GRAPH-BASED MODELS AND TRANSFORMS FOR SIGNAL/DATA PROCESSING WITH APPLICATIONS TO VIDEO CODING

A Dissertation Presented to the
FACULTY OF THE USC GRADUATE SCHOOL
UNIVERSITY OF SOUTHERN CALIFORNIA
In Partial Fulfillment of the
Requirements for the Degree
DOCTOR OF PHILOSOPHY
(ELECTRICAL ENGINEERING)

Hilmi Enes Eğilmez

December 2017

To my family...

Aileme...

Acknowledgments

I would like to express my sincere gratitude towards my advisor, Professor Antonio Ortega, for his constant guidance, support and understanding during the course of my studies at the University of Southern California (USC). It has been a pleasure and privilege to work with Professor Ortega who helped me grow as an independent researcher. This thesis would not be possible without his help and support. Also, I would like to thank Professor C.-C. Jay Kuo and Professor Jinchi Lv for serving on my dissertation committee, as well as Professor Salman Avestimehr and Professor Mahdi Soltanolkotabi for being in my qualifying exam committee. Their comments and suggestions have definitely helped improve this thesis.

I thank all my friends in the Signal Transformation, Analysis and Compression Group, especially, Sungwon Lee, Amin Rezapour, Akshay Gadde, Aamir Anis, Eduardo Pavez, Yung-Hsuan (Jessie) Chao, Jiun-Yu (Joanne) Kao and Yongzhe Wang, for making our research environment enjoyable through both technical and non-technical discussions. Special thanks goes to Eduardo Pavez, Yung-Hsuan (Jessie) Chao and Yongzhe Wang, with whom I collaborated on different projects. I acknowledge their inputs and contributions which shaped parts of this thesis.

Moreover, I would like to express my thanks to Dr. Amir Said, Dr. Onur G. Güleryüz and Dr. Marta Karczewicz for mentoring me during my several summer internships where we had detailed discussions on video coding. I also thank to Professor A. Murat Tekalp for advising my M.Sc. studies at Koc University prior to joining USC as a Ph.D. student.

Furthermore, I am thankful to my brother M. Bedi Eğilmez and my friends, C. Ümit Baş, Enes Özel, C. Ozan Oğuz, Ezgi Kantarcı, Utku Baran, Ahsan Javed, Vanessa Landes and Daoud Burghal for making “life in LA” even more fun.

Last but not least, I am deeply grateful to my family for their love, support and encouragement. Particularly, I thank to my wife İlknur whose love, sacrifice and patience motivated me throughout my Ph.D. studies. Also, none of this would be possible without my parents, Reyhan and Hulusi Eğilmez, to whom I cannot thank enough.

Contents

List of Tables

List of Figures

Abstract

Graphs are fundamental mathematical structures used in various fields to represent data, signals and processes. Particularly in signal processing, machine learning and statistics, structured modeling of signals and data by means of graphs is essential for a broad number of applications. In this thesis, we develop novel techniques to build graph-based models and transforms for signal/data processing, where the models and transforms of interest are defined based on graph Laplacian matrices. For graph-based modeling, various graph Laplacian estimation problems are studied. Firstly, we consider estimation of three types of graph Laplacian matrices from data and develop efficient algorithms by incorporating associated Laplacian and structural constraints. Then, we propose a graph signal processing framework to learn graph-based models from classes of filtered signals, defined based on functions of graph Laplacians. The proposed approach can also be applied to learn diffusion (heat) kernels, which are popular in various fields for modeling diffusion processes. Additionally, we study the problem of multigraph combining, which is estimating a single optimized graph from multiple graphs, and develop an algorithm. Finally, we propose graph-based transforms for video coding and develop two different techniques, based on graph learning and image edge adaptation, to design orthogonal transforms capturing the statistical characteristics of video signals. Theoretical justifications and comprehensive experimental results for the proposed methods are presented.

Chapter 1

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). They provide a natural abstraction by representing the objects of interest as vertices and their pairwise relations as edges. 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 [1] are defined using weighted graphs. In signal/data-oriented problems, weighted graphs provide concise (sparse) representations for robust modeling of signals/data [2]. 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 [3], 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) [4, 5, 6, 7, 8, 9]. Furthermore, in graph signal processing [3], weighted graphs provide useful spectral representations for signals/data, referred as *graph signals*, where graph Laplacian matrices are used to define basic operations such as transformation [8, 10], filtering [9, 11] and sampling [12] for graph signals. Depending on the application, graph signals can be further considered as *smooth* with respect to functions of a graph Laplacian, defining graph-based (smoothing) filters for modeling processes such as diffusion. For example, in a social network, a localized signal (information) can diffuse on the network (i.e., on vertices of the graph) where smoothness of the signal increases as it spreads over time. In a wireless sensor network, sensor measurements (such as temperature) can be considered as smooth signals on a graph connecting communicating sensors, since sensors generally communicate with their close neighbors where the measurements are similar (i.e., spatially smooth). However, in practice, 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, learning, 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 [13, 14] by exploiting the available prior knowledge and assumptions about the desired graph (e.g., structural information including connectivity and sparsity level) and the observed data (e.g., signal smoothness).

The main focus of this thesis is on graph-based modeling, where the models of interest are defined based on graph Laplacian matrices (i.e., weighted graphs with nonnegative edge weights), so that the basic goal is to optimize a graph Laplacian matrix from data. For this purpose, various graph learning problems are formulated to estimate graph Laplacian matrices under different model assumptions on graphs and data. The graph learning problems studied in this thesis can be categorized as follows:

1. *Graph learning from data via structured graph Laplacian estimation*: An optimization framework is proposed to estimate different types of graph Laplacian matrices from a data statistic (e.g., covariance and symmetric kernel matrices), which is empirically obtained by using the samples in a given dataset. The problems of interest are formulated as finding the closest graph Laplacian fit to the inverse covariance of observed signal/data samples in a maximum a posteriori (MAP) sense. The additional prior constraints on the graph structure (e.g., graph connectivity and sparsity level) are also incorporated into the problems. The proposed framework constitutes a fundamental part of this work, since the optimization problems and applications considered throughout the thesis involve estimation of graph Laplacian matrices.

2. *Graph learning from filtered signals*: In this class of problems, the data is modeled based on a *graph system*, defined by a graph Laplacian and a graph-based filter (i.e., a matrix function of a graph Laplacian), where the observed set of signals are assumed to be outputs of a graph system with a specific type of graph-based filter. In order to learn graphs from empirical covariances of filtered signals, an optimization problem is formulated for joint identification of a graph Laplacian and a graph-based filter. The proposed problem is motivated by graph signal processing applications [3] and diffusion-based models [15] where the signals/data are intrinsically smooth with respect to an unknown graph.

3. *Graph learning from multiple graphs*: An optimization problem called *multigraph combining* is formulated to learn a single graph from a dataset consisting of multiple graph Laplacian matrices. This problem is motivated by signal processing and machine learning applications working with clusters of signals/data where each cluster can be modeled by a different graph, and an optimized ensemble model is desired to characterize the overall relations between the objects of interest. Especially when a signal/data model is uncertain (or unknown), combining

multiple candidate models would allow us to design a model that is robust to model uncertainties. Besides, multigraph combining can be used to summarize a dataset consisting multiple graphs into a single graph, which is favorable for graph visualization in data analytics.

In order to solve these classes of graph learning problems, specialized algorithms are developed. The proposed methods allow us to capture the statistics of signals/data by means of graphs (i.e., graph Laplacians), which are useful in a broad range of signal/data-oriented applications, discussed in Sections 1.2 and 1.3. Among different possible applications, this thesis primarily focuses on video coding, where the goal is to design graph-based transforms (GBTs) adapting to characteristics of video signals in order to improve the coding efficiency. Two distinct graph-based modeling techniques are developed to construct GBTs derived from graph Laplacian matrices:

1. Instances of a structured graph Laplacian estimation problem are solved to learn graphs based on empirical covariance matrices obtained from different classes of video block samples. The optimized graphs are used to derive GBTs that effectively capture statistical characteristics of video signals.

2. Graph Laplacian matrices are constructed on a per-block basis by first detecting image edges[a] (i.e., discontinuities) for each video block and then modifying weights of a predetermined graph based on the detected image edges. Thus, the resulting graph represents a class of block signals with a specific image edge structure, from which an edge-adaptive GBT is derived.

The main contributions of the thesis on graph-based modeling and video coding are summarized in Section 1.4.

1.1 Notation, Graphs and Graph Laplacian Matrices

In this section, we present the notation and basic definitions related to graphs and graph Laplacian matrices used throughout the thesis.

1.1.1 Notation

Throughout the thesis, lowercase normal (e.g., a and θ), lowercase bold (e.g., $\mathbf{a}$ and $\boldsymbol{\theta}$) and uppercase bold (e.g., $\mathbf{A}$ and $\boldsymbol{\Theta}$) letters denote scalars, vectors and matrices, respectively. Unless otherwise stated, calligraphic capital letters (e.g., $\mathcal{E}$ and $\mathcal{S}$) represent sets. $O(\cdot)$ and $\Omega(\cdot)$ are the standard big-O and big-Omega notations used in complexity theory [1]. The rest of the notation is presented in Table 1.1.

[a]We use the term *image edge* to distinguish edges in image/video signals with edges in graphs.

1.1.2 Graphs and Their Algebraic Representations

In this thesis, the models of interest are defined based on undirected, weighted graphs with nonnegative edge weights, which are formally defined as follows.

Definition 1 (Weighted Graph)**.** The graph $\mathcal{G}=(\mathcal{V},\mathcal{E},f_w,f_v)$ is a weighted graph with n vertices in the set $\mathcal{V}=\{v_1,\dots,v_n\}$. The edge set $\mathcal{E}=\{\,e\,|\,f_w(e)\neq 0,\,\forall\, e\in\mathcal{P}_u\}$ is a subset of $\mathcal{P}_u$, the set of all unordered pairs of distinct vertices, where $f_w((v_i,v_j))\geq 0$ for $i\neq j$ is a real-valued edge weight function, and $f_v(v_i)$ for $i=1,\dots,n$ is a real-valued vertex (self-loop) weight function.

Definition 2 (Simple Weighted Graph)**.** A simple weighted graph is a weighted graph with no self-loops (i.e., $f_v(v_i)=0$ for $i=1,\dots,n$).

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 $\mathcal{G}=(\mathcal{V},\mathcal{E},f_w,f_v)$ with n vertices, $v_1,\dots,v_n$:

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

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

Based on the defintions above, any weighted graph with positive edge weights 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., $\mathbf{V}=\mathbf{O}$). Moreover, if a weighted graph has nonnegative vertex weights (i.e., $\mathbf{V}\geq\mathbf{O}$), its corresponding generalized Laplacian matrix is also diagonally dominant. Furthermore, graph Laplacian matrices are symmetric and positive semidefinite. So, each of them has a complete set of orthogonal eigenvectors, $\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_n$, whose associated eigenvalues, $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$ are nonnegative real numbers. Specifically for combinatorial Laplacians of connected graphs, the first eigenvalue is zero ($\lambda_1 = 0$) and the associated eigenvector is $\mathbf{u}_1 = (1/\sqrt{n})\mathbf{1}$.

In this thesis, we consider three different types of graph Laplacian matrices, which lead to the following sets of graphs for a given vertex set $\mathcal{V}$.

