Graph Learning from Data under Structural and Laplacian Constraints
Hilmi E. Egilmez, Eduardo Pavez, and Antonio Ortega

Abstract—Graphs are fundamental mathematical structures used in various fields to represent data, signals and processes. In this paper, we propose a novel framework for learning/estimating graphs from data. The proposed framework includes (i) formulation of various graph learning problems, (ii) their probabilistic interpretations and (iii) associated algorithms. Specifically, graph learning problems are posed as estimation of graph Laplacian matrices from some observed data under given structural constraints (e.g., graph connectivity and sparsity level). From a probabilistic perspective, the problems of interest correspond to maximum a posteriori (MAP) parameter estimation of Gaussian-Markov random field (GMRF) models, whose precision (inverse covariance) is a graph Laplacian matrix. For the proposed graph learning problems, specialized algorithms are developed by incorporating the graph Laplacian and structural constraints. The experimental results demonstrate that the proposed algorithms outperform the current state-of-the-art methods in terms of accuracy and computational efficiency.

Index Terms—Graph learning, sparse graph learning, graph estimation, optimization, graph Laplacian matrices, Gaussian Markov random fields (GMRFs).

I. INTRODUCTION

GRAPHS are generic mathematical structures consisting of sets of vertices and edges, which are used for modeling pairwise relations (edges) between a number of objects (vertices). In practice, this representation is often extended to weighted graphs, for which a set of scalar values (weights) are assigned to edges and potentially to vertices. Thus, weighted graphs offer general and flexible representations for modeling affinity relations between the objects of interest.

Many practical problems can be represented using weighted graphs. For example, a broad class of combinatorial problems such as weighted matching, shortest-path and network-flow [2] are defined using weighted graphs. In signal/data-oriented problems, weighted graphs provide concise (sparse) representations for robust modeling of signals/data [3]. Such graph-based models are also useful for analyzing and visualizing the relations between their samples/features. Moreover, weighted graphs naturally emerge in networked data applications, such as learning, signal processing and analysis on computer, social, sensor, energy, transportation and biological networks [4], where the signals/data are inherently related to a graph associated with the underlying network. Similarly, image and video signals can be modeled using weighted graphs whose weights capture the correlation or similarity between neighboring pixel values (such as in nearest-neighbor models) [5]–[10].

In practice, many datasets typically consist of an unstructured list of samples, where the associated graph information (representing the structural relations between samples/features) is latent. For analysis, processing and algorithmic purposes, it is often useful to build graph-based models that provide a concise/simple explanation for datasets and also reduce the dimension of the problem by exploiting the available prior knowledge/assumptions about the desired graph (e.g., structural information including connectivity and sparsity level).

The focus of this paper is on learning graphs (i.e., graph-based models) from data, where the basic goal is to optimize a weighted graph with nonnegative edge weights characterizing the affinity relationship between the entries of a signal/data vector based on multiple observed vectors. For this purpose, we propose a general framework where graph learning is formulated as the estimation of different types of graph Laplacian matrices from data. Specifically, our formulations involve, for a given data statistic $S$ (e.g., empirically obtained covariance/kernel matrices) and a symmetric regularization matrix $H$, minimization of objective functions of the following form:

$$\frac{\text{Tr} (\Theta S) - \log \det (\Theta) + \| \Theta \odot H \|_1}{D(\Theta, S)} + R(\Theta, H),$$

where $\Theta$ denotes the $n \times n$ target matrix variable, $R(\Theta, H)$ is the sparsity promoting weighted $\ell_1$-regularization term [11] multiplying $\Theta$ and $H$ element-wise, and $D(\Theta, S)$ is the data-fidelity term, a log-determinant Bregman divergence [12], whose minimization corresponds to the maximum likelihood estimation of inverse covariance (precision) matrices for multivariate Gaussian distributions. Thus, minimizing (1) for arbitrary data can be interpreted as finding the parameters of a multivariate Gaussian model that best approximates the data [13][14]. In addition to the objective in (1), our formulations incorporate problem-specific Laplacian and structural constraints depending on (i) the desired type of graph Laplacian and (ii) the available structural information about the graph structure. Particularly, we consider three types of graph Laplacian matrices, namely, generalized graph Laplacians (GGLs), diagonally dominant generalized graph Laplacians (DDGLs), and combinatorial graph Laplacians (CGLs), and develop novel techniques to estimate them from data (i.e., data statistic $S$). As illustrated in Fig. 1 and further discussed in Section IV, the proposed graph Laplacian estimation techniques can also be viewed as methods to learn different classes of Gaussian-Markov random fields (GMRFs) [15] [16], whose precision...
matrices are graph Laplacians. Moreover, in our formulations, structural (connectivity) constraints are introduced to exploit available prior information about the target graph. When graph connectivity is unknown, graph learning involves estimating both graph structure and graph weights, with the regularization term controlling the level of sparsity. Otherwise, if graph connectivity is given (e.g., based on application-specific assumptions or prior knowledge), graph learning reduces to the estimation of graph weights only.

Graph Laplacian matrices have multiple applications in various fields. In spectral graph theory [17], basic properties of graphs are investigated by analyzing characteristic polynomials, eigenvalues and eigenvectors of the associated graph Laplacian matrices. In machine learning, graph Laplacians are extensively used as kernels, especially in spectral clustering [18] and graph regularization [19] tasks. Moreover, in graph signal processing [4], basic signal processing operations such as filtering [10] [20], sampling [21], transformation [9] [22] are extended to signals defined on graphs associated with Laplacian matrices. Although the majority of studies and applications primarily focus on CGLs (and their normalized versions) [17] [23], which represent graphs with zero vertex weights (i.e., graphs with no self-loops), there are recent studies where GGLs [24] (i.e., graphs with nonzero vertex weights) are shown to be useful. Particularly, GGLs are proposed for modeling image and video signals in [8] [9], and their potential machine learning applications are discussed in [25]. In [26], a Kron reduction procedure is developed based on GGLs for simplified modeling of electrical networks. Furthermore, DDGLs are utilized in [27]–[29] to develop efficient algorithms for graph partitioning [27], graph sparsification [28] and solving linear systems [29]. Our work complements these methods and applications by proposing efficient algorithms for estimation of these three types of graph Laplacians from data.

In the literature, several approaches have been proposed for estimating graph-based models. Dempster [30] originally proposed the idea of introducing zero entries in inverse covariance matrices for simplified covariance estimation. Later, a neighborhood selection approach was proposed for graphical model estimation [31] by using the Lasso algorithm [32]. Friedman et al. [33] formulated a regularization framework for sparse inverse covariance estimation and developed the Graphical Lasso algorithm to solve the regularized problem. Some algorithmic extensions of the Graphical Lasso are discussed in [13], [34], and a few computationally efficient variations are presented in [35]–[37]. However, inverse covariance estimation methods, such as the Graphical Lasso, search for solutions in the set of the positive semidefinite matrices ($\mathcal{M}_{psd}$ in Fig. 1), which lead to a different notion of graphs by allowing both negative and positive edge weights, while we focus on learning graphs with nonnegative edge weights, associated with graph Laplacian matrices ($\mathcal{L}_g$, $\mathcal{L}_d$ or $\mathcal{L}_c$ in Fig. 1). Although graph Laplacian matrices represent a more restricted set of models (attractive GMRFs) compared to positive semidefinite matrices (modeling general GMRFs), attractive GMRFs cover an important class of random vectors whose entries can be optimally predicted by nonnegative linear combinations of the other entries. For this class of signals/data, our proposed algorithms incorporating Laplacian constraints provide more accurate graph estimation than sparse inverse covariance methods (e.g., Graphical Lasso). Even when such model assumptions do not strictly hold, the proposed algorithms can be employed to find the best (closest) graph Laplacian fit with respect to the Bregman divergence in (1) for applications where graph Laplacians are useful (e.g., in [4], [9], [17]–[29]).

This paper proposes a general optimization framework for estimating graph Laplacian matrices by (i) introducing new problem formulations with Laplacian and structural (i.e., connectivity) constraints and (ii) developing novel algorithms specialized for the proposed problems, which provide more accurate and efficient graph estimation compared to the current state-of-the-art approaches.

Several recent publications address learning of different types of graph Laplacians from data. Closest to our work, Slawski and Hein address the problem of estimating symmetric M-matrices [38], or equivalently GGLs, and propose an efficient primal algorithm [39], while our recent work [40] proposes an alternative dual algorithm for GGL estimation. Our work in this paper addresses the same GGL estimation problem as [39] [40], based on a primal approach analogous to that of [39], but unlike both [39] and [40], we incorporate connectivity constraints in addition to sparsity promoting regularization. For estimation of CGLs, Lake and Tenenbaum [41] also consider minimization of the objective function in (1), which is unbounded for CGLs (since they are singular matrices). To avoid working with singular matrices, they propose to optimize a different target matrix obtained by adding a positive constant value to diagonal entries of a combinatorial Laplacian, but no efficient algorithm is developed. Dong et al. [42] and Kalofolias [43] propose minimization of two objective functions different from (1) in order to overcome issues related to the singularity of CGLs. Instead, by restricting our learning problem to connected graphs (which have exactly one eigenvector with eigenvalue 0), we can directly use a modified version of (1) as the objective function and develop an efficient algorithm that guarantees convergence to the optimal solution, with significant improvements in experimental results over prior work.

There are also a few recent studies that focus on inferring graph topology (i.e., connectivity) information from signals.
assumed to be diffused on a graph. Particularly, Segarra et al. [44] and Pasdeloup et al. [45] focus on learning graph shift/diffusion operators (such as adjacency and Laplacian matrices) from a set of observed graph signals, and Sardellitti et al. [46] propose an approach to estimate a graph Laplacian from bandlimited graph signals. None of these works [44]–[46] considers the minimization of (1). In fact, techniques proposed in these papers directly use the eigenvectors of the empirical covariance matrix and only optimize the choice of eigenvalues of the Laplacian or adjacency matrices under specific criteria, for the given eigenvectors. In contrast, our methods implicitly optimize both eigenvectors and eigenvalues by minimizing (1). Addressing specifically the problem of learning diffusion-based models as in [44]–[46] is out of the scope of this paper and will be considered in our future work.

