38)$$

where $\mathbf{a} \in \mathbb{R}^N$ contains all entries of the j-th row of $\mathbf{A}$, x_j denotes the j-th entry of $\mathbf{x}$, and $\mathbf{x}_{\setminus j} \in \mathbb{R}^{N-1}$ contains all entries of $\mathbf{x}$ except the j-th one.

Let $\hat{\mathbf{x}}$ denote the optimal solution of Problem (38). Proposition 1 below, proved in Appendix B, presents the optimal solution of Problem (38).

Proposition 1: The optimal solution $\hat{\mathbf{x}}$ to Problem (38) can be obtained as follows:
- If $\sum_{i \neq j} \max(a_i, 0) \geq 1$, then $\hat{x}_j = 0$, and $\hat{x}_i = \max(a_i, 0)$, for $i \neq j$.
- If $\sum_{i \neq j} \max(a_i, 0) < 1$, then $\hat{x}_j = 0$, and $\hat{x}_i = \max(a_i + \phi, 0)$, for $i \neq j$, where ϕ satisfies $\sum_{i \neq j} \max(a_i + \phi, 0) = 1$.

Several efficient approaches have been developed to tackle the piecewise linear equation $\sum_{i \neq j} \max(a_i + \phi, 0) = 1$. Among these, the sorting-based method described in [41] is noteworthy. Central to this method is the sorting of the vector $\mathbf{a}$, which constitutes the most computationally intensive step, generally necessitating $\mathcal{O}(N \log N)$ operations.

APPENDIX B
PROOF OF PROPOSITION 1

Proof: The Lagrangian of the optimization in (38) is

$$L(\mathbf{x}, u, \mathbf{v}) = \frac{1}{2}\|\mathbf{x} - \mathbf{a}\|^2 - u(\mathbf{x}^\top \mathbf{1} - 1) - \langle \mathbf{v}_{\setminus j}, \mathbf{x}_{\setminus j} \rangle + v_j x_j,$$

where $u \in \mathbb{R}$ and $\mathbf{v} \in \mathbb{R}^N$ are KKT multipliers. Let $(\hat{\mathbf{x}}, \hat{u}, \hat{\mathbf{v}})$ be the primal and dual optimal point. Then $(\hat{\mathbf{x}}, \hat{u}, \hat{\mathbf{v}})$ must satisfy the KKT system:

$$\hat{x}_i - a_i - \hat{u} - \hat{v}_i = 0, \quad \text{for } i \neq j; \quad (39)$$

$$\hat{x}_j - a_j - \hat{u} + \hat{v}_j = 0; \quad (40)$$

$$\hat{x}_i \geq 0, \ \hat{v}_i \geq 0, \ \hat{v}_i \hat{x}_i = 0, \quad \text{for } i \neq j; \quad (41)$$

$$\hat{x}_j = 0, \ \hat{u} \geq 0, \ \hat{\mathbf{x}}^T \mathbf{1} \geq 1; \quad (42)$$

$$\hat{u}(\hat{\mathbf{x}}^T \mathbf{1} - 1) = 0; \quad (43)$$

Therefore, for any $i \neq j$, it holds that $\hat{x}_i = a_i + \hat{u} + \hat{v}_i$. Then we obtain the following results:
- If $a_i + \hat{u} < 0$, then $\hat{v}_i = -(a_i + \hat{u})$ and $\hat{x}_i = 0$, following from the fact that $\hat{v}_i \hat{x}_i = 0$ and $\hat{x}_i \geq 0$.
- If $a_i + \hat{u} \geq 0$, then $\hat{v}_i = 0$. This can be obtained as follows: if $\hat{x}_i = 0$, then $\hat{v}_i = -(a_i + \hat{u}) \leq 0$. Since $\hat{v}_i \geq 0$, one has $\hat{v}_i = 0$; On the other hand, if $\hat{x}_i \neq 0$, then $\hat{v}_i = 0$, following from the fact that $\hat{v}_i \hat{x}_i = 0$. As a result, we get $\hat{x}_i = a_i + \hat{u}$.