- *Generalized Graph Laplacians (GGLs)*:

$$\mathcal{L}_g = \{\mathbf{L} \,|\, \mathbf{L} \succeq 0,\, (\mathbf{L})_{ij} \leq 0 \text{ for } i \neq j\}\,. \tag{1.1}$$

- *Diagonally Dominant Generalized Graph Laplacians (DDGLs)*:

$$\mathcal{L}_d = \{\mathbf{L} \,|\, \mathbf{L} \succeq 0,\, (\mathbf{L})_{ij} \leq 0 \text{ for } i \neq j,\, \mathbf{L1} \geq \mathbf{0}\}\,. \tag{1.2}$$

- *Combinatorial Graph Laplacians (CGLs)*:

$$\mathcal{L}_c = \{\mathbf{L} \,|\, \mathbf{L} \succeq 0,\, (\mathbf{L})_{ij} \leq 0 \text{ for } i \neq j,\, \mathbf{L1} = \mathbf{0}\}\,. \tag{1.3}$$

Moreover, the problems of interest include learning graphs with a specific choice of connectivity $\mathbf{A}$, that is, finding the best weights for the edges contained in $\mathbf{A}$. In order to incorporate the given connectivity information into the problems, we define the following set of all graph Laplacians depending on $\mathbf{A}$:

$$\mathcal{L}(\mathbf{A}) = \left\{ \mathbf{L} \in \mathcal{L} \;\middle|\; \begin{array}{ll} (\mathbf{L})_{ij} \leq 0 & \text{if } (\mathbf{A})_{ij} = 1 \\ (\mathbf{L})_{ij} = 0 & \text{if } (\mathbf{A})_{ij} = 0 \end{array} \;\text{for } i \neq j \right\}, \tag{1.4}$$

where $\mathcal{L}$ denotes a set of graph Laplacians (which can be $\mathcal{L}_g$, $\mathcal{L}_d$ or $\mathcal{L}_c$).

The sets described in (1.1)–(1.4) are used to specify Laplacian and connectivity constraints in our problem formulations.

1.2 Applications of Graph Laplacian Matrices

Graph Laplacian matrices have multiple applications in various fields. In spectral graph theory [16], 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 [15, 17], especially in clustering [18, 19, 20] and graph regularization [21] tasks. Moreover, in graph signal processing [3], basic signal processing operations such as filtering [9, 11], sampling [12], transformation [8, 10] 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) [16, 22], which represent graphs with zero vertex weights (i.e., graphs with no self-loops), there are recent studies where GGLs [23] (i.e., graphs with nonzero vertex weights) are shown to be useful. Particularly, GGLs are proposed for modeling image and video signals in [7, 8], and their potential machine learning applications are discussed in [24]. In [25], a Kron reduction procedure is developed based on GGLs for simplified modeling of electrical networks. Furthermore, DDGLs are utilized in [26, 27, 28] to develop efficient algorithms for graph partitioning [26], graph sparsification [27] and solving linear systems [28]. Our work discussed in the rest of the thesis complements these methods and applications by proposing efficient algorithms for estimation of graph Laplacians from data. The following section reviews some basic concepts from graph signal processing, which is a major area for our methods, because CGLs are extensively used.

1.3 Signal/Data Processing on Graphs: Graph-based Transforms and Filters

Traditional signal processing defines signals on regular Euclidean domains, where there is a fixed notion of frequency defined by the Fourier transform characterizing the smoothness of signals. Graph signal processing aims to extend basic signal processing operations on irregular non-Euclidean domains by defining signals on graphs, where the notion of frequency is graph-dependent. Specifically, the graph frequency spectrum is defined by eigenvalues of the graph Laplacian[a], $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$, which are called graph frequencies, and its eigenvectors $\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_n$ are the harmonics (basis vectors) associated with the graph frequencies. Based on the eigenvectors of a graph Laplacian matrix, the graph-based transform (GBT)[b] is formally defined as follows.

Definition 5 (Graph-based Transform or Graph Fourier Transform)**.** Let $\mathbf{L}$ be a graph Laplacian of a graph $\mathcal{G}$. The graph-based transform is the orthogonal matrix $\mathbf{U}$, satisfying $\mathbf{U}^\mathsf{T}\mathbf{U} = \mathbf{I}$, obtained by eigendecomposition of $\mathbf{L} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\mathsf{T}$, where $\mathbf{\Lambda}$ is the diagonal matrix consisting of eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_n$ (graph frequencies).

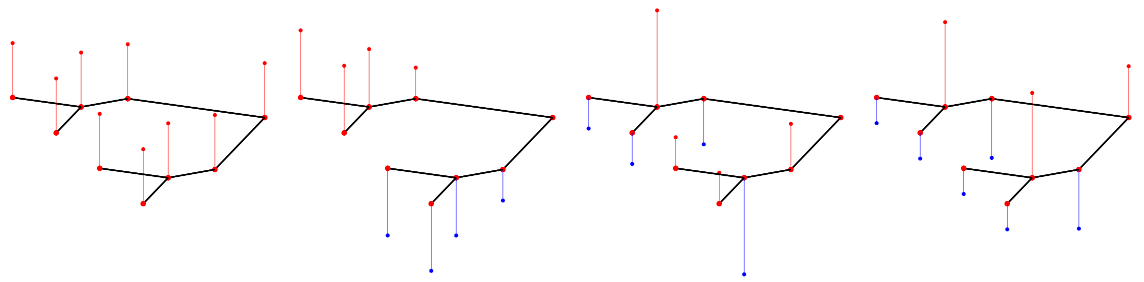
For a given graph signal $\mathbf{x} = [x_1\, x_2\, \cdots\, x_n]^\mathsf{T}$ defined on a graph $\mathcal{G}$ with n vertices, where x_i is attached to v_i (i-th vertex), the GBT of $\mathbf{x}$ is obtained by $\widehat{\mathbf{x}} = \mathbf{U}^\mathsf{T}\mathbf{x}$ where $\widehat{\mathbf{x}}$ denotes the GBT coefficients[c]. GBTs provide useful (Fourier-like) spectral representations for graph signals. As illustrated in Figure 1.1, the variation of GBT basis vectors gradually increase with the graph frequencies, and the basis vectors corresponding to low frequencies are relatively smooth.

To quantify smoothness of a graph signal $\mathbf{x}$, a common variation measure used in graph signal processing is the *graph Laplacian quadratic form*, $\mathbf{x}^\mathsf{T}\mathbf{L}\mathbf{x}$, which can be written in terms of edge and

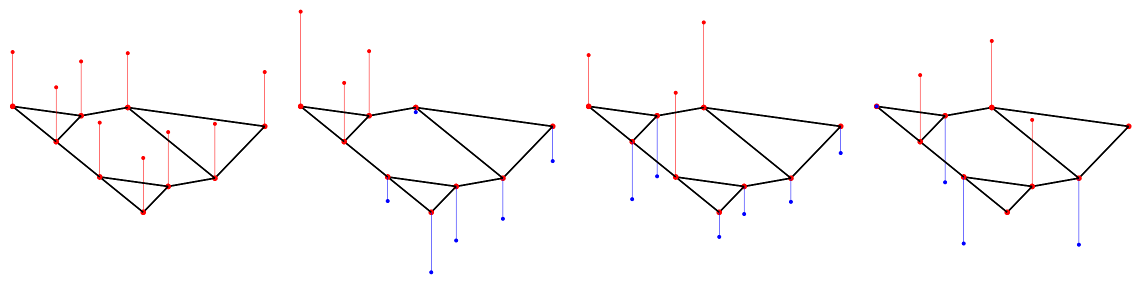
[a]In [29], adjacency matrices are used to define graph spectra. In this thesis, we adopt the graph Laplacian-based construction in [3].

[b]GBTs are also commonly referred as graph Fourier transforms (GFTs) [3].

[c]In Chapter 5, we design GBTs for video coding, where the video blocks are considered as graph signals and the resulting GBT coefficients are encoded.



(a) $\lambda_1 = 0$, $\mathbf{x} = \mathbf{u}_1$ (b) $\lambda_2 = 0.1392$, $\mathbf{x} = \mathbf{u}_2$ (c) $\lambda_8 = 4.1149$, $\mathbf{x} = \mathbf{u}_8$ (d) $\lambda_9 = 4.3028$, $\mathbf{x} = \mathbf{u}_9$



(e) $\lambda_1 = 0$, $\mathbf{x} = \mathbf{u}_1$ (f) $\lambda_2 = 0.6972$, $\mathbf{x} = \mathbf{u}_2$ (g) $\lambda_8 = 4.3028$, $\mathbf{x} = \mathbf{u}_8$ (h) $\lambda_9 = 5$, $\mathbf{x} = \mathbf{u}_9$

Figure 1.1: Illustration of GBTs derived from two different simple graphs with 9 vertices (i.e., 9×9 CGLs), where all edges are weighted as 1. The basis vectors associated with λ_1, λ_2, λ_8 and λ_9 are shown as graph signals, attached to vertices. Note that the notion of frequency (in terms of both GBTs and associated graph frequencies) changes depending on the underlying graph. For example, the additional edges in (f) lead to smaller signal differences (pairwise variations) at most of the vertices that are originally disconnected in (b).

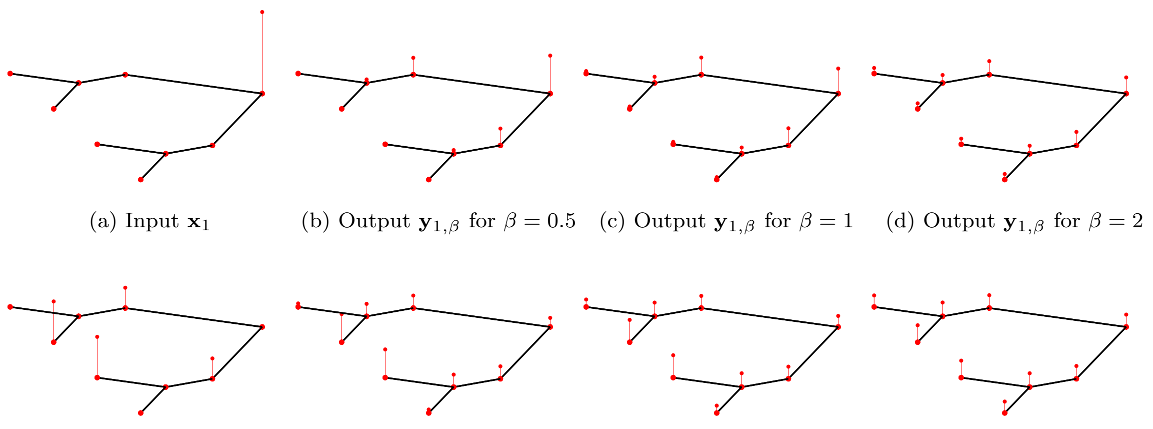
vertex weights of $\mathcal{G}$ as

$$\mathbf{x}^\mathsf{T}\mathbf{L}\mathbf{x} = \sum_{i=1}^{n} (\mathbf{V})_{ii}\, x_i^2 + \sum_{(i,j)\in\mathcal{I}} (\mathbf{W})_{ij}\, (x_i - x_j)^2 \tag{1.5}$$

where $(\mathbf{W})_{ij} = -(\mathbf{L})_{ij}$, $(\mathbf{V})_{ii} = \sum_{j=1}^{n} (\mathbf{L})_{ij}$ and $\mathcal{I} = \{(i,j)\,|\,(v_i, v_j) \in \mathcal{E}\}$ is the set of index pairs of vertices associated with the edge set $\mathcal{E}$. A smaller Laplacian quadratic term ($\mathbf{x}^\mathsf{T}\mathbf{L}\mathbf{x}$) indicates a smoother signal ($\mathbf{x}$), and for combinatorial Laplacians (simple weighted graphs), the smoothness depends only on edge weights ($\mathbf{W}$), since there are no self-loops (i.e., $\mathbf{V} = \mathbf{O}$). In graph frequency domain, the above measure can be written in terms of GBT coefficients, $\widehat{x}_i = \mathbf{u}_i^\mathsf{T}\mathbf{x}$ for $i = 1, \dots, n$, and graph frequencies $\lambda_1, \lambda_2, \dots, \lambda_n$ as

$$\mathbf{x}^\mathsf{T}\mathbf{L}\mathbf{x} = \mathbf{x}^\mathsf{T}\mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^\mathsf{T}\mathbf{x} = \sum_{i=1}^{n} (\mathbf{x}^\mathsf{T}\mathbf{u}_i)\lambda_i(\mathbf{u}_i{}^\mathsf{T}\mathbf{x}) = \sum_{i=1}^{n} \lambda_i \widehat{x}_i^2. \tag{1.6}$$