Summary of contributions. We address estimation of three different types of graph Laplacian with structural constraints. To the best of our knowledge, the estimation of DDGLs is considered for the first time in this paper. For CGL estimation, we propose a novel formulation for the objective function in (1), whose direct minimization is not possible due to the singularity of CGLs. Our formulation allows us to improve the accuracy of CGL estimation significantly compared to the approaches in [41]–[43]. For GGL estimation, the prior formulations in [39]–[40] are extended in order to accommodate structural constraints. To solve the proposed problems, we develop efficient block-coordinate descent (BCD) algorithms [47] exploiting the structural constraints within the problems, which can significantly improve the accuracy and reduce the computational complexity depending on the degree of sparsity introduced by the constraints. Moreover, we theoretically show that the proposed algorithms guarantee convergence to the optimal solution. Previously, numerous BCD-type algorithms are proposed for sparse inverse covariance estimation [13]–[33] which iteratively solve an $\ell_1$-regularized quadratic program. However, our algorithms are specifically developed for graph Laplacian estimation problems, where we solve a nonnegative quadratic program for block-coordinate updates. Finally, we present probabilistic interpretations of our proposed problems by showing that their solutions lead to optimal parameter estimation for special classes of GMRFs, as depicted in Fig. 1. While recent work has noted the relation between graph Laplacians and GMRFs [7]–[48], the present paper provides a more comprehensive classification of GMRFs and proposes specific methods for estimation of their parameters.

The rest of the paper is organized as follows. In Section II, we present the notations and basic concepts used throughout the paper. Section III formulates our proposed problems and summarizes some of the related formulations in the literature. Section IV discusses the probabilistic interpretation of our proposed problems. In Section V, we derive necessary and sufficient optimality conditions and develop novel algorithms for the proposed graph learning problems. Experimental results are presented in Section VI, and some concluding remarks are discussed in Section VII. Our proofs for the propositions stated throughout the paper are included in the appendix.
Weighted graphs can be represented by adjacency, degree and self-loop matrices, which are used to define graph Laplacian matrices. Moreover, we use connectivity matrices to incorporate structural constraints in our formulations. In the following, we present formal definitions for these matrices.

**Definition 3** (Algebraic representations of graphs). For a given weighted graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, f_u, f_v)$ with $n$ vertices, $v_1, \ldots, v_n$:

- The **adjacency matrix** of $\mathcal{G}$ is an $n \times n$ symmetric matrix, $W$, such that $(W)_{ij} = (W)_{ji} = f_u((v_i, v_j))$ for $(v_i, v_j) \in \mathcal{E}$.
- The **degree matrix** of $\mathcal{G}$ is an $n \times n$ diagonal matrix, $D$, with entries $(D)_{ii} = \sum_{j=1}^{n} (W)_{ij}$ and $(D)_{ij} = 0$ for $i \neq j$.
- The **self-loop matrix** of $\mathcal{G}$ is an $n \times n$ diagonal matrix, $V$, with entries $(V)_{ii} = f_v(v_i)$ for $i = 1, \ldots, n$ and $(V)_{ij} = 0$ for $i \neq j$. If $\mathcal{G}$ is a simple weighted graph, then $V = \mathbf{0}$.
- The **connectivity matrix** of $\mathcal{G}$ is an $n \times n$ matrix, $A$, such that $(A)_{ij} = 1$ if $(W)_{ij} \neq 0$, and $(A)_{ij} = 0$ if $(W)_{ij} = 0$ for $i, j = 1, \ldots, n$, where $W$ is the adjacency matrix of $\mathcal{G}$.
- The **generalized graph Laplacian** of a weighted graph $\mathcal{G}$ is defined as $L = D - W + V$.
- The **combinatorial graph Laplacian** of a simple weighted graph $\mathcal{G}$ is defined as $L = D - W$.

**Definition 4** (Diagonally Dominant Matrix). An $n \times n$ matrix $\Theta$ is diagonally dominant if $|\Theta_{ii}| \geq \sum_{j \neq i} |\Theta_{ij}| \forall i$, and it is strictly diagonally dominant if $|\Theta_{ii}| > \sum_{j \neq i} |\Theta_{ij}| \forall i$.

Note that any weighted graph with positive edge weights (see Definitions 1 and 2) can be represented by a generalized graph Laplacian, while simple weighted graphs lead to combinatorial graph Laplacians, since they have no self-loops (i.e., $V = \mathbf{0}$). Moreover, if a weighted graph has nonnegative vertex weights (i.e., $V \geq \mathbf{0}$), its generalized Laplacian matrix is also diagonally dominant.

Based on the definitions above, the sets of graph Laplacian matrices considered in this paper can be written as

- $\mathcal{L}_g = \{ L | L \geq \mathbf{0}, (L)_{ij} \leq 0 \text{ for } i \neq j \}$,
- $\mathcal{L}_d = \{ L | L \geq \mathbf{0}, (L)_{ij} \leq 0 \text{ for } i \neq j, L1 \geq \mathbf{0} \}$,
- $\mathcal{L}_c = \{ L | L \geq \mathbf{0}, (L)_{ij} \leq 0 \text{ for } i \neq j, L1 = \mathbf{0} \}$,

which are illustrated in Fig. 1.

**III. PROBLEM FORMULATIONS FOR GRAPH LEARNING**

**A. Proposed Formulations: Graph Laplacian Estimation**

Assume that we are given a $k \times n$ data matrix $X$ (where each column corresponds to a vertex), and the goal is to estimate a graph Laplacian matrix from a data statistic, $S$. Depending on the application and underlying statistical assumptions, $S$ may stand for the sample covariance of $X$ or a kernel matrix $S = K(X, X)$ derived from data, where $K$ is a positive definite kernel function (e.g., polynomial and RBF kernels). For the purpose of graph learning, we formulate three different optimization problems for a given $S$, a connectivity matrix $A$ and a regularization matrix $H$. In our problems, we minimize the function in (1) under the following set of constraints defined based on a given set of graph Laplacians $\mathcal{L}$ and a connectivity matrix $A$.

$$\mathcal{L}(A) = \left\{ \Theta \in \mathcal{L} | (\Theta)_{ij} \leq 0 \text{ if } (A)_{ij} = 1 \right\} \cup \left\{ (\Theta)_{ij} = 0 \text{ if } (A)_{ij} = 0 \text{ for } i \neq j \right\}.$$  \hspace{1cm} (2)

Based on the nonpositivity constraints on $\Theta$ (i.e., nonnegativity of edge weights), a regularization matrix $H$ can be selected such that $T(\Theta, H)$ term in (1) is compactly written as

$$||\Theta \circ H||_1 = \text{Tr}(\Theta H).$$  \hspace{1cm} (3)

For example, the following standard $\ell_1$-regularization terms with parameter $\alpha$ can be written in the above form as,

$$\alpha||\Theta||_1 = \text{Tr}(\Theta H) \quad \text{where} \quad H = \alpha(2\mathbf{I} - 11'),$$  \hspace{1cm} (4)

$$\alpha||\Theta||_1,\text{off} = \text{Tr}(\Theta H) \quad \text{where} \quad H = \alpha(1 - 11').$$  \hspace{1cm} (5)

Since the trace operator is linear, we can rewrite the objective function in (1) as

$$\text{Tr}(\Theta K) - \log\det(\Theta) \quad \text{where} \quad K = S + H,$$  \hspace{1cm} (6)

which is the form used in our optimization problems. Note that the nonnegativity of edge weights allows us to transform the nonsmooth function in (1) into the smooth function in (6) by rewriting the regularization term as in (3).

**Problem 1** (GGL Problem). The optimization problem formulated for estimating generalized graph Laplacian (GGL) matrices is

$$\begin{align*}
\text{minimize} & \quad \text{Tr}(\Theta K) - \log\det(\Theta) \\
\text{subject to} & \quad \Theta \in \mathcal{L}_g(A) \\
& \quad \text{where } K = S + H \text{ as in (6), and the set of constraints } \mathcal{L}_g(A) \text{ leads to } \Theta \text{ being a GGL matrix.}
\end{align*}$$  \hspace{1cm} (7)

**Problem 2** (DDGL Problem). The diagonally dominant generalized graph Laplacian (DDGL) estimation problem is formulated as

$$\begin{align*}
\text{minimize} & \quad \text{Tr}(\Theta K) - \log\det(\Theta) \\
\text{subject to} & \quad \Theta \in \mathcal{L}_d(A) \\
& \quad \text{where the additional } \Theta \mathbf{1} \geq 0 \text{ constraint in } \mathcal{L}_d(A) \text{ ensures that all vertex weights are nonnegative, and therefore the optimal solution is a diagonally dominant matrix.}
\end{align*}$$  \hspace{1cm} (8)

**Problem 3** (CGL Problem). The combinatorial graph Laplacian (CGL) estimation problem is formulated as

$$\begin{align*}
\text{minimize} & \quad \text{Tr}(\Theta (K + J)) - \log\det(\Theta + J) \\
\text{subject to} & \quad \Theta \in \mathcal{L}_c(A) \\
& \quad \text{where the objective function involves the pseudo-determinant term } (\Theta), \text{ since the target matrix } \Theta \text{ is singular. However, the problem is hard to solve because of the } (\Theta) \text{ term. To cope with this, we propose to reformulate (9) as the following problem 1,}
\end{align*}$$  \hspace{1cm} (9)

$$\begin{align*}
\text{minimize} & \quad \text{Tr}(\Theta (K + J)) - \log\det(\Theta + J) \\
\text{subject to} & \quad \Theta \in \mathcal{L}_c(A) \\
& \quad \text{where the } \Theta \mathbf{1} = 0 \text{ constraint in } \mathcal{L}_c(A) \text{ guarantees that the solution is a CGL matrix, and } J = u_1 u_1^T = (1/n) 11^T \text{ such that } u_1 \text{ is the eigenvector corresponding to the zero eigenvalue of CGL matrices.}
\end{align*}$$  \hspace{1cm} (10)

**Proposition 1.** The optimization problems stated in (9) and (10) are equivalent.

**Proposition 2.** Problems 1, 2 and 3 are convex optimization problems.
In Problems 1, 2 and 3, prior knowledge/assumptions about the graph structure are built into the choice of $A$, determining the structural constraints. In practice, if the graph connectivity is unknown, then $A$ can be set to represent a fully connected graph, $A = A_{	ext{full}} = 11^T - I$, and the regularization matrix $H$ (or the parameter $\alpha$ for the $\ell_1$-regularizations in (4) and (5)) can be tuned until the desired level of sparsity is achieved.

B. Related Prior Formulations

Sparse Inverse Covariance Estimation [33]. The goal is to estimate a sparse inverse covariance matrix from $S$ by solving:

$$\text{minimize}_{\Theta \succeq 0, \nu \geq 0} \text{Tr} (\Theta S) - \logdet (\Theta) + \alpha \|\Theta\|_1.$$  \hfill (11)

In this paper, we are interested in minimization of the same objective function under Laplacian and structural constraints.