Overall, we obtain that

$$\hat{x}_j = 0, \quad \text{and} \quad \hat{x}_i = \max(a_i + \hat{u}, 0), \quad \forall i \neq j. \quad (44)$$

On the other hand, $\hat{\mathbf{x}}$ and $\hat{u}$ satisfy that $\hat{\mathbf{x}}^T \mathbf{1} \geq 1$, $\hat{u} \geq 0$, and $\hat{u}(\hat{\mathbf{x}}^T \mathbf{1} - 1) = 0$. To this end, we can obtain the following results:
- If $\sum_{i \neq j} \max(a_i, 0) \geq 1$, then $\hat{u} = 0$, indicating that $\hat{x}_i = \max(a_i, 0)$, for any $i \neq j$.
- If $\sum_{i \neq j} \max(a_i, 0) < 1$, then $\hat{u} \neq 0$. This is because $\hat{u} = 0$ will result in $\hat{\mathbf{x}}^\top \mathbf{1} < 1$, which does not satisfy the KKT system. Together with the KKT condition that $\hat{u}(\hat{\mathbf{x}}^\top \mathbf{1} - 1) = 0$, one has $\hat{\mathbf{x}}^\top \mathbf{1} = 1$. Therefore, one obtains that, for any $i \neq j$, $\hat{x}_i = \max(a_i + \hat{u}, 0)$, where $\hat{u}$ is chosen such that $\sum_{i \neq j} \max(a_i + \hat{u}, 0) = 1$.

We note that the ϕ in Proposition 1 is exactly the dual optimal point $\hat{u}$. $\qquad \square$

APPENDIX C
PROOF OF THEOREM 1

Proof: The convergence result stated in Theorem 1 is based on the framework established by Theorem 2.3 in [42]. To demonstrate the validity of Theorem 1, it suffices to establish that the conditions and assumptions of Theorem 2.3 are satisfied in our context. Our approach to block updates aligns with the procedure delineated in equation (1.3a) of [42].

We first verify the conditions the requisite conditions of Theorem 2.3 in [42] are met. Specifically, Theorem 2.3 stipulates that the objective function, along with the feasible set of the optimization problem, should exhibit *block multiconvexity*. For Problem (8), the objective function f is convex with respect to each of the blocks $\boldsymbol{\Theta}$ and $\mathbf{S}$ when the other block is fixed, a property that defines *block multiconvexity* as per [42]. Moreover, the function f is strongly convex with respect to both $\boldsymbol{\Theta}$ and $\mathbf{S}$.

The feasibility constraints of Problem (8) form a set $\mathcal{X}$ that satisfies the criteria for *block multiconvexity* as defined in [42]. This is due to the convexity of the individual set maps $\mathcal{X}_{\boldsymbol{\Theta}}$ and $\mathcal{X}_{\mathbf{S}}$. The set map $\mathcal{X}_{\boldsymbol{\Theta}}$ is defined as

$$\mathcal{X}_{\boldsymbol{\Theta}} = \{\boldsymbol{\Theta} \in \mathbb{R}^{N \times N} | \boldsymbol{\Theta} \succeq \mathbf{0}, \|\boldsymbol{\Theta}\mathbf{S} - \mathbf{S}\boldsymbol{\Theta}\|_F \leq \delta\} \quad (45)$$

for some given $\mathbf{S}$, and similarly, the set map $\mathcal{X}_{\mathbf{S}}$ is defined as

$$\mathcal{X}_{\mathbf{S}} = \{\mathbf{S} \in \mathbb{R}^{N \times N} | \mathbf{S} \in \mathcal{S}, \|\boldsymbol{\Theta}\mathbf{S} - \mathbf{S}\boldsymbol{\Theta}\|_F \leq \delta\} \quad (46)$$

for some given $\boldsymbol{\Theta}$. Consequently, the optimization subproblems with respect to $\boldsymbol{\Theta}$ and $\mathbf{S}$ in Problem (8) are convex.