Moreover, the filtering operation for graph signals is defined in graph spectral (GBT) domain. We



(a) Input $\mathbf{x}_1$ (b) Output $\mathbf{y}_{1,\beta}$ for $\beta = 0.5$ (c) Output $\mathbf{y}_{1,\beta}$ for $\beta = 1$ (d) Output $\mathbf{y}_{1,\beta}$ for $\beta = 2$

(e) Input $\mathbf{x}_2$ (f) Output $\mathbf{y}_{2,\beta}$ for $\beta = 0.5$ (g) Output $\mathbf{y}_{2,\beta}$ for $\beta = 1$ (h) Output $\mathbf{y}_{2,\beta}$ for $\beta = 2$

Figure 1.2: Input-output relations of graph systems defined by $h(\mathbf{L}) = \exp(-\beta\mathbf{L})$ for different β, where $\mathbf{L}$ is the CGL associated with the shown graph whose all edges are weighted as 1. For given input graph signals (a) $\mathbf{x}_1$ and (e) $\mathbf{x}_2$, the corresponding output signals are (b–d) $\mathbf{y}_{1,\beta} = \exp(-\beta\mathbf{L})\mathbf{x}_1$ and (f–h) $\mathbf{y}_{2,\beta} = \exp(-\beta\mathbf{L})\mathbf{x}_2$ for $\beta = \{0.5, 1, 2\}$. Note that the output signals become more smooth as the parameter β increases.

formally define graph-based filters (GBFs)[a] as follows.

Definition 6 (Graph-based Filter)**.** Let $\mathbf{L}$ be a graph Laplacian of a graph $\mathcal{G}$. The graph-based filter is a matrix function h of graph Laplacian matrices, $h(\mathbf{L}) = \mathbf{U}h(\mathbf{\Lambda})\mathbf{U}^{\mathsf{T}}$, where $\mathbf{U}$ is the graph-based transform and $(h(\mathbf{\Lambda}))_{ii} = h((\mathbf{\Lambda})_{ii}) = h(\lambda_i)$ $\forall i$.

By definition, a graph-based filter h serves as a scalar function of λ (i.e., graph frequencies), so that we have

$$h(\mathbf{L}) = \mathbf{U}h(\mathbf{\Lambda}_\lambda)\mathbf{U}^{\mathsf{T}} = \sum_{i=1}^{n} h(\lambda_i)\mathbf{u}_i{\mathbf{u}_i}^{\mathsf{T}}. \tag{1.7}$$

Essentially, a GBF h maps graph frequencies $\lambda_1, \dots, \lambda_n$ to filter responses[b] $h(\lambda_1), \dots, h(\lambda_n)$. GBFs are used to define graph system as follows.

Definition 7 (Graph System)**.** A graph system is defined by a graph $\mathcal{G}$ and a graph-based filter h such that the input-output relation of the system is $\mathbf{y} = h(\mathbf{L})\mathbf{x}$, where $\mathbf{L}$ is a graph Laplacian associated with $\mathcal{G}$, and $\mathbf{x}$ is the input signal vector.

For a given input graph signal $\mathbf{x}$, the graph system output $\mathbf{y} = h(\mathbf{L})\mathbf{x}$ is obtained by modulating

[a] In Chapter 3, GBFs are used in modeling classes of filtered signals.

[b] Filter responses corresponding to the eigenvalues with multiplicity more than one have the same value. Formally, if $\lambda_i = \lambda_{i+1} = \cdots = \lambda_{i+c-1}$ for some $i > 1$ and multiplicity $c > 1$, then $h(\lambda_i) = h(\lambda_{i+1}) = \cdots = h(\lambda_{i+c-1})$.

the GBT coefficients of the input signal in $\widehat{\mathbf{x}} = \mathbf{U}^{\mathsf{T}}\mathbf{x}$ using $h(\mathbf{\Lambda}_\lambda)$ as

$$\mathbf{y} = h(\mathbf{L})\mathbf{x} = \mathbf{U}h(\mathbf{\Lambda}_\lambda)\mathbf{U}^{\mathsf{T}}\mathbf{x} = \mathbf{U}h(\mathbf{\Lambda}_\lambda)\widehat{\mathbf{x}} = \sum_{i=1}^{n} h(\lambda_i)\widehat{x}_i\mathbf{u}_i, \tag{1.8}$$

where $\widehat{x}_i = \mathbf{u}_i^{\mathsf{T}}\mathbf{x}$ for $i = 1, \dots, n$ as in (1.6). As an example, Figure 1.2 illustrates input-output relations of graph systems defined by $h(\mathbf{L}) = \exp(-\beta\mathbf{L})$ for three different β parameters. The matrix function $\exp(-\beta\mathbf{L})$ is called the exponential decay GBF, which is one of the GBFs used for modeling smooth graph signals in Chapter 3, where the parameter β determines the smoothness of the output signal.

1.4 Contributions of the Thesis

1.4.1 Graph-based Modeling

Graph Learning From Data: Structured Graph Laplacian Estimation

A general optimization framework is proposed in Chapter 2 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). Particularly, we consider three types of graph Laplacian matrices, namely, GGLs, DDGLs and CGLs. 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. Our experimental results demonstrate that the proposed algorithms outperform the current state-of-the-art methods [30, 31, 32, 33, 34] in terms of accuracy and computational efficiency.

Graph Learning From Filtered Signals: Graph System Identification

In Chapter 3, a novel graph signal processing framework is introduced for building graph-based models from classes of filtered signals, defined based on functions of a graph Laplacian matrix. In this framework, the graph-based modeling is formulated as a graph system identification problem where the goal is to learn a weighted graph (graph Laplacian) and a graph-based filter (function of a graph Laplacian). An algorithm is proposed to jointly identify a graph and an associated graph-based filter (GBF) from multiple signal/data observations under the assumption that GBFs are one-to-one functions. The proposed approach can also be applied to learn diffusion (heat) kernels [15], which are popular in various fields for modeling diffusion processes. In addition, for a specific choice of graph-based filters, the proposed problem reduces to a graph Laplacian estimation problem. Our experimental results demonstrate that the proposed approach outperforms the current state-of-the-art methods [31, 32, 34]. We also implement our framework on a real climate dataset for modeling of temperature signals.

Graph Learning From Multiple Graphs: Multigraph Combining

Chapter 4 of this thesis addresses the multigraph combining problem, which we define as designing an optimized graph from multiple graphs. Specifically, an optimization problem is formulated to find the best graph Laplacian that minimizes weighted sum of maximum likelihood (ML) criteria corresponding to given graph Laplacians. Based on the optimality conditions of the problem, an algorithm is proposed. Our experimental results show that the proposed solution provides better modeling compared to the commonly used averaging method.

1.4.2 Graph-based Transforms for Video Coding

In many state-of-the-art compression systems, signal transformation is an integral part of the encoding and decoding process, where transforms provide compact representations for the signals of interest. Chapter 5 of this thesis proposes GBTs for video compression, and develops two different techniques to design them. In the first technique, we solve specific instances of the GGL estimation problem by using the graph Laplacian estimation algorithms developed in Chapter 2, and the optimized graphs are used to design separable and nonseparable GBTs. The optimality of the proposed GBTs is also theoretically analyzed based on 1-D and 2-D Gaussian-Markov random field (GMRF) models for intra and inter predicted block signals. The second technique develops edge-adaptive GBTs (EA-GBTs) in order to flexibly adapt transforms to block signals with image edges (discontinuities) in order to improve coding efficiency. The advantages of EA-GBTs are both theoretically and empirically demonstrated. Our experimental results demonstrate that the proposed transforms can outperform the traditional Karhunen-Loeve transform (KLT).

Table 1.1: List of Symbols and Their Meaning

Symbols	**Meaning**
$\mathcal{G} \mid \mathbf{L}$	weighted graph | graph Laplacian matrix
h, $h_\beta \mid \beta$	graph-based filter | filter parameter β
$\lambda_i, \lambda_i(\mathbf{L})$	i-th eigenvalue of $\mathbf{L}$ in ascending order
$\mathcal{V} \mid \mathcal{E} \mid \mathcal{S}^c$	vertex set | edge set | complement of set $\mathcal{S}$
$\mathcal{P}_u$	set of unordered pairs of vertices
$\lvert\mathcal{S}\rvert$	cardinality of set $\mathcal{S}$
$n \mid k$	number of vertices | number of data samples
N	block size ($N \times N$) of an image/video patch
$\mathbf{O} \mid \mathbf{I}$	matrix of zeros | identity matrix
$\mathbf{0} \mid \mathbf{1}$	column vector of zeros | column vector of ones
$\mathbf{W} \mid \mathbf{A}$	adjacency matrix | connectivity matrix
$\mathbf{D} \mid \mathbf{V}$	degree matrix | self-loop matrix
$\mathbf{H} \mid \alpha$	regularization matrix | regularization parameter
$\mathbf{\Theta}^{-1} \mid \mathbf{\Theta}^{\dagger}$	inverse of $\mathbf{\Theta}$ | pseudo-inverse of $\mathbf{\Theta}$
$\mathbf{\Theta}^{\intercal} \mid \boldsymbol{\theta}^{\intercal}$	transpose of $\mathbf{\Theta}$ | transpose of $\boldsymbol{\theta}$
$\det(\mathbf{\Theta}) \mid \lvert\mathbf{\Theta}\rvert$	determinant of $\mathbf{\Theta}$ | pseudo-determinant of $\mathbf{\Theta}$
$(\mathbf{\Theta})_{ij}$	entry of $\mathbf{\Theta}$ at i-th row and j-th column
$(\mathbf{\Theta})_{i,:} \mid (\mathbf{\Theta})_{:,j}$	i-th row vector of $\mathbf{\Theta}$ | j-th column vector of $\mathbf{\Theta}$
$(\mathbf{\Theta})_{\mathcal{S}\mathcal{S}}$	submatrix of $\mathbf{\Theta}$ formed by selecting indexes in $\mathcal{S}$
$(\boldsymbol{\theta})_i$	i-th entry of $\boldsymbol{\theta}$
$(\boldsymbol{\theta})_{\mathcal{S}}$	subvector of $\boldsymbol{\theta}$ formed by selecting indexes in $\mathcal{S}$
$\geq (\leq)$	elementwise greater (less) than or equal to operator
$\mathbf{\Theta} \succeq 0$	$\mathbf{\Theta}$ is a positive semidefinite matrix
$\mathbf{\Theta} \succ 0$	$\mathbf{\Theta}$ is a positive definite matrix
$\mathrm{Tr} \mid \mathrm{logdet}(\mathbf{\Theta})$	trace operator | natural logarithm of $\det(\mathbf{\Theta})$
$\mathrm{diag}(\boldsymbol{\theta})$	diagonal matrix formed by elements of $\boldsymbol{\theta}$
$\mathrm{ddiag}(\mathbf{\Theta})$	diagonal matrix formed by diagonal elements of $\mathbf{\Theta}$
$\mathsf{p}(\mathbf{x})$	probability density function of random vector $\mathbf{x}$
$\mathbf{x} \sim \mathsf{N}(\mathbf{0}, \mathbf{\Sigma})$	zero-mean multivariate Gaussian with covariance $\mathbf{\Sigma}$
$x \sim \mathsf{U}(a,b)$	uniform distribution on the interval $[a,b]$
$\lVert\boldsymbol{\theta}\rVert_1$, $\lVert\mathbf{\Theta}\rVert_1$	sum of absolute values of all elements (ℓ_1-norm)
$\lVert\mathbf{\Theta}\rVert_{1,\text{off}}$	sum of absolute values of all off-diagonal elements
$\lVert\boldsymbol{\theta}\rVert_2^2$, $\lVert\mathbf{\Theta}\rVert_F^2$	sum of squared values of all elements
$\lVert\mathbf{\Theta}\rVert_{F,\text{off}}^2$	sum of squared values of all off-diagonal elements