Shifted CGL Estimation [41]. The goal is to estimate a shifted CGL matrix, which is defined by adding a scalar value to diagonal entries of a combinatorial Laplacian matrix:

$$\text{minimize}_{\Theta \succeq 0, \nu \geq 0} \text{Tr} (\Theta S) - \logdet (\Theta) + \alpha \|\Theta\|_1\|.$$  \hfill (12)

subject to $\Theta 1 = 0, (\Theta)_{ij} \leq 0 \ i \neq j$

where $\nu$ denotes the positive scalar (i.e., shift) variable added to diagonal elements of $\Theta$, which is constrained to be a CGL matrix, so that $\Theta$ is the target variable. By solving this problem, a CGL matrix $\widetilde{\Theta}$ is estimated by subtracting the shift variable as $\widetilde{\Theta} = (\Theta - \nu I)$. However, this generally leads to a different solution than our method (see Appendix for the proof of the following proposition).

Proposition 3. The objective functions of Problem 3 and the shifted CGL problem in (12) are different.

Graph Learning from Smooth Signals [42] [43]. The goal is to estimate a CGL from $n$-dimensional signals that are assumed to be smooth with respect to the corresponding graph:

$$\text{minimize}_{\Theta \succeq 0} \text{Tr} (\Theta S) + \alpha_1 \|\Theta\|_F^2\|.$$  \hfill (13)

subject to $\Theta 1 = 0, \text{Tr} (\Theta) = n, (\Theta)_{ii} \leq 0 \ i \neq j$

where the sum of degrees is constrained as $\text{Tr} (\Theta) = n$. This is a limitation that is later relaxed in [43] by introducing the following problem with regularization parameters

$$\text{minimize}_{\Theta \succeq 0} \text{Tr} (\Theta S) + \alpha_1 \|\Theta\|_F^2 + \alpha_2 \sum_{i=1}^n \log ((\Theta)_{ii})\|.$$  \hfill (14)

subject to $\Theta 1 = 0, (\Theta)_{jj} \leq 0 \ i \neq j$

where the constraints in (13) and (14) lead to a CGL solution. The following proposition relates the objective function in (14) with $\alpha_1 = 0$ to the objective in our proposed CGL estimation problem (see Appendix for a proof).

Proposition 4. The objective function in Problem 3 with $\alpha = 0$ is lower-bounded by the objective function in (14) for $\alpha_1 = 0$ and $\alpha_2 = 1$. 

Graph Topology Inference. Various approaches for graph topology (connectivity) inference from data (under diffusion-based model assumptions) have been proposed in [44]–[46]. As the most related to our work, Segarra et al. [44] introduce a sparse recovery problem to infer the graph topology information from the eigenbasis, $U$, associated with a graph shift/diffusion operator. Specifically for CGL estimation, the following problem is formulated:

$$\text{minimize}_{\Theta \succeq 0, \Lambda} \|\Theta\|_1\|.$$  \hfill (15)

subject to $\Theta = U \Lambda U^T$

where the eigenbasis $U$ is the input to the problem, so that the goal is to find the set of eigenvalues (i.e., the diagonal matrix $\Lambda$) minimizing $\|\Theta\|_1$. Note that the problems in [44]–[46] require that $U$ be given (or calculated beforehand), while our goal is to directly estimate a graph Laplacian so that both $U$ and $\Lambda$ are jointly optimized.

IV. Probabilistic Interpretation of Proposed Graph Learning Problems

The proposed graph learning problems can be viewed from a probabilistic perspective by assuming that the data has been sampled from a zero-mean $n$-variate Gaussian distribution $x \sim N(0, \Sigma = \Omega I)$. The zero-mean assumption is made to simplify the notation. Our analysis can be trivially extended to a multivariate Gaussian with nonzero mean.

The proposed graph learning problems can be viewed from a probabilistic perspective by assuming that the data has been sampled from a zero-mean $n$-variate Gaussian distribution $x \sim N(0, \Sigma = \Omega)$. The zero-mean assumption is made to simplify the notation. Our analysis can be trivially extended to a multivariate Gaussian with nonzero mean.
\( \Omega = L \), we can show that there is a one-to-one correspondence (bijection) between different classes of attractive GMRFs and types of graph Laplacian matrices by their definitions, as illustrated in Fig. 1:
- \( L \) is a GGL matrix (\( L \in L_g \)) if and only if \( p(x|L) \) is an attractive GMRF,
- \( L \) is a DDGL matrix (\( L \in L_d \)) if and only if \( p(x|L) \) is an attractive, diagonally dominant GMRF,
- \( L \) is a CGL matrix (\( L \in L_c \)) if and only if \( p(x|L) \) is an attractive, DC-intrinsic GMRF.

Note that, in our characterization, the GMRFs corresponding to CGL matrices are classified as DC-intrinsic GMRFs, which are specific cases of intrinsic GMRFs [15] with no probability density along the direction of the eigenvector \( u_1 = 1/\sqrt{n} \) 1 associated with the zero eigenvalue (\( \lambda_1(L) = 0 \)). On the other hand, if \( L \) is a nonsingular GGL matrix, then \( x \) has a proper (non-degenerate) distribution.

Moreover, for \( \Omega = L \), the quadratic term in the exponent in (16) can be interpreted in terms of edge and vertex weights as
\[
\sum_{i=1}^{n} (V)_{ii} x_i^2 + \sum_{(i,j) \in \mathcal{E}} (W)_{ij} (x_i - x_j)^2
\]
where \( x = [x_1, \ldots, x_n]^T \), \( (W)_{ij} = -(L)_{ij} \), \( (V)_{ii} = \sum_{j \neq i} (L)_{ij} \) and \( \mathcal{E} = \{(i,j) \mid (v_i, v_j) \in E\} \) is the set of index pairs of vertices associated with the edge set \( E \). By inspecting (16) and (20), we can observe that assigning a larger (resp. smaller) edge weight (e.g., to \( (W)_{ij} \)) increases the probability of having a smaller (resp. larger) squared difference between corresponding data points (i.e., \( x_i \) and \( x_j \)). Similarly, assigning a larger (resp. smaller) vertex weight (e.g., to \( (V)_{ii} \)) increases the probability of that the corresponding data point (i.e., \( x_i \)) will have smaller (resp. larger) magnitude. In a graph signal processing context [4], the term in (20) is also known as the graph Laplacian quadratic form, and is used to quantify smoothness of graph signals, with a smaller Laplacian quadratic term (\( x^T L x \)) indicating a smoother signal (\( x \)). In our formulations, the Laplacian quadratic from \( x^T L x \) relates to the trace term in our objective function, which is derived based on the likelihood function for GMRFs as discussed in the following.

The proposed graph learning problems can be probabilistically formulated as parameter estimation for attractive GMRFs from data. Assuming that \( k \) independent, identically distributed samples, \( x_i \) for \( i = 1, \ldots, k \), are obtained from an attractive GMRF with unknown parameters, the likelihood of a random variable \( x_i \) and a candidate graph Laplacian \( L \) can be written as
\[
\prod_{i=1}^{k} p(x_i|L) = (2\pi)^{-\frac{k}{2}} |L|^{-\frac{1}{2}} \prod_{i=1}^{k} \exp \left( -\frac{1}{2} x_i^T L x_i \right).
\]
(21)

Let \( L(w, v) \) be defined by edge weight and vertex weight vectors \( w = [f_w(e_1), \ldots, f_w(e_m)]^T \) and \( v = [f_v(v_1), \ldots, f_v(v_n)]^T \), where \( n \) is the number of vertices, and \( m = n(n-1)/2 \) is the number of all possible (undirected) edges. The maximization of the likelihood function in (21) can be equivalently formulated as minimizing the negative log-likelihood, that is
\[
\hat{L}_{ML} = \arg\min_{L(w,v)} \left\{ -\frac{k}{2} \log |L| - \frac{1}{2} \sum_{i=1}^{k} \Tr (x_i^T L x_i) \right\}
\]
\[
= \arg\min_{L(w,v)} \left\{ \Tr (LS) - \log |L| \right\}_L
\]
(22)
where \( S \) is the sample covariance matrix, and \( \hat{L}_{ML} \) denotes the maximum likelihood estimate of \( L(w,v) \). Moreover, we can derive maximum a posteriori (MAP) estimation problems by incorporating the information known about \( L \) into a prior distribution \( p(L) \) as
\[
\hat{L}_{MAP} = \arg\min_{L(w,v)} \left\{ \Tr (LS) - \log |L| - \log (p(L)) \right\}.
\]
(23)
For example, we can choose the following \( m \)-variate exponential prior for sparse estimation of \( w \),
\[
p(w) = (2\alpha)^m \exp (-2\alpha \|w\|_1) \quad \text{for} \ w \geq 0,
\]
(24)
so that the MAP estimation in (23) can be written as follows:
\[
\hat{L}_{MAP} = \arg\min_{L(w,v)} \left\{ \Tr (LS) - \log |L| - \log (p(w)) \right\}
\]
\[
= \arg\min_{L(w,v)} \left\{ \Tr (LS) - \log |L| - 2\alpha \|w\|_1 \right\}
\]
(25)
where the resulting minimization is equivalent to the objective of our problems with the regularization in (5).

**Proposition 5.** Let the data model be \( x \sim N(0, L^T L) \) as in (16). Then, Problems 1, 2 and 3 are specific instances of the maximum a posteriori estimation problem in (23).

Hence, our graph learning problems can be interpreted as MAP parameter estimation for different classes of attractive GMRFs. Thus, when the data model assumptions are satisfied, solving our problems produces the optimal parameters in MAP sense. Given \( S \), which is obtained from the data, the solution of (23) corresponds to the closest Laplacian in terms of a regularized log-determinant divergence criterion [12]. In practice, in order to capture nonlinear relations between random variables, different types of kernels (e.g., polynomial and RBF kernels) can also be used to construct \( S \).

V. PROPOSED GRAPH LEARNING ALGORITHMS

Problems 1, 2 and 3 can be solved using general purpose solvers such as CVX [51]. However, these solvers generally implement second-order methods that require calculation of a Hessian matrix and are therefore computationally inefficient. Simpler gradient descent algorithms would also be computationally complex, since the full gradient calculation of the objective function involves inverting the current estimate of the Laplacian matrix at each iteration (e.g., see the derivative of (32) in (36)). In order to develop efficient methods, we propose iterative block-coordinate descent algorithms [47], where each iterate (block-coordinate update) is obtained by fixing some of the elements in the set of target variables while updating the rest. Thus, the original problem is decomposed into a series of lower-dimensional subproblems that are relatively easier to solve. Particularly, at each iteration, the update variables are formed by a row/column of the target graph Laplacian matrix (\( \Theta \)), and they are updated by solving the subproblem derived based on the optimality conditions of corresponding Laplacian estimation problem, where the available structural constraints are also incorporated into the subproblem. Basically, to estimate an \( n \times n \) graph Laplacian matrix, our algorithms iteratively update rows/columns of \( \Theta \)
and its inverse \(C\), so that the cycle of \(n\) row/column updates is repeated until convergence is achieved. Also, depending on the type of target Laplacian, our algorithms potentially apply projections to satisfy the Laplacian constraints.