We now validate the assumptions required by Theorem 2.3 in [42] within the context of our setting. Specifically, it is required that the objective function f is bounded below over the feasible set $\mathcal{X}$, that is, $\inf_{(\boldsymbol{\Theta}, \mathbf{S}) \in \mathcal{X}} f(\boldsymbol{\Theta}, \mathbf{S}) > -\infty$. This is indeed the case here, because the term $\rho\|\mathbf{S}\|_1 + \frac{\eta}{2}\|\mathbf{S}\|_F^2$ is nonnegative. Additionally, the function $-\log(\det(\boldsymbol{\Theta})) + \text{tr}(\hat{\boldsymbol{\Sigma}}\boldsymbol{\Theta})$ attains a finite infimum when $\hat{\boldsymbol{\Sigma}}$ is positive definite under Assumption 2. To see this, first observe that $\det(\boldsymbol{\Theta}) \leq \|\boldsymbol{\Theta}\|_2^N$, where $\|\boldsymbol{\Theta}\|_2$ is the largest eigenvalue of $\boldsymbol{\Theta}$. Consequently, $-\log\det(\boldsymbol{\Theta}) \geq$

$-N \log(\|\mathbf{\Theta}\|_2)$. Further, since $\mathrm{tr}(\hat{\mathbf{\Sigma}}\mathbf{\Theta}) \geq \gamma\mathrm{tr}(\mathbf{\Theta}) \geq \gamma\|\mathbf{\Theta}\|_2$, with $\gamma > 0$ being the smallest eigenvalue of $\hat{\mathbf{\Sigma}}$, we obtain

$$-\log\det(\mathbf{\Theta}) + \mathrm{tr}(\hat{\mathbf{\Sigma}}\mathbf{\Theta}) \geq -N\log(\|\mathbf{\Theta}\|_2) + \gamma\|\mathbf{\Theta}\|_2, \quad (47)$$

which indeed has a finite minimum. Thus, we conclude that $\inf_{(\mathbf{\Theta},\mathbf{S})\in\mathcal{X}} -\log\det(\mathbf{\Theta}) + \mathrm{tr}(\hat{\mathbf{\Sigma}}\mathbf{\Theta}) > -\infty$.

Furthermore, the existence of *block coordinatewise minimizers* is assured by the compactness of the feasible set. Moreover, Theorem 2.3 in [42] stipulates that set maps change continuously during iterations. Referring to (45) and (46), it is clear that the only constraint that changes through iterations is $\|\mathbf{\Theta}\mathbf{S} - \mathbf{S}\mathbf{\Theta}\|_F \leq \delta$, while the other constraints, $\mathbf{\Theta} \succeq \mathbf{0}$ and $\mathbf{S} \in \mathcal{S}$, remain constant. Given that $\|\mathbf{\Theta}\mathbf{S} - \mathbf{S}\mathbf{\Theta}\|_F$ is a continuous function with respect to both $\mathbf{\Theta}$ and $\mathbf{S}$, the set maps indeed change continuously, satisfying the theorem's conditions.

These verifications above have demonstrated that the conditions and assumptions of Theorem 2.3 are satisfied in our context, completing the proof. □

REFERENCES

[1] A. Buciulea and A. G. Marques, "Graph learning from Gaussian and stationary graph signals," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, 2023, pp. 1–5.

[2] E. D. Kolaczyk, *Statistical Analysis of Network Data: Methods and Models*. New York, NY, USA: Springer, 2009.

[3] O. Sporns, *Discovering the Human Connectome*. Boston, MA, USA: MIT Press, 2012.

[4] D.I Shuman, S.K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains," *IEEE Signal Process. Mag.*, vol. 30, no. 3, pp. 83–98, May 2013.

[5] A. Ortega, P. Frossard, J. Kovacevic, J. M. F. Moura, and P. Vandergheynst, "Graph signal processing: Overview, challenges, and applications," *Proc. IEEE*, vol. 106, no. 5, pp. 808–828, May 2018.

[6] J. V. D. M. Cardoso, J. Ying, and D. P Palomar, "Algorithms for learning graphs in financial markets," 2020, *arXiv:2012.15410*.

[7] G. Mateos, S. Segarra, A. G. Marques, and A. Ribeiro, "Connecting the dots: Identifying network structure via graph signal processing," *IEEE Signal Process. Mag.*, vol. 36, no. 3, pp. 16–43, May 2019.