Chapter 2

Graph Learning from Data: Structured Graph Laplacian Estimation

The focus of this chapter is on learning graphs (i.e., graph-based models) from data, where the basic goal is to find the nonnegative edge weights of a graph in order to characterize 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, for a given $k \times n$ data matrix $\mathbf{X}$ consisting of k observed data vectors with dimension n, the problems of interest are formulated as minimization of objective functions of the following form:

$$\underbrace{\operatorname{Tr}(\mathbf{\Theta}\mathbf{S}) - \operatorname{logdet}(\mathbf{\Theta})}_{\mathcal{D}(\mathbf{\Theta},\mathbf{S})} + \underbrace{\|\mathbf{\Theta} \odot \mathbf{H}\|_1}_{\mathcal{R}(\mathbf{\Theta},\mathbf{H})}, \tag{2.1}$$

where $\mathbf{\Theta}$ is the $n \times n$ target matrix variable and $\mathbf{S}$ denotes the data statistic obtained from $\mathbf{X}$. Depending on the application and underlying statistical assumptions, $\mathbf{S}$ may stand for the sample covariance of $\mathbf{X}$ or a kernel matrix $\mathbf{S} = \mathcal{K}(\mathbf{X}, \mathbf{X})$ derived from data, where $\mathcal{K}$ is a positive definite kernel function (e.g., polynomial and RBF kernels). $\mathcal{R}(\mathbf{\Theta}, \mathbf{H})$ is the sparsity promoting weighted ℓ_1-regularization term [40] multiplying $\mathbf{\Theta}$ and a selected regularization matrix $\mathbf{H}$ element-wise, and $\mathcal{D}(\mathbf{\Theta}, \mathbf{S})$ is the data-fidelity term, a log-determinant Bregman divergence [41], whose minimization corresponds to the maximum likelihood estimation of inverse covariance (precision) matrices for multivariate Gaussian distributions. Thus, minimizing (2.1) for arbitrary data can be interpreted as

Most of the work presented in this chapter is published in [35, 36, 37]. `MATLAB` [38] implementations of the proposed algorithms are available online [39].

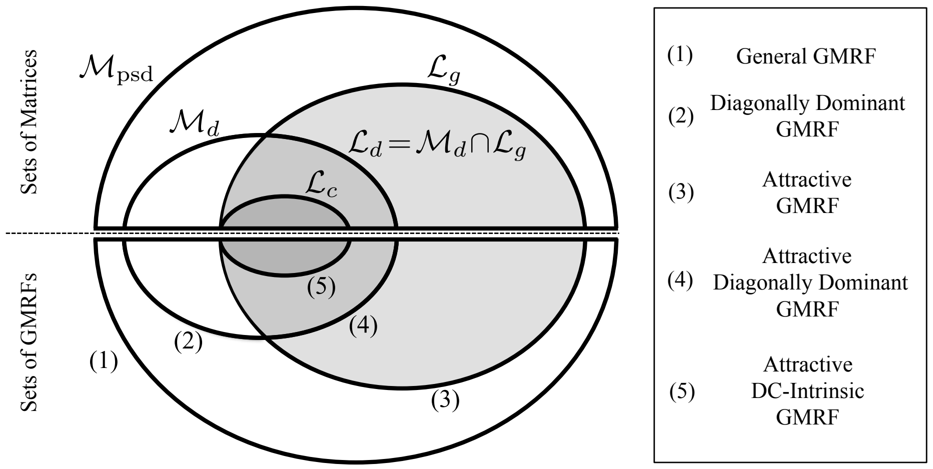

Sets of Matrices
Sets of GMRFs
$\mathcal{M}_{\text{psd}}$
$\mathcal{L}_g$
$\mathcal{M}_d$
$\mathcal{L}_d = \mathcal{M}_d \cap \mathcal{L}_g$
$\mathcal{L}_c$
(1)
(2)
(3)
(4)
(5)
(1) General GMRF
(2) Diagonally Dominant GMRF
(3) Attractive GMRF
(4) Attractive Diagonally Dominant GMRF
(5) Attractive DC-Intrinsic GMRF

Figure 2.1: The set of positive semidefinite matrices ($\mathcal{M}_{\text{psd}}$) containing the sets of diagonally dominant positive semidefinite matrices ($\mathcal{M}_d$), generalized ($\mathcal{L}_g$), diagonally dominant ($\mathcal{L}_d$) and combinatorial Laplacian ($\mathcal{L}_c$) matrices. The corresponding classes of GMRFs are enumerated as (1)–(5), respectively. In this work, we focus on estimating/learning the sets colored in gray.

finding the parameters of a multivariate Gaussian model that best approximates the data [42, 43]. In addition to the objective in (2.1), our formulations incorporate problem-specific Laplacian and structural constraints depending on (i) the desired type of graph Laplacian and (ii) the available information about the graph structure. Particularly, we consider three types of graph Laplacian matrices which are GGL, DDGL and CGL matrices (defined in Chapter 1) and develop novel techniques to estimate them from data (i.e., data statistic $\mathbf{S}$). As illustrated in Figure 2.1 and further discussed in Section 2.3, the proposed graph Laplacian estimation techniques can also be viewed as methods to learn different classes of Gaussian-Markov random fields (GMRFs) [44, 45], 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.

This chapter is organized as follows. In Section 2.1, we discuss the related studies and our contributions. Section 2.2 formulates our proposed problems and summarizes some of the related formulations in the literature. Section 2.3 discusses the probabilistic interpretation of our proposed problems. In Section 2.4, we derive necessary and sufficient optimality conditions and develop novel algorithms for the proposed graph learning problems. Experimental results are presented in Section 2.5, and some concluding remarks are discussed in Section 2.6.

2.1 Related Work and Contributions

2.1.1 Sparse Inverse Covariance Estimation

In the literature, several approaches have been proposed for estimating graph-based models. Dempster [46] 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 [47] by using the Lasso algorithm [48]. Friedman *et al.* [30] 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 [42, 49], and a few computationally efficient variations are presented in [50, 51, 52]. However, inverse covariance estimation methods, such as the Graphical Lasso, search for solutions in the set of the positive semidefinite matrices ($\mathcal{M}_{\text{psd}}$ in Figure 2.1), 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 Figure 2.1). 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 (2.1) for applications where graph Laplacians are useful (see Section 1.2).

2.1.2 Graph Laplacian Estimation

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 [53], or equivalently GGLs, and propose an efficient primal algorithm [54], while our recent work [55] proposes an alternative dual algorithm for GGL estimation. Our work addresses the same GGL estimation problem as [54, 55], based on a primal approach analogous to that of [54], but unlike both [54] and [55], we incorporate connectivity constraints in addition to sparsity promoting regularization. For estimation of CGLs, Lake and Tenenbaum [33] also consider minimization of the objective function in (2.1), 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.* [31] and Kalofolias [32] propose minimization of two objective functions different from (2.1) 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 (2.1) 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.

2.1.3 Graph Topology Inference

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.* [34] and Pasdeloup *et al.* [56] focus on learning graph shift/diffusion operators (such as adjacency and Laplacian matrices) from a set of diffused graph signals, and Sardellitti *et al.* [57] propose an approach to estimate a graph Laplacian from bandlimited graph signals. None of these works [34, 56, 57] considers the minimization of (2.1). 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 (2.1). The problem of learning diffusion-based models is addressed in Chapter 3.

2.1.4 Summary of Contributions

In this work, we address estimation of three different types of graph Laplacian with structural constraints. For CGL estimation, we propose a novel formulation for the objective function in (2.1), 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 [33, 31, 32]. For GGL estimation, the prior formulations in [54, 55] are extended in order to accommodate structural constraints. To solve the proposed problems, we develop efficient block-coordinate descent (BCD) algorithms [58] 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 [42, 30, 49] which iteratively solve an ℓ_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 Figure 2.1. While recent work has noted the relation between graph Laplacians and GMRFs [6, 59], this chapter provides a more comprehensive classification of GMRFs and proposes specific methods for estimation of their parameters.

2.2 Problem Formulations for Graph Learning

2.2.1 Proposed Formulations: Graph Laplacian Estimation

For the purpose of graph learning, we formulate three different optimization problems for a given $\mathbf{S}$, a connectivity matrix $\mathbf{A}$ and a regularization matrix $\mathbf{H}$. In our problems, we minimize the function in (2.1) under the set constraint $\mathbf{\Theta} \in \mathcal{L}(\mathbf{A})$ defined in (1.4), where the choices of $\mathcal{L}$ and $\mathbf{A}$ determine the Laplacian and connectivity constraints, respectively.

Based on the Laplacian constraints on $\mathbf{\Theta}$ (i.e., nonnegativity of edge weights), a regularization matrix $\mathbf{H}$ can be selected such that $\mathcal{R}(\mathbf{\Theta}, \mathbf{H})$ term in (2.1) is compactly written as

$$\|\mathbf{\Theta} \odot \mathbf{H}\|_1 = \operatorname{Tr}(\mathbf{\Theta}\mathbf{H}). \tag{2.2}$$

For example, the following standard ℓ_1-regularization terms with parameter α can be written in the above form as,

$$\alpha\|\mathbf{\Theta}\|_1 = \operatorname{Tr}(\mathbf{\Theta}\mathbf{H}) \text{ where } \mathbf{H} = \alpha(2\mathbf{I} - \mathbf{1}\mathbf{1}^{\mathsf{T}}), \tag{2.3}$$

and

$$\alpha\|\mathbf{\Theta}\|_{1,\text{off}} = \operatorname{Tr}(\mathbf{\Theta}\mathbf{H}) \text{ where } \mathbf{H} = \alpha(\mathbf{I} - \mathbf{1}\mathbf{1}^{\mathsf{T}}). \tag{2.4}$$

Note that $\alpha\|\mathbf{\Theta}\|_{1,\text{off}}$ applies ℓ_1-regularization to off-diagonal entries of $\mathbf{\Theta}$ only (see in Table 1.1). Since the trace operator is linear, we can rewrite the objective function in (2.1) as

$$\operatorname{Tr}(\mathbf{\Theta}\mathbf{K}) - \operatorname{logdet}(\mathbf{\Theta}) \text{ where } \mathbf{K}{=}\mathbf{S}{+}\mathbf{H}, \tag{2.5}$$

which is the form used in our optimization problems. Note that the nonnegativity of edge weights allows us to transform the nonsmooth function in (2.1) into the smooth function in (2.5) by rewriting the regularization term as in (2.2).

In what follows, we formally introduce three different optimization problems with Laplacian and structural constraints.

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

$$\begin{aligned} \underset{\mathbf{\Theta}}{\text{minimize}} \quad & \operatorname{Tr}(\mathbf{\Theta}\mathbf{K}) - \operatorname{logdet}(\mathbf{\Theta}) \\ \text{subject to} \quad & \mathbf{\Theta} \in \mathcal{L}_g(\mathbf{A}) \end{aligned} \tag{2.6}$$

where $\mathbf{K}{=}\mathbf{S}{+}\mathbf{H}$ as in (2.5), and the set of constraints $\mathcal{L}_g(\mathbf{A})$ leads to $\mathbf{\Theta}$ being a GGL matrix.

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

$$\begin{aligned} \underset{\boldsymbol{\Theta}}{\text{minimize}} \quad & \operatorname{Tr}(\boldsymbol{\Theta}\mathbf{K}) - \operatorname{logdet}(\boldsymbol{\Theta}) \\ \text{subject to} \quad & \boldsymbol{\Theta} \in \mathcal{L}_d(\mathbf{A}) \end{aligned} \tag{2.7}$$

where the additional $\boldsymbol{\Theta}\mathbf{1} \geq \mathbf{0}$ constraint in $\mathcal{L}_d(\mathbf{A})$ ensures that all vertex weights are nonnegative, and therefore the optimal solution is a diagonally dominant matrix.