In what follows, we first provide matrix update formulas used to efficiently update entries of \(\Theta\) and \(C\) in our algorithms (Section V-A). Then, Algorithms 1 and 2 are presented with the derivations of subproblems based on the optimality conditions of corresponding graph learning problems\(^3\). Specifically, Algorithm 1 is proposed to solve Problems 1 and 2 (Section V-B), while Algorithm 2 solves Problem 3 (Section V-C). Finally, the convergence and computational complexity of proposed algorithms are analyzed (Section V-D).

A. Matrix Update Formulas

The proposed algorithms exploit the following formulas to update the target variable \(\Theta\) and its inverse \(C\) iteratively.

**Row/column updates.** Updating the \(u\)-th row/column of \(\Theta\) results in updating all the elements in its inverse \(C = \Theta^{-1}\), which can be obtained by the matrix inversion lemma [53],

\[
(P^T \Theta P)^{-1} = \begin{bmatrix} \Theta_u & \theta_u \\ \theta_u^T & \theta_u \end{bmatrix}^{-1} = P^T \Theta P = \begin{bmatrix} C_{uu} & C_{uu} \\ C_{uu}^T & C_{uu} \end{bmatrix}
\]

where \(P\) is the permutation matrix used to rearrange updated and fixed elements in block partitions, so that the submatrix \(\Theta_u\) represents the elements that remain unchanged, while vector \(\theta_u\) and scalar \(\theta_u\) (i.e., \(\theta_u = (\Theta)_{uu}\)) are the \(u\)-th row/columns of \(\Theta\), which are being updated. Based on the block partitions in (26), we can calculate \(C\), using updated \(\theta_u\) and \(\theta_u\), for fixed \(\theta_u\) as follows:

\[
C_u = \Theta_u - \theta_u^T \Theta_u^{-1} \theta_u = \Theta_u - \theta_u^T \Theta_u^{-1} \theta_u,
\]

\[
c_u = -C_u \theta_u^T \theta_u = C_{uu} \theta_u^T \theta_u - C_{uu} \theta_u^T \theta_u = 0,
\]

\[
c_u = \frac{1}{\theta_u - \theta_u^T \Theta_u^{-1} \theta_u},
\]

where \(\Theta_u^{-1}\) can be calculated from partitions of updated \(C\) as:

\[
\Theta_u^{-1} = (C_{uu} - c_u \theta_u \theta_u^T). \tag{30}
\]

**Diagonal updates.** After adding a scalar value \(\nu\) to \((\Theta)_{ii}\), we use the Sherman-Morrison formula [54] to update \(C\) as:

\[
\hat{C} = \Theta^{-1} = (\Theta + \nu \delta_i \delta_i^T)^{-1} = C - \frac{\nu C \delta_i \delta_i^T C}{1 + \nu \delta_i \delta_i^T C}, \tag{31}
\]

where \(\delta_i\) is the vector whose entries are zero, except for its \(i\)-th entry which is equal to one.

B. Generalized Laplacian Estimation

**Derivation of the optimality conditions.** To derive necessary and sufficient optimality conditions, we use Lagrangian duality theory [55], [56], which requires introducing a set of Lagrange multipliers (i.e., dual variables) and a Lagrangian function. For Problems 1 and 2 the Lagrangian functions have the form:

\[
-\logdet(\Theta) + Tr(\Theta K) + Tr(\Theta M), \tag{32}
\]

where \(M\) is the matrix of Lagrange multipliers associated with the constraints. In particular, for Problem 1, \(M = M_1 + M_2\), with multiplier matrices, \(M_1\) and \(M_2\), whose entries are:

\[
\begin{align*}
&M_1(1)_{ij} = (M_1)_{ij} = \begin{cases} 
\mu_{ij}^{(1)} & \text{if } (A)_{ij} = 1, \ i \neq j \\
0 & \text{if } (A)_{ij} = 0, \ i \neq j \\
0 & \text{if } i = j
\end{cases} \quad \text{for } i, j = 1, \ldots, n \quad \text{where } \mu_{ij}^{(1)} \geq 0
\end{align*}
\]

and

\[
\begin{align*}
&M_2(1)_{ij} = (M_2)_{ij} = \begin{cases} 
\mu_{ij}^{(2)} & \text{if } (A)_{ij} = 0, \ i \neq j \\
0 & \text{if } (A)_{ij} = 1, \ i \neq j \\
0 & \text{if } i = j
\end{cases} \quad \text{for } i, j = 1, \ldots, n
\end{align*}
\]

where \(\mu_{ij}^{(2)} \in \mathbb{R}\). For Problem 2, the multiplier matrix is \(M = M_1 + M_2\), and the corresponding optimality conditions include those in (37) as well as:

\[
\Theta \geq 0 \quad (M_{1})_{ij} = 0. \tag{38}
\]

**Subproblems for block-coordinate descent updates.** In our algorithm, we solve instances of the subproblem derived based on the optimality conditions of Problem 1. So, letting \(C = \Theta^{-1}\) and using the conditions in (37), the optimality conditions for the \(u\)-th row/column of \(\Theta\) can be written as:

\[
-c_u + k_u + m_u = 0 \tag{39}
\]

\[
-c_u + k_u = 0 \tag{40}
\]

where the vectors, \(c_u\), \(k_u\) and \(m_u\), and the scalars, \(c_u\), \(k_u\) and \(m_u\), are obtained by partitioning \(C\), \(K\) and \(M\), as in (26) such that \(c_u = (C)_{uu}\), \(k_u = (K)_{uu}\) and \(m_u = (M)_{uu} = 0\). By using the relations in (28) and (29), we can rewrite (39) as:

\[
\Theta_u^{-1} \theta_u c_u + k_u + m_u = 0. \tag{41}
\]

Based on the relations in (37) and (40) the optimality conditions for the \(u\)-th column of \(\Theta\) (i.e., \(\theta_u\)) include:

\[
\Theta_u^{-1} \theta_u k_u + c_u + m_u = 0 \tag{42}
\]

\[
\theta_u k_u \leq 0 \quad \text{if } (a_u) = 1
\]

where \(\theta_u\) and \(a_u\) are obtained by obtaining \(\Theta\) and \(A\) as in (26), respectively, and the optimality conditions on \(m_u\) follow

---

\(3\)The implementations of our proposed algorithms are available in [52].
from (33) and (34). Based on the above optimality conditions, in (37) and (42), the optimal update for the u-th row/column of \( \Theta \) (i.e., \( \theta_u \)) corresponds to the solution of the following quadratic program:

\[
\min_{\theta_u} \frac{1}{2} \theta_u^T \Theta^{-1} \theta_u + c_u \theta_u^T k_u \\
\text{subject to } (\theta_u)_i \leq 0 \text{ if } (a_u)_i = 1 \quad (\theta_u)_i = 0 \text{ if } (a_u)_i = 0
\]

The above problem can be simplified by eliminating its equality constraints determined by \( A \) (i.e., \( a_u \)), so that we formulate an equivalent version of (43) as the following nonnegative quadratic program [57], whose solution satisfies the optimality conditions in (42),

\[
\min_{\beta} \frac{1}{2} \beta^T Q \beta - \beta^T p \\
\text{subject to } \beta \geq 0
\]

where \( \beta = - (\theta_u)_S \) \( p = (k_u/k_u)_S \) \( Q = (\Theta^{-1})_{SS} \) \( S = \{ i | (a_u)_i = 1 \} \)

so that \( \beta \) is the vector whose elements are selected from the original variable vector \( \theta_u \) based on index set \( S \). For example, if \( S = \{ 1, 2, 5 \} \), then \( \beta = - ([\theta_u]_1, [\theta_u]_2, [\theta_u]_5) \). Similarly, \( Q \) is constructed by selecting rows and columns of \( \Theta^{-1} \) with index values in \( S \), so the resulting \( Q \) is a submatrix of \( \Theta^{-1} \). It is important to note that the connectivity constraints (i.e., \( A \) or \( a_u \)) allow us to reduce the dimension of the variable \( \theta_u \) and therefore, the dimension of (43).

For Problem 2, based on the conditions in (37) and (38), we can similarly formulate a quadratic program to update u-th row/column of \( \Theta \):

\[
\min_{\theta_u} \frac{1}{2} \theta_u^T \Theta^{-1} \theta_u + c_u \theta_u^T k_u \\
\text{subject to } (\theta_u)_i \leq 0 \text{ if } (a_u)_i = 1 \quad (\theta_u)_i = 0 \text{ if } (a_u)_i = 0 \\
\theta_u^T 1 \leq \theta_u
\]

where \( \theta_u = (\Theta)_u \). The above problem is also a nonnegative quadratic program. To solve (46) for all \( u \), we first iteratively update each row/column of \( \Theta \) by solving the subproblem in (44). After completing a cycle of \( n \) row/column updates, we modify the diagonal entries of the updated \( \Theta \), so that it satisfies the constraints in (46). The diagonal update parameters \( \nu \) are the optimal solutions of the following projection problem for given \( \Theta \):

\[
\min_{\nu} \| \Theta - \hat{\Theta} \|^2_F \\
\text{subject to } \hat{\Theta} = \Theta + \text{diag}(\nu) \quad \hat{\Theta} \in \mathbb{L}_d
\]

where \( \nu \) is the vector of update parameters, and \( \mathbb{L}_d \) denotes the set of diagonally dominant generalized Laplacian matrices.

**Proposed Algorithm.** Algorithm 1 is proposed to solve Problems 1 and 2 for a given connectivity matrix \( A \) of desired Laplacian matrix (i.e., \( L_q \) or \( L_d \)) and regularization matrix \( H \). Basically, the proposed algorithm iteratively updates each row/column of the working estimate of Laplacian matrix \( \Theta \) and its inverse \( \hat{C} \) by solving the subproblem in (44). The main reason of updating \( \hat{C} \) is that the derived subproblem is parametrized by \( \Theta^{-1} \), which depends on \( C \) as formulated in (30). In Algorithm 1, the for loop in lines 5–12 implements the cycle of \( n \) row/column updates, where the update formulas (see lines 7, 9 and 10) are derived based on the relations in (27)–(30). If we are interested in solving Problem 1 (if \( L = L_y \)), then the algorithm skips the lines 13–19. For Problem 2 (for \( L = L_d \)) the for loop between the lines 14–18 iteratively modifies the diagonal elements of \( \Theta \) by solving the projection problem in (47) ensuring that the resulting \( \Theta \) is a diagonally dominant matrix. The inverse of \( \Theta \) (i.e., \( C \)) is also iteratively updated, accordingly (see line 16). The overall procedure is repeated until a stopping criterion (line 20) has been satisfied.