[8] S. Sardellitti, S. Barbarossa, and P. Di Lorenzo, "Graph topology inference based on sparsifying transform learning," *IEEE Trans. Signal Process.*, vol. 67, no. 7, pp. 1712–1727, Apr. 2019.

[9] N. Meinshausen and P. Buhlmann, "High-dimensional graphs and variable selection with the lasso," *Ann. Statist.*, vol. 34, pp. 1436–1462, 2006.

[10] G. V. Karanikolas, G. B. Giannakis, K. Slavakis, and R. M. Leahy, "Multi-kernel based nonlinear models for connectivity identification of brain networks," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Shanghai, China, Mar. 20–25, 2016, pp. 6315–6319.

[11] J. Mei and J.M.F. Moura, "Signal processing on graphs: Estimating the structure of a graph," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, 2015, pp. 5495–5499.

[12] 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, Sep. 2017.

[13] S. Segarra, A. G. Marques, G. Mateos, and A. Ribeiro, "Network topology inference from spectral templates," *IEEE Trans. Signal Info. Process. Netw.*, vol. 3, no. 3, pp. 467–483, Sep. 2017.

[14] J. Ying, J. V. de M. Cardoso, and D. Palomar, "Nonconvex sparse graph learning under Laplacian constrained graphical model," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 7101–7113.

[15] V. Chandrasekaran, P. A. Parrilo, and A. S. Willsky, "Latent variable graphical model selection via convex optimization," *Ann. Statist.*, vol. 40, no. 4, pp. 1935–1967, 2012.

[16] A. Chang, T. Yao, and G. I. Allen, "Graphical models and dynamic latent factors for modeling functional brain connectivity," in *Proc.*

[17] A. Buciulea, S. Rey, and A. G. Marques, "Learning graphs from smooth and graph-stationary signals with hidden variables," *IEEE Trans. Signal Process.*, vol. 8, pp. 273–287, 2022.

[18] P. Danaher, P. Wang, and D. M. Witten, "The joint graphical lasso for inverse covariance estimation across multiple classes," *J. Roy. Statist. Soc.: Ser. B (Statist. Methodol.)*, vol. 76, no. 2, pp. 373–397, 2014.

[19] M. Navarro, Y. Wang, A. G. Marques, C. Uhler, and S. Segarra, "Joint inference of multiple graphs from matrix polynomials," *J. Mach. Learn. Res. (JMLR)*, vol. 23, no. 1, pp. 3302–3336, 2022.

[20] A. G. Marques, S. Segarra, G. Leus, and A. Ribeiro, "Stationary graph processes and spectral estimation," *IEEE Trans. Signal Process.*, vol. 65, no. 22, pp. 5911–5926, Nov. 2017.

[21] A. Sandryhaila and J. M. F. Moura, "Discrete signal processing on graphs," *IEEE Trans. Signal Process.*, vol. 61, no. 7, pp. 1644–1656, Apr. 2013.

[22] N. Perraudin and P. Vandergheynst, "Stationary signal processing on graphs," *IEEE Trans. Signal Process.*, vol. 65, no. 13, pp. 3462–3477, Jul. 2017.

[23] B. Girault, "Stationary graph signals using an isometric graph translation," in *Proc. Eur. Signal Process. Conf. (EUSIPCO)*, Aug. 2015, pp. 1516–1520.

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

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

[26] S. Segarra M. Sevilla, A. G. Marques, "Estimation of partially known Gaussian graphical models with score-based structural priors," in *Proc. 27th Int. Conf. Artif. Intell. Statist. (AISTATS)*, 2024, pp. 1558–1566.

[27] X. Cai, J. A. Bazerque, and G. B. Giannakis, "Inference of gene regulatory networks with sparse structural equation models exploiting genetic perturbations," *PLoS Comput. Biol.*, vol. 9, no. 5, 2013, Art. no. e1003068.

[28] H. Zou and T. Hastie, "Regularization and variable selection via the elastic net," *J. Roy. Statist. Soc.: Ser. B (Statist. Methodol.)*, vol. 67, no. 2, pp. 301–320, 2005.