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

$$\begin{aligned} \underset{\boldsymbol{\Theta}}{\text{minimize}} \quad & \operatorname{Tr}(\boldsymbol{\Theta}\mathbf{K}) - \log|\boldsymbol{\Theta}| \\ \text{subject to} \quad & \boldsymbol{\Theta} \in \mathcal{L}_c(\mathbf{A}) \end{aligned} \tag{2.8}$$

where the objective function involves the pseudo-determinant term ($|\boldsymbol{\Theta}|$), since the target matrix $\boldsymbol{\Theta}$ is singular. However, the problem is hard to solve because of the $|\boldsymbol{\Theta}|$ term. To cope with this, we propose to reformulate (2.8) as the following problem[a],

$$\begin{aligned} \underset{\boldsymbol{\Theta}}{\text{minimize}} \quad & \operatorname{Tr}(\boldsymbol{\Theta}(\mathbf{K}+\mathbf{J})) - \operatorname{logdet}(\boldsymbol{\Theta}+\mathbf{J}) \\ \text{subject to} \quad & \boldsymbol{\Theta} \in \mathcal{L}_c(\mathbf{A}) \end{aligned} \tag{2.9}$$

where the $\boldsymbol{\Theta}\mathbf{1}=\mathbf{0}$ constraint in $\mathcal{L}_c(\mathbf{A})$ guarantees that the solution is a CGL matrix, and $\mathbf{J} = \mathbf{u}_1\mathbf{u}_1^\intercal$ such that $\mathbf{u}_1 = (1/\sqrt{n})\mathbf{1}$ is the eigenvector corresponding to the zero eigenvalue of CGL matrices.

Proposition 1. *The optimization problems stated in (2.8) and (2.9) are equivalent.*

Proof. The problems in (2.8) and (2.9) have the same constraints. To prove their equivalence, we show that their objective functions are also the same. First, note that

$$\operatorname{Tr}(\boldsymbol{\Theta}(\mathbf{K}+\mathbf{J})) = \operatorname{Tr}(\boldsymbol{\Theta}\mathbf{K}) + \frac{1}{n}\operatorname{Tr}(\boldsymbol{\Theta}\mathbf{1}\mathbf{1}^\intercal) = \operatorname{Tr}(\boldsymbol{\Theta}\mathbf{K})$$

since $\boldsymbol{\Theta}\mathbf{1}=\mathbf{0}$ based on the CGL problem constraints. Next, we can write

$$\operatorname{logdet}(\boldsymbol{\Theta} + 1/n\,\mathbf{1}\mathbf{1}^\intercal) = \log\left(\prod_{i=1}^{n} \lambda_i(\boldsymbol{\Theta} + 1/n\,\mathbf{1}\mathbf{1}^\intercal)\right) \tag{2.10}$$

where $\lambda_i(\boldsymbol{\Theta})$ denotes the i-th eigenvalue of $\boldsymbol{\Theta}$ in ascending order ($\lambda_1(\boldsymbol{\Theta}) \leq \cdots \leq \lambda_n(\boldsymbol{\Theta})$). Since the eigenvector corresponding to the first (zero) eigenvalue (i.e., $\lambda_1(\boldsymbol{\Theta}) = 0$) is $\mathbf{u}_1 = 1/\sqrt{n}\;\mathbf{1}$, by the

[a] An alternative second-order approach is proposed to solve (2.9) in [60], which is published after the initial version of this work [36]. Yet, the equivalence of (2.8) and (2.9) is not discussed in [60].

problem constraints (i.e., by definition of CGL matrices), we have that

$$\mathbf{\Theta} + \frac{1}{n}\mathbf{1}\mathbf{1}^{\mathsf{T}} = (\underbrace{\lambda_1(\mathbf{\Theta})}_{0}+1)\mathbf{u}_1\mathbf{u}_1^{\mathsf{T}} + \sum_{i=2}^{n}\lambda_i(\mathbf{\Theta})\mathbf{u}_i\mathbf{u}_i^{\mathsf{T}}. \tag{2.11}$$

Since the determinant of a matrix is equal to the product of its eigenvalues, from (2.11) we have

$$\operatorname{logdet}(\mathbf{\Theta} + 1/n\,\mathbf{1}\mathbf{1}^{\mathsf{T}}) = \log\left(1\cdot\prod_{i=2}^{n}\lambda_i(\mathbf{\Theta})\right) = \log|\mathbf{\Theta}|.$$

Therefore, the problems in (2.8) and (2.9) are equivalent. □

Proposition 2. *Problems 1, 2 and 3 are convex optimization problems.*

Proof. The function $\operatorname{logdet}(\mathbf{\Theta})$ defined over positive semidefinite matrices ($\mathbf{\Theta} \succeq 0$) is a concave function (see [61] for a proof), and $\operatorname{Tr}(\cdot)$ 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. □

In Problems 1, 2 and 3, prior knowledge/assumptions about the graph structure are built into the choice of $\mathbf{A}$, determining the structural constraints. In practice, if the graph connectivity is unknown, then $\mathbf{A}$ can be set to represent a fully connected graph, $\mathbf{A}=\mathbf{A}_{\text{full}}=\mathbf{1}\mathbf{1}^{\mathsf{T}}-\mathbf{I}$, and the regularization matrix $\mathbf{H}$ (or the parameter α for the ℓ_1-regularizations in (2.3) and (2.4)) can be tuned until the desired level of sparsity is achieved.

2.2.2 Related Prior Formulations

In this section, we review some of the related problems previously proposed in the literature.

Sparse Inverse Covariance Estimation [30]. The goal is to estimate a sparse inverse covariance matrix from $\mathbf{S}$ by solving:

$$\underset{\mathbf{\Theta}\succeq 0}{\text{minimize}}\ \operatorname{Tr}(\mathbf{\Theta}\mathbf{S}) - \operatorname{logdet}(\mathbf{\Theta}) + \alpha\|\mathbf{\Theta}\|_1. \tag{2.12}$$

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

Shifted CGL Estimation [33]. 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:

$$\begin{aligned} \underset{\mathbf{\Theta}\succeq 0,\,\nu\geq 0}{\text{minimize}}\quad & \operatorname{Tr}(\mathbf{\Theta}\mathbf{S}) - \operatorname{logdet}(\mathbf{\Theta}) + \alpha\|\mathbf{\Theta}\|_1 \\ \text{subject to}\quad & \mathbf{\Theta} = \widetilde{\mathbf{\Theta}} + \nu\mathbf{I} \\ & \widetilde{\mathbf{\Theta}}\mathbf{1} = \mathbf{0},\ (\widetilde{\mathbf{\Theta}})_{ij} \leq 0 \quad i \neq j \end{aligned} \tag{2.13}$$

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

Proposition 3. *The objective functions of Problem 3 and the shifted CGL problem in (2.13) are different.*

Proof. The optimal ν 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 (2.13) can be decomposed as follows

$$\mathcal{J}(\widetilde{\mathbf{\Theta}}) + \nu \mathrm{Tr}(\mathbf{S}) - \sum_{i=2}^{n} \log\left(1 + \frac{\nu}{\lambda_i(\widetilde{\mathbf{\Theta}})}\right) - \log(\nu) \tag{2.14}$$

where $\mathcal{J}(\widetilde{\mathbf{\Theta}})$ is the objective of Problem 3 with $\mathbf{H} = \mathbf{O}$. □

Graph Learning from Smooth Signals [31, 32]**.** The goal is to estimate a CGL from n-dimensional signals that are assumed to be smooth with respect to the corresponding graph:

$$\begin{aligned} &\underset{\mathbf{\Theta}\succeq 0}{\text{minimize}} \;\; \mathrm{Tr}\left(\mathbf{\Theta}\mathbf{S}\right) + \alpha_1 \|\mathbf{\Theta}\|_F^2 \\ &\text{subject to} \;\; \mathbf{\Theta}\mathbf{1} = \mathbf{0}, \; \mathrm{Tr}\left(\mathbf{\Theta}\right) = n, \; (\mathbf{\Theta})_{ij} \leq 0 \quad i \neq j \end{aligned} \tag{2.15}$$

where the sum of degrees is constrained as $\mathrm{Tr}\left(\mathbf{\Theta}\right) = n$. This is a limitation that is later relaxed in [32] by introducing the following problem with regularization parameters

$$\begin{aligned} &\underset{\mathbf{\Theta}\succeq 0}{\text{minimize}} \;\; \mathrm{Tr}\left(\mathbf{\Theta}\mathbf{S}\right) + \alpha_1 \|\mathbf{\Theta}\|_{F,\text{off}}^2 - \alpha_2 \sum_{i=1}^{n} \log\left((\mathbf{\Theta})_{ii}\right) \\ &\text{subject to} \;\; \mathbf{\Theta}\mathbf{1} = \mathbf{0}, \quad (\mathbf{\Theta})_{ij} \leq 0 \quad i \neq j \end{aligned} \tag{2.16}$$

where the constraints in (2.15) and (2.16) lead to a CGL solution. The following proposition relates the objective function in (2.16) with $\alpha_1 = 0$ to the objective in our proposed CGL estimation problem.

Proposition 4. *The objective function in Problem 3 with $\alpha = 0$ is lower-bounded by the objective function in (2.16) for $\alpha_1 = 0$ and $\alpha_2 = 1$.*

Proof. For $\alpha_1 = 0$ and $\alpha_2 = 1$, the objective function in (2.16) is written as $\mathrm{Tr}(\mathbf{\Theta}\mathbf{S}) - \sum_{i=1}^{n} \log((\mathbf{\Theta})_{ii})$. By using Hadamard's inequality $\det(\mathbf{\Theta}) \leq \prod_{i=1}^{n} (\mathbf{\Theta})_{ii}$ [62] and taking the log of both sides, the following bound is obtained

$$\mathrm{Tr}(\mathbf{\Theta}\mathbf{S}) - \sum_{i=1}^{n} \log((\mathbf{\Theta})_{ii}) \leq \mathrm{Tr}(\mathbf{\Theta}\mathbf{S}) - \operatorname{logdet}(\mathbf{\Theta}) \tag{2.17}$$

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

Graph Topology Inference. Various approaches for graph topology (connectivity) inference from data (under diffusion-based model assumptions) have been proposed in [34, 56, 57]. As the most related to our work, Segarra *et al.* [34] introduce a sparse recovery problem to infer the graph topology information from the eigenbasis, $\mathbf{U}$, associated with a graph shift/diffusion operator. Specifically for CGL estimation, the following problem is formulated:

$$\begin{aligned} \underset{\boldsymbol{\Theta}\succeq 0,\boldsymbol{\Lambda}}{\text{minimize}} \quad & \|\boldsymbol{\Theta}\|_1 \\ \text{subject to} \quad & \boldsymbol{\Theta} = \mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^{\mathsf{T}} \\ & \boldsymbol{\Theta}\mathbf{1} = \mathbf{0},\ (\boldsymbol{\Theta})_{ij} \leq 0 \quad i \neq j \end{aligned} \tag{2.18}$$

where the eigenbasis $\mathbf{U}$ is the input to the problem, so that the goal is to find the set of eigenvalues (i.e., the diagonal matrix $\boldsymbol{\Lambda}$) minimizing $\|\boldsymbol{\Theta}\|_1$. Note that the problems in [34, 56, 57] require that $\mathbf{U}$ be given (or calculated beforehand), while our goal is to directly estimate a graph Laplacian so that both $\mathbf{U}$ and $\boldsymbol{\Lambda}$ are jointly optimized.

The estimation of Laplacians from data under diffusion-based assumptions be discussed later in Chapter 3.