**C. Combinatorial Laplacian Estimation**

**Derivation of the optimality conditions.** Similar to our derivations in the previous subsection, in order to derive optimality conditions, we first define the Lagrangian function corresponding to Problem 3 as follows,

\[
\text{logdet}(\Theta + J) + \text{Tr}(\Theta(K + J)) + \text{Tr}(\Theta M),
\]

where \( M = M_1 + M_2 + M_4 \) consists of Lagrange multipliers associated with the constraints in (10) such that \( M_1 \) and \( M_2 \) are as defined in (33) and (34), and the entries of \( M_4 \) are

\[
(M_4)_{ij} = \mu_i^{(4)} + \mu_j^{(4)} \quad \text{where } \mu_i^{(4)} \in \mathbb{R}, \mu_j^{(4)} \in \mathbb{R}
\]

for \( i, j = 1, \ldots, n \). Based on the Lagrangian stated in (48), the necessary and sufficient optimality conditions for the problem

**Algorithm 1 Generalized Graph Laplacian (GGL)**

**Input:** Sample statistic \( S \), connectivity matrix \( A \), regularization matrix \( H \), target Laplacian set \( L \) and tolerance \( \epsilon \)

**Output:** \( \Theta \) and \( C \)

1. Set \( K = S + H \)
2. Initialize \( \hat{C} = \text{ddiag}(K) \) and \( \hat{\Theta} = \hat{C}^{-1} \)
3. repeat
   4. Set \( \Theta_{\text{pre}} = \hat{\Theta} \)
   5. for \( u = 1 \) to \( n \) do
      6. Partition \( \Theta, \hat{C}, K \) and \( A \) as in (26) for \( u \)
      7. Update \( \hat{\Theta}_u = \hat{C}_u - \hat{c}_u \hat{C}_u / \hat{c}_u \)
      8. Solve (44) for \( \beta \) with \( Q, p \) and \( S \) in (45)
      9. Update \( \theta_u \) and \( \hat{\Theta}_u \) using the solution \( \beta \) from above:
         \( (\hat{\Theta}_u)_S = -\beta (\hat{\Theta}_u)_S \hat{C}_u = 0 \)
         \( \hat{\theta}_u = 1 / k_u + \beta^T Q \hat{\Theta}_u \)
      10. Update \( \hat{c}_u, \hat{c}_u, \hat{c}_u \):
          \( \hat{c}_u = 1 / (\hat{\theta}_u - \hat{\Theta}_u \hat{c}_u) \)
          \( \hat{c}_u = \hat{\Theta}_u \hat{\theta}_u / \hat{c}_u \)
          \( \hat{\Theta}_u = \hat{\Theta}_u + \hat{c}_u \hat{c}_u / \hat{c}_u \)
      11. Rearrange \( \hat{\Theta} \) and \( \hat{C} \) using \( P \) for \( u \) as in (26)
   end for
   12. if \( L = L_y \) (target Laplacian is a DDGL) then
      13. for \( i = 1 \) to \( n \) do
         14. if \( (\Theta)_i < 0 \) then \( \nu = - (\Theta)_i \)
         15. Set \( (\Theta)_i = (\Theta)_i + \nu \) and update \( \hat{C} \) using (31)
      end if
   end for
   16. end if
   17. end for
   18. end if
   19. end for
   20. until criterion\( \hat{\Theta}, \Theta_{\text{pre}} \) \( \leq \epsilon \)
   21. return \( \Theta = \hat{\Theta} \) and \( C = \hat{C} \)
Algorithm 2 Combinatorial Graph Laplacian (CGL)

Input: Sample statistic $S$, connectivity matrix $A$, regularization matrix $H$ and tolerance $\epsilon$

Output: $\Theta$ and $C$

1: Set $J = (1/n)11^T$, $\tilde{K} = S + H + J$
2: Initialize $C = \text{ddiag}(\tilde{K})$ and $\hat{\Theta} = \tilde{C}^{-1}$
3: repeat
4: Set $\hat{\Theta}_{\text{pre}} = \hat{\Theta}$
5: for $u = 1$ to $n$ do
6: Partition $\hat{\Theta}$, $C$, $\tilde{K}$ and $A$ as in (26) for $u$
7: Calculate $\hat{\Theta}^{-1} = \tilde{C}_u - \tilde{c}_u \tilde{c}_u^T / \tilde{c}_u$
8: Solve (56) for $\beta$ with $Q$, $p$ and $S$ in (57)
9: Update $\theta_u$ and $\hat{\theta}_u$ using the solution $\beta$ from above:
   $$\hat{\theta}_u = 1 / \tilde{K}_u + (\beta - (1/n)1)^T Q(\beta - (1/n)1)$$
10: Update $\tilde{c}_u$, $\tilde{\theta}_u$, $\tilde{C}_u$:
   $$\tilde{c}_u = 1 / (\tilde{\theta}_u - \theta_u \tilde{\theta}_u) \quad \tilde{\theta}_u = \tilde{\Theta}^{-1} \theta_u / \tilde{c}_u$$
   $$\tilde{C}_u = \tilde{C}^{-1}_u + \tilde{c}_u \tilde{c}_u^T / \tilde{c}_u$$
11: Rearrange $\hat{\Theta}$ and $C$ using $P$ for $u$ as in (26)
12: end for
13: for $i = 1$ to $n$ do
14: if $|\hat{\Theta}[i,1] - 1| \neq 0$ then $\nu = -|\hat{\Theta}[i,1]| + 1$
15: Set $(\hat{\Theta})_{ii} = (\hat{\Theta})_{ii} + \nu$ and update $C$ using (31)
16: end if
17: end for
18: until criterion $|\hat{\Theta}_{\text{pre}} - \hat{\Theta}| \leq \epsilon$
19: return $\Theta = \hat{\Theta} - 1$ and $C = \tilde{C} - J$

By changing variables $\tilde{\theta}_u = \theta_u + (1/n)1$, $\tilde{\theta}_u = \theta_u + (1/n)$ and dividing the objective function with $\tilde{c}_u^T$, we rewrite (54) as a quadratic program of the standard form,

$$\begin{align*}
\min_{\theta_u} & \quad \frac{1}{2} \tilde{\theta}_u^T \tilde{\Theta}^{-1} \theta_u + \tilde{\theta}_u^T \left( \frac{\tilde{k}_u}{\tilde{c}_u} + \frac{1}{n} \tilde{\Theta}^{-1} \right) \\
\text{subject to} & \quad (\theta_u)_i \leq 1/n \text{ if } (a_u)_i = 1 \\
& \quad (\theta_u)_i = 1/n \text{ if } (a_u)_i = 0 \\
& \quad (\theta_u - (1/n)1)^T 1 = \tilde{\theta}_u - (1/n)
\end{align*}$$

In order to solve (55) for all $u$, we first iteratively update each row/column by solving the nonnegative quadratic program in (56). After each cycle of $n$ row/column updates, the diagonal entries of the resulting matrix ($\Theta$) are modified to satisfy the combinatorial Laplacian constraints $- \tilde{\theta}_u 1 = \theta_u$ for $u = 1, ..., n$ in (55). The diagonal update parameters are optimized by solving the following projection problem for given $\Theta$:

$$\begin{align*}
\min_{\nu} & \quad \|\Theta - \hat{\Theta}\|_F \\
\text{subject to} & \quad \hat{\Theta} = \tilde{\Theta} + \text{diag}(\nu) \quad (\hat{\Theta} - J) \in L_c
\end{align*}$$

where $\nu$ is the vector of update parameters, $L_c$ denotes the set of combinatorial Laplacian matrices and $J = (1/n)11^T$.

Proposed Algorithm. Algorithm 2 is proposed to solve Problem 3 for a given connectivity matrix $A$ and regularization matrix $H$. Although the basic structures of Algorithms 1 and 2 are similar, Algorithm 2 has three major differences. Firstly, the algorithm does not directly estimate the target Laplacian matrix (i.e., $\Theta$). Instead, it iteratively solves for the matrix $\tilde{\Theta} = \Theta + J$ whose entries are shifted by $(1/n)$. Second, the subproblem solved for updating each row/column and the associated update formulas are different (see lines 8, 9 and 10 in Algorithm 2). Thirdly, the for loop in lines 13–17 maintains that the estimate of $\Theta$ leads to a CGL matrix (via the $\Theta - J$ transformation) by solving the projection problem in (58).

In Algorithm 2, we propose to estimate the shifted matrix $\Theta$ because of the singularity of combinatorial Laplacian matrices, by their construction. However, our result in Proposition 1 shows that one can solve the CGL problem in (9) by solving the equivalent problem in (10). Based on this result, we derive optimality conditions for (10), and develop Algorithm 2 which iteratively estimates $\Theta$ until the convergence criterion has been satisfied. Then, the optimal $\Theta$ is recovered by subtracting $J$ from the estimated $\Theta$ as in line 19 of Algorithm 2.

D. Convergence and Complexity Analysis of Algorithms

Convergence. The following proposition addresses the convergence of our proposed algorithms based on the results from [47][55][58] (see Appendix for the proof).
Proposition 6. Algorithms 1 and 2 guarantee convergence to the global optimal solutions of Problems 1, 2 and 3.

Complexity. In Algorithms 1 and 2, each block-coordinate descent iteration has \( O(T_p(n) + n^2) \) complexity where \( O(T_p(n)) \) is the worst-case complexity of solving the subproblem with dimension \( n \), and the \( n^2 \) term is due to updating \( \Theta_p^{-1} \). After a cycle of \( n \) row/column updates, updating a diagonal entry of \( \Theta \) and its inverse, \( C \), also has \( O(n^2) \) complexity. Depending on the sparsity of solutions (i.e., graphs) the complexity can be reduced to \( O(T_p(s) + n^2) \) where \( s \) is the maximum number of edges connected to a vertex (i.e., number of non-zero elements in any row/column of \( \Theta \)). Moreover, both proposed algorithms use diagonal matrices to initialize \( \Theta \) and \( C \). In practice, better initializations (i.e., “warm starts”) and randomized implementations [47] can be exploited to reduce the algorithm runtime. To solve the subproblems in (44) and (56), which are specifically nonnegative quadratic programs, we employ an extension of Lawson-Hanson algorithm [59] with a block principal pivoting method [60]. Since nonnegative quadratic programs require varying number of iterations to converge for each row/column update, it is hard to characterize the overall complexity of our algorithms. Yet, the complexity of solving the subproblems is \( T_p(n) = \Omega(n^3) \), which can be significantly reduced if the solutions are sparse \( (T_p(s) = \Omega(s^3) \) for \( s \)-sparse solutions). In Section VI, we present empirical complexity results for the proposed algorithms.