[29] J. K. Tay, B. Narasimhan, and T. Hastie, "Elastic net regularization paths for all generalized linear models," *J. Statist. Softw.*, vol. 106, pp. 1–31, 2023.

[30] R. Tibshirani, "Regression shrinkage and selection via the lasso," *J. Roy. Statist. Soc.: Ser. B (Statist. Methodol.)*, vol. 58, no. 1, pp. 267–288, 1996.

[31] E. J Candes, M. B. Wakin, and S. P. Boyd, "Enhancing sparsity by reweighted 1 minimization," *J. Fourier Anal. Appl.*, vol. 14, no. 5–6, pp. 877–905, 2008.

[32] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Found. Trends® Mach. Learn.*, vol. 3, no. 1, pp. 1–122, 2011.

[33] Y. Sun, P. Babu, and D. P. Palomar, "Majorization-minimization algorithms in signal processing, communications, and machine learning," *IEEE Trans. Signal Process.*, vol. 65, no. 3, pp. 794–816, Feb. 2017.

[34] J. P. Boyle and R. L. Dykstra, "A method for finding projections onto the intersection of convex sets in Hilbert spaces," in *Advances in Order Restricted Statistical Inference*. Springer, 1986, pp. 28–47.

[35] M. Li, D. Sun, and K.-C. Toh, "A majorized ADMM with indefinite proximal terms for linearly constrained convex composite optimization," *SIAM J. Optim.*, vol. 26, no. 2, pp. 922–950, 2016.

[36] D. M. Witten and R. Tibshirani, "Covariance-regularized regression and classification for high dimensional problems," *J. Roy. Statist. Soc.: Ser. B (Statist. Methodol.)*, vol. 71, no. 3, pp. 615–636, 2009.

[37] B. He, F. Ma, and X. Yuan, "Optimally linearizing the alternating direction method of multipliers for convex programming," *Comput. Optim. Appl.*, vol. 75, no. 2, pp. 361–388, 2020.

[38] X. Cai, J. A. Bazerque, and G. B. Giannakis, "Sparse structural equation modeling for inference of gene regulatory networks exploiting genetic perturbations," *PLoS, Comput. Biol.*, Jun. 2013.

[39] M. Grant and S. Boyd, "CVX: Matlab software for disciplined convex programming, version 2.1," Mar. 2014. [Online.] Available: https://cvxr.com/cvx

[40] J. V. de M. Cardoso and D. Palomar, "Learning undirected graphs in financial markets," in *Proc. Asilomar Conf. Signals, Syst., Comput.*, 2020, pp. 741–745.

[41] L. Condat, "Fast projection onto the simplex and the l_1 ball," *Math. Program.*, vol. 158, no. 1–2, pp. 575–585, 2016.

[42] Y. Xu and W. Yin, "A block coordinate descent method for regularized multiconvex optimization with applications to nonnegative tensor factorization and completion," *SIAM J. Imag. Sci.*, vol. 6, no. 3, pp. 1758–1789, 2013.

Andrei Buciulea (Member, IEEE) received the bachelor's and master's degrees in telecommunications engineering from King Juan Carlos University, Madrid, Spain, in 2017 and 2019, respectively, and the Ph.D. (*cum laude*) degree in multimedia and communications, earning the international doctorate mention, from King Juan Carlos University and Carlos III University, in 2024. Currently, he is an Assistant Professor with the Department of Signal Theory and Communications, King Juan Carlos University. His research interests include data science, network science, graph signal processing, network topology inference, and machine learning. In 2020, he was awarded a predoctoral grant at King Juan Carlos University. In 2023, he received two competitive grants for research visits from the Tailor Connectivity Fund and the Comunidad de Madrid. He served as a reviewer for IEEE SPS conferences and journals and was the IEEE Student Branch Chair with King Juan Carlos University.
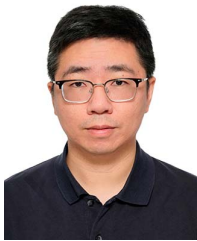
Jiaxi Ying (Member, IEEE) received the Ph.D. degree from the Department of Electronic and Computer Engineering, The Hong Kong University of Science and Technology, Hong Kong, in 2022. Currently, he is a Postdoctoral Fellow with the Department of Mathematics, Hong Kong University of Science and Technology. His research interests include the intersection of optimization, machine learning, signal processing, and statistics. He was a recipient of the Hong Kong Research Grants Council Postdoctoral Fellowship, the Outstanding Master's Thesis Award of the Chinese Institute of Electronics, the Excellent Master Thesis in Fujian Province, the HKUST RedBird Ph.D. Scholarship Program, and the NeurIPS Scholar Award.