2.3 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[a] $\mathbf{x} \sim \mathsf{N}(\mathbf{0}, \boldsymbol{\Sigma} = \boldsymbol{\Omega}^{\dagger})$, parametrized with a positive semidefinite precision matrix $\boldsymbol{\Omega}$, defining a Gaussian Markov random field (GMRF)

$$\mathsf{p}(\mathbf{x}|\boldsymbol{\Omega}) = \frac{1}{(2\pi)^{n/2}|\boldsymbol{\Omega}^{\dagger}|^{1/2}} \exp\left(-\frac{1}{2}\mathbf{x}^{\mathsf{T}}\boldsymbol{\Omega}\mathbf{x}\right), \tag{2.19}$$

with covariance matrix $\boldsymbol{\Sigma} = \boldsymbol{\Omega}^{\dagger}$. Based on its precision matrix ($\boldsymbol{\Omega}$), a GMRF is classified as [44, 45]:

- a *general GMRF* if its precision $\boldsymbol{\Omega}$ is positive semidefinite,
- an *attractive GMRF* if its precision $\boldsymbol{\Omega}$ has nonpositive off-diagonal entries,
- a *diagonally dominant GMRF* if its precision $\boldsymbol{\Omega}$ is diagonally dominant,
- an *intrinsic GMRF* if its precision $\boldsymbol{\Omega}$ is positive semidefinite and singular.

[a]The zero-mean assumption is made to simplify the notation. Our analysis can be trivially extended to a multivariate Gaussian with nonzero mean.

The entries of the precision matrix $\mathbf{\Omega}$ can be interpreted in terms of the following conditional dependence relations among the variables in $\mathbf{x}$,

$$\mathsf{E}\left[x_i \,|\, (\mathbf{x})_{\mathcal{S}\setminus\{i\}}\right] = -\frac{1}{(\mathbf{\Omega})_{ii}} \sum_{j\in\mathcal{S}\setminus\{i\}} (\mathbf{\Omega})_{ij} x_j \tag{2.20}$$

$$\mathsf{Prec}\left[x_i \,|\, (\mathbf{x})_{\mathcal{S}\setminus\{i\}}\right] = (\mathbf{\Omega})_{ii} \tag{2.21}$$

$$\mathsf{Corr}\left[x_i x_j \,|\, (\mathbf{x})_{\mathcal{S}\setminus\{i,j\}}\right] = -\frac{(\mathbf{\Omega})_{ij}}{\sqrt{(\mathbf{\Omega})_{ii}(\mathbf{\Omega})_{jj}}} \quad i \neq j, \tag{2.22}$$

where $\mathcal{S}=\{1,\dots,n\}$ is the index set for $\mathbf{x} = [x_1\, x_2\, \cdots\, x_n]^\mathsf{T}$. The conditional expectation in (2.20) represents the minimum mean square error (MMSE) prediction of x_i using all the other random variables in $\mathbf{x}$. The *precision* of x_i is defined as in (2.21), and the relation in (2.22) corresponds to the *partial correlation* between x_i and x_j (i.e., correlation between random variables x_i and x_j given all the other variables in $\mathbf{x}$). For example, if x_i and x_j are conditionally independent ($(\mathbf{\Omega})_{ij}=0$), there is no edge between corresponding vertices v_i and v_j. For GMRFs, whose precision matrices are graph Laplacian matrices (i.e., $\mathbf{\Omega}=\mathbf{L}$), 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 Figure 2.1:

- $\mathbf{L}$ is a GGL matrix ($\mathbf{L}\in\mathcal{L}_g$) if and only if $\mathsf{p}(\mathbf{x}|\mathbf{L})$ is an attractive GMRF,
- $\mathbf{L}$ is a DDGL matrix ($\mathbf{L} \in \mathcal{L}_d$) if and only if $\mathsf{p}(\mathbf{x}|\mathbf{L})$ is an attractive, diagonally dominant GMRF,
- $\mathbf{L}$ is a CGL matrix ($\mathbf{L}\in\mathcal{L}_c$) if and only if $\mathsf{p}(\mathbf{x}|\mathbf{L})$ 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 [44] with no probability density along the direction of the eigenvector $\mathbf{u}_1=1/\sqrt{n}\,\mathbf{1}$ associated with the zero eigenvalue ($\lambda_1(\mathbf{L})=0$). On the other hand, if $\mathbf{L}$ is a nonsingular GGL matrix, then $\mathbf{x}$ has a proper (non-degenerate) distribution.

Moreover, for $\mathbf{\Omega} = \mathbf{L}$, the $\mathbf{x}^\mathsf{T}\mathbf{\Omega}\mathbf{x}$ term in (2.19) becomes the graph Laplacian quadratic form stated in (1.5), which is used to quantify smoothness of graph signals [3]. In our formulations, the Laplacian quadratic from $\mathbf{x}^\mathsf{T}\mathbf{L}\mathbf{x}$ 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, $\mathbf{x}_i$ for $i = 1,\dots,k$, are obtained from an attractive GMRF with unknown parameters, the likelihood of a candidate graph Laplacian $\mathbf{L}$ can be written as

$$\prod_{i=1}^{k} \mathsf{p}(\mathbf{x}_i|\mathbf{L}) = (2\pi)^{-\frac{kn}{2}} |\mathbf{L}^{\dagger}|^{-\frac{k}{2}} \prod_{i=1}^{k} \exp\left(-\frac{1}{2}\mathbf{x}_i{}^\mathsf{T}\mathbf{L}\mathbf{x}_i\right). \tag{2.23}$$

Let $\mathbf{L}(\mathbf{w},\mathbf{v})$ be defined by edge weight and vertex weight vectors $\mathbf{w}=[f_w(e_1),\dots,f_w(e_m)]^\mathsf{T}$ and $\mathbf{v}=[f_v(v_1),\dots,f_v(v_n)]^\mathsf{T}$, 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 (2.23) can be equivalently formulated as minimizing the negative log-likelihood, that is

$$\begin{aligned}\widehat{\mathbf{L}}_{\mathrm{ML}} &= \underset{\mathbf{L}(\mathbf{w},\mathbf{v})}{\operatorname{argmin}} \left\{ -\frac{k}{2}\log|\mathbf{L}| + \frac{1}{2}\sum_{i=1}^{k} \operatorname{Tr}\left(\mathbf{x}_i{}^\mathsf{T}\mathbf{L}\mathbf{x}_i\right)\right\} \\ &= \underset{\mathbf{L}(\mathbf{w},\mathbf{v})}{\operatorname{argmin}} \left\{\operatorname{Tr}(\mathbf{L}\mathbf{S}) - \log|\mathbf{L}|\right\}\end{aligned} \tag{2.24}$$

where $\mathbf{S}$ is the sample covariance matrix, and $\widehat{\mathbf{L}}_{\mathrm{ML}}$ denotes the maximum likelihood estimate of $\mathbf{L}(\mathbf{w},\mathbf{v})$. Moreover, we can derive maximum a posteriori (MAP) estimation problems by incorporating the information known about $\mathbf{L}$ into a prior distribution $\mathsf{p}(\mathbf{L})$ as

$$\widehat{\mathbf{L}}_{\mathrm{MAP}} = \underset{\mathbf{L}(\mathbf{w},\mathbf{v})}{\operatorname{argmin}} \left\{\operatorname{Tr}(\mathbf{L}\mathbf{S}) - \log|\mathbf{L}| - \log(\mathsf{p}(\mathbf{L}))\right\}. \tag{2.25}$$

For example, we can choose the following m-variate exponential prior for sparse estimation of $\mathbf{w}$,

$$\mathsf{p}(\mathbf{w}) = (2\alpha)^m \exp\left(-2\alpha\mathbf{1}^\mathsf{T}\mathbf{w}\right) \quad \text{for } \mathbf{w}\geq\mathbf{0}, \tag{2.26}$$

so that the MAP estimation in (2.25) can be written as follows:

$$\begin{aligned}\widehat{\mathbf{L}}_{\mathrm{MAP}} &= \underset{\mathbf{L}(\mathbf{w},\mathbf{v})}{\operatorname{argmin}} \left\{\operatorname{Tr}(\mathbf{L}\mathbf{S}) - \log|\mathbf{L}| - \log(\mathsf{p}(\mathbf{w}))\right\} \\ &= \underset{\mathbf{L}(\mathbf{w},\mathbf{v})}{\operatorname{argmin}} \left\{\operatorname{Tr}(\mathbf{L}\mathbf{S}) - \log|\mathbf{L}| + 2\alpha\|\mathbf{w}\|_1\right\} \\ &= \underset{\mathbf{L}(\mathbf{w},\mathbf{v})}{\operatorname{argmin}} \left\{\operatorname{Tr}(\mathbf{L}\mathbf{S}) - \log|\mathbf{L}| + \alpha\|\mathbf{L}\|_{1,\mathrm{off}}\right\}\end{aligned} \tag{2.27}$$

where the resulting minimization is equivalent to the objective of our problems with the regularization in (2.4).

Proposition 5. *Let the data model be $\mathbf{x}\sim\mathsf{N}(\mathbf{0},\mathbf{L}^\dagger)$ as in (2.19). Then, Problems 1, 2 and 3 are specific instances of the maximum a posteriori estimation problem in (2.25).*

Proof. With proper choices of the prior $\mathsf{p}(\mathbf{L})$ in (2.25), the objective function in (2.1) can be constructed for any $\mathbf{H}$, except the pseudo-determinant term $|\mathbf{\Theta}|$ needed for estimating CGL matrices. For this case, Proposition 1 shows that we can equivalently formulate (2.25) in the form of Problem 3. The construction in (2.25)–(2.27) can be trivially extended for the weighted ℓ_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 [63] on $\mathbf{v}$ and $\mathbf{w}$, so that spike priors correspond to zero edge weights, and slab priors allow nonnegative edge weights. □

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 $\mathbf{S}$, which is obtained from the data, the solution of (2.25) corresponds to the closest Laplacian in terms of a regularized log-determinant divergence criterion [41]. 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 $\mathbf{S}$.

2.4 Proposed Graph Learning Algorithms

Problems 1, 2 and 3 can be solved using general purpose solvers such as CVX [64]. 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 (2.34) in (2.38)). In order to develop efficient methods, we propose iterative block-coordinate descent algorithms [58], 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 ($\mathbf{\Theta}$), 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 $\mathbf{\Theta}$ and its inverse ($\mathbf{C}$), 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 $\mathbf{\Theta}$ and $\mathbf{C}$ in our algorithms (Section 2.4.1). Then, Algorithms 1 and 2 are presented with the derivations of subproblems based on the optimality conditions of corresponding graph learning problems. Specifically, Algorithm 1 is proposed to solve Problems 1 and 2 (Section 2.4.2), while Algorithm 2 solves Problem 3 (Section 2.4.3). Finally, the convergence and computational complexity of proposed algorithms are analyzed (Section 2.4.4).

2.4.1 Matrix Update Formulas

The proposed algorithms exploit the following formulas to update the target variable $\mathbf{\Theta}$ and its inverse $\mathbf{C}$ iteratively.