VI. EXPERIMENTAL RESULTS

In this section, we present experimental results for comprehensive evaluation of our proposed graph learning techniques. Firstly, we compare the accuracy of our proposed algorithms and the state-of-the-art methods, where the datasets are artificially generated based on attractive GMRFs. Specifically, our GGL and DDGL estimation methods based on Algorithm 1 (GGL and DDGL) are evaluated by benchmarking against the Graphical Lasso algorithm (Glasso) [33]. Algorithm 2 (CGL) is compared to the methods proposed for estimating shifted CGF matrices (SCGL) [41] and learning graphs from smooth signals (GLS) [42][43], as well as the graph topology inference approach (GTI) proposed in [44]. Secondly, empirical computational complexity results are presented, and the advantage of exploiting connectivity constraints is demonstrated. Thirdly, the proposed methods are applied to learn similarity graphs from a real categorical dataset with binary entries, and the results are visually investigated. Finally, we evaluate the effect of model mismatch and of incorporating inaccurate connectivity constraints on graph Laplacian estimation.

A. Comparison of Graph Learning Methods

Datasets. In order to demonstrate the performance of the proposed algorithms, we create several artificial datasets based on different graph-based models. Then the baseline and proposed algorithms are used to recover graphs (i.e., graph-based models) from the artificially generated data. To create a dataset, we first construct a graph, then its associated Laplacian matrix, \( L \), is used to generate independent data samples from the distribution \( N(0, L) \). A graph (i.e., \( L \)) is constructed in two steps. In the first step, we determine the graph structure (i.e., connectivity) based on one of the following three options (see Fig. 2):

- Grid graph, \( G_{\text{grid}}^{(n)} \): consisting of \( n \) vertices attached to their four nearest neighbors (except the vertices at boundaries).
- Random Erdos-Renyi graph, \( G_{\text{ER}}^{(n,p)} \), with \( n \) vertices attached to other vertices with probability \( p \).
- Random modular graph (also known as stochastic block model), \( G_{\text{M}}^{(n,L_1,L_2)} \) with \( n \) vertices and four modules (sub-graphs) where the vertex attachment probabilities across modules and within modules are \( p_1 \) and \( p_2 \), respectively.

In the second step, the graph weights (i.e., edge and vertex weights) are randomly selected based on a uniform distribution from the interval \([0.1, 3]\), denoted as \( U(0.1, 3) \). Note that this procedure leads to DDGLs, which are used in comparing our Algorithm 1 against GLasso. For evaluation of Algorithm 2, the edge weights are randomly selected from the same distribution \( U(0.1, 3) \), while all vertex weights are set to zero.

Experimental setup. For comprehensive evaluation, we create various experimental scenarios by choosing different graph structures (grid, Erdos-Renyi or modular) with varying number of vertices. Particularly, the GGL, DDGL and Glasso methods are tested on graphs consisting of 64 or 256 vertices. Since SCGL and GTI approaches [41] [44] do not currently have an efficient algorithm, the CGL estimation methods are only...
evaluated on graphs with 36 vertices. We also test the performance of the proposed methods with and without connectivity constraints. For each scenario, Monte-Carlo simulations are performed to test baseline and proposed algorithms with varying number of data samples \((k)\). All algorithms use the following convergence criterion with the tolerance \(\epsilon = 10^{-4}\),

\[
\text{criterion}(\hat{\Theta}, \Theta_{\text{pre}}) = \frac{\|\hat{\Theta} - \Theta_{\text{pre}}\|_F}{\|\Theta_{\text{pre}}\|_F} \leq \epsilon,
\]

which is the relative error between the ground truth graph \((\Theta^*)\) and estimated graph parameters \((\Theta)\), and

\[
\text{FS}(\hat{\Theta}, \Theta^*) = \frac{2tp}{2tp + fn + fp},
\]

is the F-score metric (commonly used metric to evaluate binary classification performance) calculated based on true-positive \((tp)\), false-negative \((fn)\) and false-negative \((fn)\) detection of graph edges in estimated \(\hat{\Theta}\) with respect to the ground truth edges in \(\Theta^*\). F-score takes values between 0 and 1, where the value 1 means perfect classification.

In our experiments, since SCGL and GLasso methods employ \(\alpha|\Theta|\) for regularization, we use the same regularization in our proposed methods (i.e., the matrix \(H\) is selected as in (4) for Problems 1, 2 and 3) to fairly compare all methods. Monte-Carlo simulations are performed for each proposed/baseline method, by successively solving the associated graph learning problem with different regularization parameters (i.e., \(\alpha_1\) and \(\alpha_2\)) to find the (best) regularization minimizing \(\text{RE}\). The corresponding graph estimate is also used to calculate \(\text{FS}\). Particularly, for the GGL, DDGL, CGL, SCGL and GLasso methods, \(\alpha\) is selected from the following set:

\[
\{0\} \cup \{0.75'(s_{\text{max}} \sqrt{\log(n)/k}) | r = 1, 2, 3, \ldots, 14\},
\]

where \(s_{\text{max}} = \max_{ij} |(S)_{ij}|\) is the maximum off-diagonal entry of \(S\) in absolute value, and the scaling term \(\sqrt{\log(n)/k}\) is used for adjusting the regularization according to \(k\) and \(n\) as suggested in [61][62]. For both of the GLS methods [42][43] addressing the problems in (13) and (14), \(\alpha_1\) is selected from \(\{0\} \cup \{0.75's_{\text{max}} | r = 1, 2, 3, \ldots, 14\}\) and for the problem in (14) the parameter \(\alpha_2\) is selected by fine tuning. For GLS [42], the \(\text{Tr}(\Theta) = n\) constraint in (13) is set as \(\text{Tr}(\Theta) = \text{Tr}(\Theta^*)\). Since GLS [43] and GTI [44] approaches generally result in severely biased solutions with respect to the ground truth \(\Theta^*\) (based on our observations from the experiments), their RE values are calculated after normalizing the estimated solution \(\hat{\Theta}\) as \(\hat{\Theta} = (\text{Tr}(\Theta^*)/\text{Tr}(\Theta))\Theta\).

**Results.** Figure 3 demonstrates graph learning performances of different methods (in terms of average RE and FS) with respect to the number of data samples, used to calculate sample covariance \(S\), per number of vertices (i.e., \(k/n\)). In our results, GGL(\(\alpha\)), DDGL(A, \(\alpha\)) and CGL(A, \(\alpha\)) refer to solving the graph learning problems with both connectivity constraints and regularization, where the constraints are determined based on the true graph connectivity. GGL(\(\alpha\)), DDGL(\(\alpha\)) and CGL(\(\alpha\)) refer to solving the problems with \(\ell_1\)-regularization only (i.e., without connectivity constraints).

As shown in Fig. 3, our proposed methods outperform all baseline approaches (namely GLasso [33], SCGL [41], GGL [42][43] and GTI [44]) in terms of both RE and FS metrics. Naturally, incorporating the connectivity constraints (e.g., in GGL(A, \(\alpha\)) and CGL(A, \(\alpha\)) significantly improves the graph learning performance. However, for small \(k/n\), it may not provide a perfect FS, even if the true graph connectivity is given. This is because there may not be a sufficient number of data samples to effectively recover the graph information. Specifically for \(k/n \leq 1\), both S and the estimated graph Laplacian have low-rank. Since low-rank graph Laplacians correspond to disconnected graphs (i.e., graphs with more than one connected components), they can cause false-negative \((fn)\) detections which reduce FS. Furthermore, the proposed methods outperform all baseline approaches regardless of the size of the graph \((n)\) and the connectivity model (A). As an example, for fixed \(k/n = 30\), Table II compares average RE results for different graph connectivity models and number of vertices \((n = 64\) and 256\)). Also, Figs. 4 and 5 illustrate sample GGL and CGL estimation results and compare different methods.

**TABLE II: AVERAGE RELATIVE ERRORS FOR DIFFERENT GRAPH CONNECTIVITY MODELS AND NUMBER OF VERTICES WITH FIXED k/n = 30**

| Connectivity models | Average RE\(n = 64\) | Average RE\(n = 256\) |
|---------------------|----------------------|----------------------|
| GGL(\(\alpha\))     | 0.079 | 0.078 | 0.053 | 0.037 | 0.040 | 0.027 |
| GGL(\(\alpha, 0.1\))| 0.105 | 0.112 | 0.077 | 0.082 | 0.053 | 0.053 |
| GGL(\(\alpha, 0.1, 0.3\)| 0.102 | 0.124 | 0.075 | 0.081 | 0.051 | 0.053 |

For estimation of GGL matrices, Fig. 3a demonstrates that the best set of results are obtained by solving the DDGL problem, DDGL\((A, \alpha)\) and DDGL\((\alpha)\). This is expected since the random data samples are generated based on DDGL matrices (as part of our experimental setup), and exploiting additional information about the type of graph Laplacian improves graph learning performance. Generally, solving the GGL problem, GGL\((A, \alpha)\) and GGL\((\alpha)\), also provides a good performance, where the difference between GGL and DDGL is often negligible. Moreover, both of the proposed GGL\((\alpha)\) and DDGL\((\alpha)\) methods (i.e., Algorithm 1) perform considerably better than GLasso, especially when the number of data samples \((k/n)\) is small, since exploiting Laplacian constraints fulfills the model assumptions of attractive GMRFs. For estimation of CGL matrices, Fig. 3b shows that CGL (i.e., Algorithm 2) provides significantly better performance with increasing number of data samples \((k/n)\) as compared to the SCGL, GGL and GTI approaches. Particularly, SCGL and GLS have limited accuracy even for large number of samples (e.g., \(k/n \geq 100\)). The main reason is due to their problem formulations, where SCGL [41] optimizes a shifted CGL instead of directly estimating a CGL matrix as stated in (12). The GLS and GTI methods solve the problems in (13), (14) and (15), whose objective functions...
to learn weighted graphs, where graph vertices denote animals and edge weights represent the similarity approximately GLasso or 2) and average execution times of the test algorithm (Algorithm 1 \( \bar{T} \) are calculated using as discussed in the previous section. The speedup values and associated datasets (with \( p = 1 \)) are derived without taking multivariate Gaussian distributions into account. Specifically, the objective in (14) serves as a lower-bound for the maximum-likelihood criterion in (22) (see Proposition 4). Besides, the performance difference against GTI is substantial [44] across different \( k/n \), and GTI generally does not converge if \( S \) has low-rank (i.e., \( k/n < 1 \)), so those results are not available.