Antonio G. Marques (Senior Member, IEEE) received the bachelor's (Hons.) degree in telecommunications engineering and the Ph.D. (Hons.) degree from Carlos III University of Madrid, Spain, in 2002 and 2007, respectively. In 2007, he became a Faculty of the Department of Signal Theory and Communications, King Juan Carlos University, Madrid, Spain, where he currently develops his research and teaching activities as a Full Professor and the Director of the Data Science and Signal Processing for Networks research group. From 2005 to 2015, he held different visiting positions with the University of Minnesota, Minneapolis. In 2015, 2016, and 2017, he was a Visitor Scholar with the University of Pennsylvania, Philadelphia. Since 2025, he has been serving as a Co-Director of the ELLIS Madrid Unit. His research focuses on signal processing, machine learning, data science and artificial intelligence over graphs, and nonlinear and stochastic optimization of wireless and transportation networks. He has served the IEEE in a number of posts, including as an Associate Editor and the Technical/General Chair of different conferences, and currently, he is a Senior Area Editor of IEEE Transactions on Signal Processing. His work has been awarded in several journals, conferences, and workshops, with recent ones including the IEEE SPS IEEE Y.A. Best Paper Award 2020 and the IEEE SPS IEEE Best Paper Award 2024. He was the recipient of the "2020 EURASIP Early Career Award" and a member of EURASIP and the ELLIS society.

Daniel P. Palomar (Fellow, IEEE) received the bachelor's degree in electrical engineering and the Ph.D. degree from the Technical University of Catalonia (UPC), Barcelona, Spain, in 1998 and 2003, respectively. Currently, he is a Professor with the Department of Electronic and Computer Engineering and the Department of Industrial Engineering and Decision Analytics, Hong Kong University of Science and Technology (HKUST), Hong Kong, which he joined in 2006. He was a Fulbright Scholar with Princeton University from 2004 to 2006. He had previously held several research appointments, namely, at King's College London (KCL), London, U.K.; Stanford University, Stanford, CA, USA; Telecommunications Technological Center of Catalonia (CTTC), Barcelona, Spain; Royal Institute of Technology (KTH), Stockholm, Sweden; University of Rome "La Sapienza," Rome, Italy; and Princeton University, Princeton, NJ. His research interests include data analytics, optimization methods, and deep learning in financial systems. He is a EURASIP Fellow. He was a recipient of a 2004/06 Fulbright Research Fellowship, in 2004, 2015, and 2020 (co-author) Young Author Best Paper Awards by the IEEE Signal Processing Society, the 2015-16 HKUST Excellence Research Award, the 2002/03 best Ph.D. prize in Information Technologies and Communications by the Technical University of Catalonia (UPC), the 2002/03 Rosina Ribalta first prize for the Best Doctoral Thesis in Information Technologies and Communications by the Epson Foundation, and the 2004 prize for the best Doctoral Thesis in Advanced Mobile Communications by the Vodafone Foundation and COIT. He has been a Guest Editor of IEEE JOURNAL OF SELECTED TOPICS IN SIGNAL PROCESSING 2016 Special Issue on "Financial Signal Processing and Machine Learning for Electronic Trading," an Associate Editor of IEEE TRANSACTIONS ON INFORMATION THEORY and IEEE TRANSACTIONS ON SIGNAL PROCESSING, a Guest Editor of *IEEE Signal Processing Magazine* 2010 Special Issue on "Convex Optimization for Signal Processing," *IEEE Journal on Selected Areas in Communications* 2008 Special Issue on "Game Theory in Communication Systems," and *IEEE Journal on Selected Areas in Communications* 2007 Special Issue on "Optimization of MIMO Transceivers for Realistic Communication Networks."