Row/column updates. Updating the u-th row/column of $\mathbf{\Theta}$ results in updating all the elements in its inverse $\mathbf{C}=\mathbf{\Theta}^{-1}$, which can be obtained by the matrix inversion lemma [65],

$$\begin{aligned}(\mathbf{P}^{\mathsf{T}}\mathbf{\Theta}\mathbf{P})^{-1} &= \begin{bmatrix}\mathbf{\Theta}_u & \boldsymbol{\theta}_u \\ \boldsymbol{\theta}_u^{\mathsf{T}} & \theta_u\end{bmatrix}^{-1} = \mathbf{P}^{\mathsf{T}}\mathbf{C}\mathbf{P} = \begin{bmatrix}\mathbf{C}_u & \mathbf{c}_u \\ \mathbf{c}_u^{\mathsf{T}} & c_u\end{bmatrix} \\ &= \begin{bmatrix}\left(\mathbf{\Theta}_u - \dfrac{\boldsymbol{\theta}_u\boldsymbol{\theta}_u^{\mathsf{T}}}{\theta_u}\right)^{-1} & -\mathbf{C}_u\dfrac{\boldsymbol{\theta}_u}{\theta_u} \\ -\dfrac{\boldsymbol{\theta}_u^{\mathsf{T}}}{\theta_u}\mathbf{C}_u^{\mathsf{T}} & \dfrac{1}{\theta_u} - \dfrac{\boldsymbol{\theta}_u^{\mathsf{T}}\mathbf{C}_u\boldsymbol{\theta}_u}{\theta_u^2}\end{bmatrix}\end{aligned} \tag{2.28}$$

where the permutation matrix $\mathbf{P}$ is used to arrange updated and fixed elements in block partitions, so that the submatrix $\mathbf{\Theta}_u$ represents the elements that remain unchanged, while vector $\boldsymbol{\theta}_u$ and scalar θ_u (i.e., $\theta_u = (\mathbf{\Theta})_{uu}$) are the u-th row/columns $\mathbf{\Theta}$, which are being updated. Based on the block partitions in (2.28), we can calculate $\mathbf{C}$, using updated $\boldsymbol{\theta}_u$ and θ_u, for fixed $\mathbf{\Theta}_u$ as follows:

$$\mathbf{C}_u = \left(\mathbf{\Theta}_u - \frac{\boldsymbol{\theta}_u\boldsymbol{\theta}_u^{\mathsf{T}}}{\theta_u}\right)^{-1} = \mathbf{\Theta}_u^{-1} - \frac{\mathbf{\Theta}_u^{-1}\boldsymbol{\theta}_u\boldsymbol{\theta}_u^{\mathsf{T}}\mathbf{\Theta}_u^{-1}}{\theta_u - \boldsymbol{\theta}_u^{\mathsf{T}}\mathbf{\Theta}_u^{-1}\boldsymbol{\theta}_u}, \tag{2.29}$$

$$\mathbf{c}_u = -\mathbf{C}_u\frac{\boldsymbol{\theta}_u}{\theta_u} = -\frac{\mathbf{\Theta}_u^{-1}\boldsymbol{\theta}_u}{\theta_u - \boldsymbol{\theta}_u^{\mathsf{T}}\mathbf{\Theta}_u^{-1}\boldsymbol{\theta}_u}, \tag{2.30}$$

$$c_u = \frac{1}{\theta_u - \boldsymbol{\theta}_u^{\mathsf{T}}\mathbf{\Theta}_u^{-1}\boldsymbol{\theta}_u}, \tag{2.31}$$

where $\mathbf{\Theta}_u^{-1}$ can be calculated from partitions of updated $\mathbf{C}$ as,

$$\mathbf{\Theta}_u^{-1} = \mathbf{C}_u - \mathbf{c}_u\mathbf{c}_u^{\mathsf{T}}/c_u. \tag{2.32}$$

Diagonal updates. After adding a scalar value ν to $(\mathbf{\Theta})_{ii}$, we use the Sherman-Morrison formula [66] to update $\mathbf{C}$ as

$$\widehat{\mathbf{C}} = \widehat{\mathbf{\Theta}}^{-1} = (\mathbf{\Theta} + \nu\,\boldsymbol{\delta}_i\boldsymbol{\delta}_i^{\mathsf{T}})^{-1} = \mathbf{C} - \frac{\nu\,\mathbf{C}\boldsymbol{\delta}_i\boldsymbol{\delta}_i^{\mathsf{T}}\mathbf{C}}{1+\nu\boldsymbol{\delta}_i^{\mathsf{T}}\mathbf{C}\boldsymbol{\delta}_i}, \tag{2.33}$$

where $\boldsymbol{\delta}_i$ is the vector whose entries are zero, except for its i-th entry which is equal to one.

2.4.2 Generalized Laplacian Estimation

Derivation of the optimality conditions. To derive necessary and sufficient optimality conditions, we use Lagrangian duality theory [67, 61], 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,

$$-\operatorname{logdet}(\mathbf{\Theta}) + \operatorname{Tr}(\mathbf{\Theta}\mathbf{K}) + \operatorname{Tr}(\mathbf{\Theta}\mathbf{M}), \tag{2.34}$$

where $\mathbf{M}$ is the matrix of Lagrange multipliers associated with the constraints. In particular, for Problem 1, $\mathbf{M}=\mathbf{M}_1+\mathbf{M}_2$, with multiplier matrices, $\mathbf{M}_1$ and $\mathbf{M}_2$, whose entries are

$$(\mathbf{M}_1)_{ij} = (\mathbf{M}_1)_{ji} = \begin{cases} \mu_{ij}^{(1)} \geq 0 & \text{if } (\mathbf{A})_{ij} = 1,\ i \neq j \\ 0 & \text{if } (\mathbf{A})_{ij} = 0,\ i \neq j \\ 0 & \text{if } i = j \end{cases} \tag{2.35}$$

$$(\mathbf{M}_2)_{ij} = (\mathbf{M}_2)_{ji} = \begin{cases} \mu_{ij}^{(2)} \in \mathbb{R} & \text{if } (\mathbf{A})_{ij} = 0,\ i \neq j \\ 0 & \text{if } (\mathbf{A})_{ij} = 1,\ i \neq j \\ 0 & \text{if } i = j \end{cases} \tag{2.36}$$

for $i,j=1,\dots,n$ where $\mu_{ij}^{(1)}$ and $\mu_{ij}^{(2)}$ are the Lagrange multipliers associated with inequality and equality constraints in Problem 1, respectively. For Problem 2, $\mathbf{M}=\mathbf{M}_1+\mathbf{M}_2-\mathbf{M}_3$ so that $\mathbf{M}_1$ and $\mathbf{M}_2$ are as in (2.35) and (2.36), and $\mathbf{M}_3$ consists of Lagrange multipliers (denoted as $\mu_i^{(3)}$) associated with the constraint $\mathbf{\Theta}\mathbf{1} \geq \mathbf{0}$. The entries of $\mathbf{M}_3$ are

$$(\mathbf{M}_3)_{ij} = \mu_i^{(3)} + \mu_j^{(3)} \text{ where } \mu_i^{(3)} \geq 0,\ \mu_j^{(3)} \geq 0 \tag{2.37}$$

for $i,j=1,\dots,n$. By taking the derivative of (2.34) with respect to $\mathbf{\Theta}$ and setting it to zero, we obtain the following optimality condition,

$$-\mathbf{\Theta}^{-1} + \mathbf{K} + \mathbf{M} = \mathbf{O}, \tag{2.38}$$

and the necessary and sufficient optimality conditions [61] for Problem 1 are

$$\begin{gathered} -\mathbf{\Theta}^{-1} + \mathbf{K} + \mathbf{M} = \mathbf{O} \\ (\mathbf{\Theta})_{ij} \leq 0 \text{ if } (\mathbf{A})_{ij} = 1,\ \ i \neq j \\ (\mathbf{\Theta})_{ij} = 0 \text{ if } (\mathbf{A})_{ij} = 0,\ \ i \neq j \\ \mathbf{\Theta} \succeq 0 \quad (\mathbf{M}_1)_{ij}(\mathbf{\Theta})_{ij} = 0 \end{gathered} \tag{2.39}$$

where $\mathbf{M}=\mathbf{M}_1+\mathbf{M}_2$. For Problem 2, the multiplier matrix is $\mathbf{M}=\mathbf{M}_1+\mathbf{M}_2-\mathbf{M}_3$, and the corresponding optimality conditions include those in (2.39) as well as:

$$\mathbf{\Theta}\mathbf{1} \geq \mathbf{0} \quad (\mathbf{M}_3)_{ii}(\mathbf{\Theta}\mathbf{1})_i = 0. \tag{2.40}$$

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 $\mathbf{C} = \boldsymbol{\Theta}^{-1}$ and using the conditions in (2.39), the optimality conditions for the u-th row/column of $\boldsymbol{\Theta}$ can be written as:

$$-\mathbf{c}_u + \mathbf{k}_u + \mathbf{m}_u = \mathbf{0} \tag{2.41}$$

$$-c_u + k_u = 0 \tag{2.42}$$

where the vectors, $\mathbf{c}_u$, $\mathbf{k}_u$ and $\mathbf{m}_u$, and the scalars, c_u, k_u and m_u, are obtained by partitioning $\mathbf{C}$, $\mathbf{K}$ and $\mathbf{M}$, as in (2.28) such that $c_u = (\mathbf{C})_{uu}$, $k_u = (\mathbf{K})_{uu}$ and $m_u = (\mathbf{M})_{uu} = 0$. By using the relations in (2.30) and (2.31), we can rewrite (2.41) as

$$\boldsymbol{\Theta}_u^{-1}\boldsymbol{\theta}_u c_u + \mathbf{k}_u + \mathbf{m}_u = \mathbf{0}. \tag{2.43}$$

Based on the relations in (2.39) and (2.42) the optimality conditions for the u-th column of $\boldsymbol{\Theta}$ (i.e., $\boldsymbol{\theta}_u$) include

$$\begin{aligned} &\boldsymbol{\Theta}_u^{-1}\boldsymbol{\theta}_u k_u + \mathbf{k}_u + \mathbf{m}_u = \mathbf{0} \\ &(\boldsymbol{\theta}_u)_i \le 0 \;\text{ if } (\mathbf{a}_u)_i = 1 \\ &(\boldsymbol{\theta}_u)_i = 0 \;\text{ if } (\mathbf{a}_u)_i = 0 \end{aligned} \tag{2.44}$$

where $\boldsymbol{\theta}_u$ and $\mathbf{a}_u$ are obtained by partitioning $\boldsymbol{\Theta}$ and $\mathbf{A}$ as in (2.28), respectively, and the optimality conditions on $\mathbf{m}_u$ follow from (2.35) and (2.36). Based on the above optimality conditions, in (2.39) and (2.44), the optimal update for the u-th row/column of $\boldsymbol{\Theta}$ (i.e., $\boldsymbol{\theta}_u$) corresponds to the solution of the following quadratic program:

$$\begin{aligned} \underset{\boldsymbol{\theta}_u}{\text{minimize}} \quad & \frac{1}{2}k_u^2\boldsymbol{\theta}_u^{\mathsf{T}}\boldsymbol{\Theta}_u^{-1}\boldsymbol{\theta}_u + k_u\boldsymbol{\theta}_u^{\mathsf{T}}\mathbf{k}_u \\ \text{subject to} \quad & (\boldsymbol{\theta}_u)_i \le 0 \;\text{ if } (\mathbf{a}_u)_i = 1 \\ & (\boldsymbol{\theta}_u)_i = 0 \;\text{ if } (\mathbf{a}_u)_i = 0 \end{aligned} \tag{2.45}$$

The above problem can be simplified by eliminating its equality constraints determined by $\mathbf{A}$ (i.e., $\mathbf{a}_u$), so that we formulate an equivalent version of (2.45) as the following nonnegative quadratic program [68], whose solution satisfies the optimality conditions in (2.44),

$$\begin{aligned} \underset{\boldsymbol{\beta}}{\text{minimize}} \quad & \frac{1}{2}\boldsymbol{\beta}^{\mathsf{T}}\mathbf{Q}\boldsymbol{\beta} - \boldsymbol{\beta}^{\mathsf{T}}\mathbf{p} \\ \text{subject to} \quad & \boldsymbol{\beta} \ge \mathbf{0} \end{aligned} \tag{2.46}$$

where

$$\begin{gathered} \boldsymbol{\beta} = -(\boldsymbol{\theta}_u)_{\mathcal{S}} \quad \mathbf{p} = (\mathbf{k}_u/k_u)_{\mathcal{S}} \quad \mathbf{Q} = (\boldsymbol{\Theta}_u^{-1})_{\mathcal{S}\mathcal{S}} \\ \mathcal{S} = \{i \,|\, (\mathbf{a}_u)_i = 1\} \end{gathered} \tag{2.47}$$

so that $\boldsymbol{\beta}$ is the vector whose elements are selected from the original variable vector $\boldsymbol{\theta}_u$ based on index set $\mathcal{S}$. For example, if $\mathcal{S}=\{1,2,5\}$, then $\boldsymbol{\beta}=-[(\boldsymbol{\theta}_u)_1\,(\boldsymbol{\theta}_u)_2\,(\boldsymbol{\theta}_u)_5]^{\mathsf{T}}$. Similarly, $\mathbf{Q}$ is constructed by selecting rows and columns of $\boldsymbol{\Theta}_u^{-1}$ with index values in $\mathcal{S}$, so the resulting $\mathbf{Q}$ is a submatrix of $\boldsymbol{\Theta}_u^{-1}$. It is important to note that the connectivity constraints (i.e., $\mathbf{A}$ or $\mathbf{a}_u$) allow us to reduce the dimension of the variable $\boldsymbol{\theta}_u$ and therefore, the dimension of (2.45).