B. Empirical Results for Computational Complexity

Figure 7 compares the computational speedups achieved by our proposed algorithms over GLasso, which is implemented according to the P-GLasso algorithm presented in [34]. In our experiments, the algorithms are tested on Erdos-Renyi graphs with \( n = 64 \) vertices \( (G_{ER}^{n,0.1}) \). By varying the parameter \( p \), we evaluate the speedups at different graph sparsity levels (\( p = 1 \) means a fully connected graph). For each \( p \), 10 different graphs and associated datasets (with \( k/n = 1000 \)) are generated as discussed in the previous section. The speedup values are calculated using \( \bar{T}_{GLasso}/T \), where \( T \) and \( \bar{T}_{GLasso} \) denote average execution times of the test algorithm (Algorithm 1 or 2) and GLasso algorithm, respectively. Since GLasso is approximately 1.5 times faster than both GLS methods [42] [43] on average, and the other methods [41] [44] do not have efficient implementations, we only use GLasso as the baseline algorithm in this experiment.

As shown in Fig. 7, the proposed methods provide faster convergence than GLasso regardless of \( p \) (i.e., sparsity level), and the computational gain is substantial for learning sparse graphs (e.g., \( p \leq 0.3 \)), where incorporating the connectivity constraints contributes to an additional 2 to 3-fold speedup over the methods without exploiting connectivity constraints. In the worst case, for dense graphs (e.g., \( p \geq 0.8 \)), our methods converge approximately 5 times faster than the GLasso. This is mainly because, at each iteration, GLasso solves an \( \ell_1 \)-regularized quadratic program [43] having a nonsmooth objective function, and it is generally harder to solve compared to the (smooth) nonnegative quadratic program in (44).

C. Illustrative Results on Real Data

We present some illustrative results to demonstrate that the proposed methods can also be useful to represent categorical (non-Gaussian) data. In our experiments, we use the Animals dataset [63] to learn weighted graphs, where graph vertices denote animals and edge weights represent the similarity...
between them. The dataset consists of binary values (0 or 1) assigned to $d = 102$ features for $n = 33$ animals. Each feature corresponds to a true-false question such as, “has lungs?”, “is warm-blooded?” and “lives in groups?”. Using this dataset, the statistic matrix $S$ is calculated as $S = (1/d) XX^T + (1/3) I$ where $X$ is the mean removed $n \times d$ data matrix. The $(1/3) I$ term is added based on the variational Bayesian approximation result in [13] for binary data. Then, $S$ is used as input to
the GLasso, GGL and CGL methods. Here, we only compare
methods minimizing the objective function in (1) which is
shown to be a suitable metric for binary data in [13][14].

Figure 6 illustrates the graphs constructed by using GLasso,
GGL and CGL with different regularization parameters. The
positive and negative edge weights estimated by GLasso are
shown side-by-side in Figs. 6a and 6d, where the magnitudes
of negative weights are substantially smaller than most of the
positive weights. In other words, positive partial correlations
are more dominant than negative partial correlations in the
precision matrix. Since the proposed GGL and CGL find graphs
with nonnegative edge weights (i.e., closest graph Laplacian
projections with no negative edge weights), the corresponding
results are similar to the graphs with nonnegative weights in
Figs. 6a and 6d. The results follow the intuition that larger po-
itive weights should be assigned to connected animals con-
trolled to be similar. Such pairs of animals include (gorilla,chimp),
(dolphin,whale), (dog,wolf) and (robin,finch).

D. Graph Laplacian Estimation under Model Mismatch

In the case of a model mismatch (i.e., when data is not
generated by a GMRF with a graph Laplacian as its precision
matrix), the proposed methods allow us to find the closest
graph Laplacian fit with respect to the original model in the
log-determinant Bregman divergence sense. Fig. 8 illustrates
two examples (simulating sparse and dense mismatches) where
the ground truth precision matrices $\Sigma$ have both negative and
positive edge weights (Fig. 8a). From their true covariances
$\Sigma = \Omega^{-1}$, GLasso recovers the original model by allowing
both positive and negative edge weights (Fig. 8b). On the other
hand, the proposed GGL finds the closest graph Laplacian with
respect to the ground truth (Fig. 8c). As shown in the figure,
GGL maintains all the edges with positive weights and also
introduces additional connectivity due to the negative weights
in the ground truth $\Omega$. Note that this form of projection (based
on log-determinant Bregman divergence) does not necessarily
assign zeros to the corresponding negative edge weights. In
general, the identification of edges with positive weights under
model mismatches is nontrivial, as also pointed out in [39].

E. Graph Laplacian Estimation under Connectivity Mismatch

We now evaluate empirically the effect inaccurate selection of
connectivity matrices ($A$), used to determine the connectiv-
ity constraints in our problems. For this purpose, in our
experiments, we randomly construct multiple $64 \times 64$ GGL
matrices based on Erdos-Renyi graphs with $p = 0.1$ (i.e.,
$G_{ER}^{(64,0.1)}$) and generate datasets associated with the GGLs, as
discussed in Section VI-A. Then, the proposed GGL method is
employed to estimate graphs from generated data using differ-
ent connectivity constraints, whose corresponding connectivity
matrices are obtained by randomly swapping the entries of the
true connectivity matrix to simulate connectivity mismatches.
Specifically, a 5% mismatch is obtained by randomly exchang-
ing 5% of the ones in $A$ with zero entries.
Figure 9 compares the accuracy of GGL estimation with connectivity constraints under different levels (5%, 10%, 15% and 25%) of connectivity mismatch against the GGL estimation without using any connectivity constraints (i.e., GGL(A_\text{full})). The results show that using slightly inaccurate connectivity information can be useful when the number of data samples is small (e.g., \(k/n < 5\)), where the performance is similar to GGL(A) and can outperform GGL(A_\text{full}), even though there is a 25% mismatch (GGL(A_{25\%})). However, the performance difference with respect to GGL(A) increases as the number of data samples increases (e.g., \(k/n > 10\)), where both GGL(A) and GGL(A_\text{full}) performs substantially better. In this case, the graph estimation without connectivity constraints (GGL(A_\text{full})) can be preferred if connectivity information is uncertain.

VII. CONCLUSION

In this work, we have formulated various graph learning problems as estimation of graph Laplacian matrices, and proposed efficient block-coordinate descent algorithms. We have also discussed probabilistic interpretations of our formulations by showing that they are equivalent to parameter estimation of different classes of attractive GMRFs. The experimental results have demonstrated that our proposed techniques outperform the state-of-the-art methods in terms of accuracy and computational efficiency, and both can be significantly improved by exploiting the additional structural (connectivity) information about the problem at hand.

The methods proposed in this paper can be used in applications that involve learning a similarity graph, parameter estimation of graph-based models (e.g., attractive GMRFs) from data. Also, depending on the application, learning specific type of graph Laplacian matrices (generalized, diagonally dominant or combinatorial Laplacian matrices) can also be useful. Since the main focus of the present paper is on formulation of graph learning problems and development of new algorithms, related applications are considered as part of our future work.

APPENDIX: PROOFS FOR PROPOSITIONS

Proof of Proposition 1. The problems in (9) and (10) have the same constraints. To prove their equivalence, we show that their objective functions are also the same. First, note that

\[
\text{Tr}(\Theta (K + J)) = \text{Tr}(\Theta K) + \frac{1}{n} \text{Tr}(\Theta 11^T) = \text{Tr}(\Theta K)
\]

since \(\Theta 1 = 0\) based on the CGL problem constraints. Next, we can write

\[
\log \text{det}(\Theta + 1/n 11^T) = \log \left(\prod_{i=1}^{n} \lambda_i(\Theta + 1/n 11^T)\right)
\]

where \(\lambda_i(\Theta)\) denotes the \(i\)-th eigenvalue of \(\Theta\) in ascending order (\(\lambda_1(\Theta) \leq ... \leq \lambda_n(\Theta)\)). Since the eigenvector corresponding to the first (zero) eigenvalue (i.e., \(\lambda_1(\Theta) = 0\)) is \(u_1 = 1/\sqrt{n} \cdot 1\), by the problem constraints (i.e., by definition of CGL matrices), we have that

\[
\Theta + 1/n 11^T = (\lambda_1(\Theta) + 1/\sqrt{n} \cdot 1) u_1 u_1^T + \sum_{i=2}^{n} \lambda_i(\Theta) u_i u_i^T.
\]

Since the determinant of a matrix is equal to the product of its eigenvalues, from (64) we have

\[
\log \text{det}(\Theta + 1/n 11^T) = \log \left(\prod_{i=2}^{n} \lambda_i(\Theta)\right) = \log |\Theta|.
\]

Therefore, the problems in (9) and (10) are equivalent. \(\square\)

Proof of Proposition 2. The function \(\log \text{det}(\Theta)\) defined over positive semidefinite matrices (\(\Theta \succeq 0\)) is a concave function (see [56] for a proof), and \(\text{Tr}(\cdot)\) is a linear function. Thus, the overall objective function is convex. The graph Laplacian constraints form a convex set. Since we have a minimization of a convex objective function over a convex set, the problems of interest are convex. \(\square\)

Proof of Proposition 3. The optimal \(\nu\) cannot be zero, since the objective function is unbounded for \(\nu = 0\). By assuming that \(\alpha = 0\) (without loss of generality), the objective function in (12) can be decomposed as follows

\[
J(\Theta) + \nu \text{Tr}(S) - \sum_{i=2}^{n} \log \left(1 + \frac{\nu}{\lambda_i(\Theta)}\right) - \log(\nu) \tag{65}
\]

where \(J(\Theta)\) is the objective of Problem 3 with \(H = O\). \(\square\)

Proof of Proposition 4. For \(\alpha_1 = 0\) and \(\alpha_2 = 1\), the objective function in (14) is written as \(\text{Tr}(\Theta S) - \sum_{i=1}^{n} \log((\Theta)_{ii})\). By using Hadamard’s inequality \(\logdet(\Theta) \leq \sum_{i=1}^{n} \log((\Theta)_{ii}) [64]\) and taking the log of both sides, the following bound is obtained

\[
\text{Tr}(\Theta S) - \sum_{i=1}^{n} \log((\Theta)_{ii}) \leq \text{Tr}(\Theta S) - \log \text{det}(\Theta) \tag{66}
\]

where the right-hand side is the objective function in Problems 1, 2 and 3 with \(\alpha = 0\), as desired. \(\square\)

Proof of Proposition 5. With proper choices of the prior \(p(L)\) in (23), the objective function in (1) can be constructed for any \(H\), except the pseudo-determinant term \(|\Theta|\) needed for estimating CGL matrices. For this case, Proposition 1 shows that we can equivalently formulate (23) in the form of Problem 3. The construction in (23)–(25) can be trivially extended for the weighted \(\ell_1\)-regularization. Also, the connectivity and Laplacian constraints in Problems 1, 2 and 3 can be incorporated in a Bayesian setting by choosing spike-and-slab prior and improper prior distributions [65] on \(\nu\) and \(w\), so that spike priors correspond to zero edge weights, and slab priors allow nonnegative edge weights. \(\square\)

Proof of Proposition 6. In minimization of a strictly convex and differentiable function over a convex set, (with a proper selection of the step size) a block-coordinate descent algorithm guarantees convergence to the optimal solution [47], [55]. As stated in Proposition 2, all of our problems of interest are convex. Also, the objective functions are differentiable (see in Section V). Moreover, convergence conditions (see in [55]) require that each block-coordinate update be optimal (i.e., each iteration optimally solves the subproblem associated with selected coordinate directions). In Algorithms 1 and 2, this condition is also satisfied, since nonnegative quadratic programs (subproblems) are derived using optimality conditions, so each of their solutions leads to optimal block-coordinate (row/column) updates. These subproblems are also strictly convex, since the \(Q\) matrix in (44) and (56) is positive definite.
throughout all iterations. To prove this, we use the following Schur complement condition on partitions of $\Theta$ as in (26),

$$\Theta > 0 \iff \Theta_u > 0 \text{ and } \theta_u - \theta^T \Theta_u^{-1} \theta_u > 0. \quad (67)$$

where $\Theta_u$ (and therefore, $\Theta_u^{-1}$) is fixed and positive definite from the previous iteration. Noting that both algorithms initialize $\Theta$ and $C$ as (diagonal) positive definite matrices, $\theta_u - \theta^T \Theta_u^{-1} \theta_u = 1/c^2 > 0$ by (29), then the updated $\Theta$ is also positive definite. Since both algorithms solve for the optimal row/column updates at each iteration, based on the result in [55], this proves convergence to the optimal solution for Problem 1. For Problems 2 and 3, Algorithms 1 and 2 implement a variation of the general projected block-coordinate descent method, whose convergence to the optimal solution is shown in [58]. Also, it is trivial to show that additional updates on diagonal elements (see in (31)) maintain positive definiteness of $\Theta$ and $C$. Therefore, both algorithms guarantee convergence to the optimal solution. □

REFERENCES

[1] H. E. Egilmez, E. Pavez, and A. Ortega, “Graph learning with laplacian constraints: Modeling attractive Gaussian Markov random fields,” in 2016 50th Asilomar Conference on Signals, Systems and Computers, Nov 2016, pp. 1470–1474.

[2] J. Kleinberg and E. Tardos, Algorithm Design. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2005.

[3] P. Buhlmann and S. van de Geet, Statistics for High-Dimensional Data: Methods, Theory and Applications. Springer Publishing Co., Inc., 2011. 

[4] D. Shuman, S. 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 Processing Magazine, vol. 30, no. 3, pp. 83–98, 2013.

[5] A. K. Jain, Fundamentals of Digital Image Processing. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1989.

[6] A. M. Tekalp, Digital Video Processing, 2nd ed. Upper Saddle River, NJ, USA: Prentice Hall Press, 2015.

[7] C. Zhang and D. Florencio, “Analyzing the optimality of predictive transform coding using graph-based models,” IEEE Signal Process. Lett., vol. 20, no. 1, pp. 106–109, 2013.

[8] E. Pavez, H. E. Egilmez, Y. Wang, and A. Ortega, “GTM: Graph template transforms with applications to image coding,” in Picture Coding Symposium (PCS), 2015, May 2015, pp. 199–203.

[9] H. E. Egilmez, Y. H. Chao, A. Ortega, B. Lee, and S. Yea, “GBST: Separable transforms based on line graphs for predictive video coding,” in 2016 IEEE International Conference on Image Processing (ICIP), Sept 2016, pp. 2375–2379.

[10] P. Milanfar, “A tour of modern image filtering: New insights and methods, both practical and theoretical,” IEEE Signal Processing Magazine, vol. 30, no. 1, pp. 106–128, Jan 2013.

[11] E. J. Candès, M. B. Wakin, and S. P. Boyd, “Enhancing sparsity by reweighted $\ell 1$ minimization,” Journal of Fourier Analysis and Applications, vol. 14, no. 5, pp. 877–905, 2008.

[12] I. S. Dhillon and J. A. Tropp, “Matrix nearness problems with Bregman divergences,” SIAM J. Matrix Anal. Appl., vol. 29, no. 4, pp. 1120–1146, Nov. 2007.

[13] O. Banerjee, L. E. Ghaoui, and A. D’aspremont, “Model selection through sparse maximum likelihood estimation for multivariate gaussian or binary data,” Journal of Machine Learning Research, vol. 9, pp. 485–516, 2008.

[14] P.-L. Loh and M. J. Wainwright, “Structure estimation for discrete graphical models: Generalized covariance matrices and their inverses,” The Annals of Statistics, vol. 41, no. 6, pp. 3022–3049, 2013.

[15] H. Rue and L. Held, Gaussian Markov random fields: Theory and applications, 1st ed., ser. Monographs on statistics and applied probability 104. Chapman & Hall/CRC, 2005.

[16] D. Koller and N. Friedman, Probabilistic Graphical Models: Principles and Techniques - Adaptive Computation and Machine Learning. The MIT Press, 2009.

[17] F. R. K. Chung, Spectral Graph Theory. USA: American Mathematical Society, 1997.
[43] V. Kalofolias, “How to learn a graph from smooth signals,” in Proceedings of the 19th International Conference on Artificial Intelligence and Statistics (AISTATS), May 2016, pp. 920–929.

[44] S. Segarra, A. G. Marques, G. Mateos, and A. Ribeiro, “Network topology inference from spectral templates,” CoRR, vol. abs/1608.03008v1, 2016. [Online]. Available: https://arxiv.org/abs/1608.03008v1

[45] B. Pasdeloup, M. Rabbat, V. Gripin, D. Pastor, and G. Mercier, “Characterization and inference of graph diffusion processes from observations of stationary signals,” CoRR, vol. abs/arXiv:1605.02569v3, 2017. [Online]. Available: https://arxiv.org/abs/arXiv:1605.02569v3

[46] S. Sardellitti, S. Barbarossa, and P. D. Lorenzo, “Graph topology inference based on transform learning,” in 2016 IEEE Global Conference on Signal and Information Processing (GlobalSIP), Dec 2016, pp. 356–360.

[47] S. J. Wright, “Coordinate descent algorithms,” Math. Program., vol. 151, no. 1, pp. 3–34, Jun. 2015.

[48] C. Zhang, D. Florencio, and P. A. Chou, “Graph signal processing—A probabilistic framework,” Microsoft Research Technical Report, 2015.

[49] S. Hassan-Moghaddam, N. K. Dhinra, and M. R. Jovanovic, “Topology identification of undirected consensus networks via sparse inverse covariance estimation,” in 2016 IEEE 55th Conference on Decision and Control (CDC), Dec 2016, pp. 4624–4629.

[50] H. E. Egilmez, E. Pavez, and A. Ortega, “Graph learning from data under structural and Laplacian constraints,” CoRR, vol. abs/1611.05181v1, 2016. [Online]. Available: https://arxiv.org/abs/1611.05181v1

[51] M. Grant and S. Boyd, “CVX: Matlab software for disciplined convex programming, version 2.1.” http://cvxr.com/cvx, Mar. 2014.

[52] H. E. Egilmez, E. Pavez, and A. Ortega, “GIL: Graph Laplacian learning package, version 1.0,” https://github.com/STAC-USC/Graph_Learning, 2017.

[53] M. A. Woodbury, Inverting Modified Matrices, ser. Statistical Research Group Memorandum Reports. Princeton, NJ: Princeton University, 1950, no. 42.

[54] J. Sherman and W. J. Morrison, “Adjustment of an inverse matrix corresponding to a change in one element of a given matrix,” The Annals of Mathematical Statistics, vol. 21, no. 1, pp. 124–127, 03 1950.

[55] D. P. Bertsekas, Nonlinear Programming. Belmont, MA: Athena Scientific, 1999.

[56] S. Boyd and L. Vandenberghe, Convex Optimization. New York, NY, USA: Cambridge University Press, 2004.

[57] D. Chen and R. J. Plemmons, “Nonnegativity constraints in numerical analysis,” in The Birth of Numerical Analysis. Singapore: World Scientific Publishing, 2010, pp. 109–140.

[58] A. Beck and L. Teboulle, “On the convergence of block coordinate descent methods,” SIAM Journal on Optimization, vol. 23, no. 4, pp. 2037–2060, 2013.

[59] C. Lawson and R. Hanson, Solving Least Squares Problems. Society for Industrial and Applied Mathematics, 1995.

[60] L. F. Portugal, J. J. Júdice, and L. N. Vicente, “A comparison of block pivoting and interior-point algorithms for linear least squares problems with nonnegative variables,” Math. Comput., vol. 63, no. 208, pp. 625–643, Oct. 1994.

[61] S. Zhou, P. Rutimann, M. Xu, and P. Buhlmann, “High-dimensional covariance estimation based on Gaussian graphical models,” J. Mach. Learn. Res., vol. 12, pp. 2975–3026, Nov. 2011.

[62] P. Ravikumar, M. Wainwright, B. Yu, and G. Raskutti, “High-dimensional covariance estimation by minimizing l1-penalized log-determinant divergence,” Electronic Journal of Statistics (EJS), vol. 5, 2011.

[63] C. Kemp and J. B. Tenenbaum, “The discovery of structural form,” Proceedings of the National Academy of Sciences, vol. 105, no. 31, pp. 10 687–10 692, 2008.

[64] T. M. Cover and J. A. Thomas, “Determinant inequalities via information theory,” SIAM J. Matrix Anal. Appl., vol. 9, no. 3, pp. 384–392, Nov. 1988.

[65] H. Ishwaran and J. S. Rao, “Spike and slab variable selection: Frequentist and bayesian strategies,” The Annals of Statistics, vol. 33, no. 2, pp. 730–773, 2005.