For Problem 2, based on the conditions in (2.39) and (2.40), we can similarly formulate a quadratic program to update u-th row/column of $\boldsymbol{\Theta}$:

$$\begin{aligned} \underset{\boldsymbol{\theta}_u}{\text{minimize}} \quad & \frac{1}{2}c_u^2\boldsymbol{\theta}_u^{\mathsf{T}}\boldsymbol{\Theta}_u^{-1}\boldsymbol{\theta}_u + c_u\boldsymbol{\theta}_u^{\mathsf{T}}\mathbf{k}_u \\ \text{subject to} \quad & (\boldsymbol{\theta}_u)_i \le 0 \ \text{ if } (\mathbf{a}_u)_i = 1 \\ & (\boldsymbol{\theta}_u)_i = 0 \ \text{ if } (\mathbf{a}_u)_i = 0 \\ & -\boldsymbol{\theta}_u^{\mathsf{T}}\mathbf{1} \le \theta_u \end{aligned} \tag{2.48}$$

where $\theta_u = (\boldsymbol{\Theta})_{uu}$. The above problem is also a nonnegative quadratic program. To solve (2.48) for all u, we first iteratively update each row/column of $\boldsymbol{\Theta}$ by solving the subproblem in (2.46). After completing a cycle of n row/column updates, we modify the diagonal entries of the updated $\boldsymbol{\Theta}$, so that it satisfies the constraints in (2.48). The diagonal update parameters ($\boldsymbol{\nu}$) are the optimal solutions of the following projection problem for given $\boldsymbol{\Theta}$:

$$\begin{aligned} \underset{\boldsymbol{\nu}}{\text{minimize}} \quad & \|\boldsymbol{\Theta}-\widehat{\boldsymbol{\Theta}}\|_F^2 \\ \text{subject to} \quad & \widehat{\boldsymbol{\Theta}} = \boldsymbol{\Theta} + \operatorname{diag}(\boldsymbol{\nu}) \quad \widehat{\boldsymbol{\Theta}} \in \mathcal{L}_d \end{aligned} \tag{2.49}$$

where $\boldsymbol{\nu}$ is the vector of update parameters, and $\mathcal{L}_d$ 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 $\mathbf{A}$, type of desired Laplacian matrix (i.e., $\mathcal{L}_g$ or $\mathcal{L}_d$) and regularization matrix $\mathbf{H}$. Basically, the proposed algorithm iteratively updates each row/column of the working estimate of Laplacian matrix ($\widehat{\boldsymbol{\Theta}}$) and its inverse ($\widehat{\mathbf{C}}$) by solving the subproblem in (2.46). The main reason of updating $\mathbf{C}$ is that the derived subproblem is parametrized by $\boldsymbol{\Theta}_u^{-1}$, which depends on $\mathbf{C}$ as formulated in (2.32). 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 (2.29)–(2.32). If we are interested in solving Problem 1 (if $\mathcal{L}=\mathcal{L}_g$), then the algorithm skips the lines 13–19. For Problem 2 (for $\mathcal{L}=\mathcal{L}_d$) the `for` loop between the lines 14–18 iteratively modifies the diagonal elements of $\widehat{\boldsymbol{\Theta}}$ by solving the projection problem in (2.49) ensuring that the resulting $\widehat{\boldsymbol{\Theta}}$ is a diagonally dominant matrix. The inverse of $\widehat{\boldsymbol{\Theta}}$ (i.e., $\widehat{\mathbf{C}}$) is also iteratively updated, accordingly (see line 16). The overall procedure is repeated until a stopping criterion (line 20) has been satisfied.

Algorithm 1 Generalized Graph Laplacian (GGL)

```
Input: Sample statistic $\mathbf{S}$, connectivity matrix $\mathbf{A}$, regularization matrix $\mathbf{H}$, target Laplacian set $\mathcal{L}$
    and tolerance $\epsilon$
Output: $\mathbf{\Theta}$ and $\mathbf{C}$
 1: Set $\mathbf{K} = \mathbf{S} + \mathbf{H}$
 2: Initialize $\widehat{\mathbf{C}} = \mathrm{ddiag}(\mathbf{K})$ and $\widehat{\mathbf{\Theta}} = \widehat{\mathbf{C}}^{-1}$
 3: repeat
 4:     Set $\widehat{\mathbf{\Theta}}_{\mathrm{pre}} = \widehat{\mathbf{\Theta}}$
 5:     for $u = 1$ to $n$ do
 6:         Partition $\widehat{\mathbf{\Theta}}$, $\widehat{\mathbf{C}}$, $\mathbf{K}$ and $\mathbf{A}$ as in (2.28) for $u$
 7:         Update $\widehat{\mathbf{\Theta}}_u^{-1} = \widehat{\mathbf{C}}_u - \widehat{\mathbf{c}}_u\widehat{\mathbf{c}}_u^{\mathsf{T}}/\widehat{c}_u$
 8:         Solve (2.46) for $\boldsymbol{\beta}$ with $\mathbf{Q}$, $\mathbf{p}$ and $\mathcal{S}$ in (2.47)
 9:         Update $\widehat{\boldsymbol{\theta}}_u$ and $\widehat{\theta}_u$ using the solution $\widehat{\boldsymbol{\beta}}$ from above:
                $(\widehat{\boldsymbol{\theta}}_u)_{\mathcal{S}} = -\widehat{\boldsymbol{\beta}} \quad (\widehat{\boldsymbol{\theta}}_u)_{\mathcal{S}^c} = \mathbf{0}$
                $\widehat{\theta}_u = 1/k_u + \widehat{\boldsymbol{\beta}}^{\mathsf{T}}\mathbf{Q}\widehat{\boldsymbol{\beta}}$
10:         Update $\widehat{\mathbf{c}}_u$, $\widehat{c}_u$ and $\widehat{\mathbf{C}}_u$:
                $\widehat{c}_u = 1/(\widehat{\theta}_u - \widehat{\boldsymbol{\theta}}_u^{\mathsf{T}}\widehat{\mathbf{\Theta}}_u^{-1}\widehat{\boldsymbol{\theta}}_u) \quad \widehat{\mathbf{c}}_u = \widehat{\mathbf{\Theta}}_u^{-1}\widehat{\boldsymbol{\theta}}_u/\widehat{c}_u$
                $\widehat{\mathbf{C}}_u = \widehat{\mathbf{\Theta}}_u^{-1} + \widehat{\mathbf{c}}_u\widehat{\mathbf{c}}_u^{\mathsf{T}}/\widehat{c}_u$
11:         Rearrange $\widehat{\mathbf{\Theta}}$ and $\widehat{\mathbf{C}}$ using $\mathbf{P}$ for $u$ as in (2.28)
12:     end for
13:     if $\mathcal{L} = \mathcal{L}_d$ (target Laplacian is a DDGL) then
14:         for $i = 1$ to $n$ do
15:             if $(\widehat{\mathbf{\Theta}}\mathbf{1})_i < 0$ then $\nu = -(\widehat{\mathbf{\Theta}}\mathbf{1})_i$
16:             Set $(\widehat{\mathbf{\Theta}})_{ii} = (\widehat{\mathbf{\Theta}})_{ii} + \nu$ and update $\widehat{\mathbf{C}}$ using (2.33)
17:             end if
18:         end for
19:     end if
20: until criterion$(\widehat{\mathbf{\Theta}}, \widehat{\mathbf{\Theta}}_{\mathrm{pre}}) \le \epsilon$
21: return $\mathbf{\Theta} = \widehat{\mathbf{\Theta}}$ and $\mathbf{C} = \widehat{\mathbf{C}}$
```

2.4.3 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,

$$-\operatorname{logdet}(\mathbf{\Theta} + \mathbf{J}) + \operatorname{Tr}\left(\mathbf{\Theta}(\mathbf{K} + \mathbf{J})\right) + \operatorname{Tr}(\mathbf{\Theta}\mathbf{M}), \tag{2.50}$$

where $\mathbf{M} = \mathbf{M}_1 + \mathbf{M}_2 + \mathbf{M}_4$ consists of Lagrange multipliers associated with the constraints in (2.9) such that $\mathbf{M}_1$ and $\mathbf{M}_2$ are as defined in (2.35) and (2.36), and the entries of $\mathbf{M}_4$ are

$$(\mathbf{M}_4)_{ij} = \mu_i^{(4)} + \mu_j^{(4)} \ \text{ where } \mu_i^{(4)} \in \mathbb{R},\ \mu_j^{(4)} \in \mathbb{R} \tag{2.51}$$

Algorithm 2 Combinatorial Graph Laplacian (CGL)

```
Input: Sample statistic $\mathbf{S}$, connectivity matrix $\mathbf{A}$, regularization matrix $\mathbf{H}$ and tolerance $\epsilon$
Output: $\mathbf{\Theta}$ and $\mathbf{C}$
 1: Set $\mathbf{J}=(1/n)\mathbf{1}\mathbf{1}^{\mathsf{T}}$   $\widetilde{\mathbf{K}}=\mathbf{S}+\mathbf{H}+\mathbf{J}$
 2: Initialize $\widehat{\mathbf{C}} = \operatorname{ddiag}(\widetilde{\mathbf{K}})$ and $\widehat{\mathbf{\Theta}} = \widehat{\mathbf{C}}^{-1}$
 3: repeat
 4:     Set $\widehat{\mathbf{\Theta}}_{\text{pre}} = \widehat{\mathbf{\Theta}}$
 5:     for $u = 1$ to $n$ do
 6:         Partition $\widehat{\mathbf{\Theta}}$, $\widehat{\mathbf{C}}$, $\widetilde{\mathbf{K}}$ and $\mathbf{A}$ as in (2.28) for $u$
 7:         Calculate $\widehat{\mathbf{\Theta}}_u^{-1} = \widehat{\mathbf{C}}_u - \widehat{\mathbf{c}}_u\widehat{\mathbf{c}}_u^{\mathsf{T}}/\widehat{c}_u$
 8:         Solve (2.58) for $\boldsymbol{\beta}$ with $\mathbf{Q}$, $\mathbf{p}$ and $\mathcal{S}$ in (2.59)
 9:         Update $\widehat{\boldsymbol{\theta}}_u$ and $\widehat{\theta}_u$ using the solution $\widehat{\boldsymbol{\beta}}$ from above:
                $(\widehat{\boldsymbol{\theta}}_u)_{\mathcal{S}} = ((1/n)\mathbf{1} - \widehat{\boldsymbol{\beta}})$   $(\widehat{\boldsymbol{\theta}}_u)_{\mathcal{S}^c} = (1/n)\mathbf{1}$
                $\widehat{\theta}_u = 1/\widetilde{k}_u + (\widehat{\boldsymbol{\beta}} - (1/n)\mathbf{1})^{\mathsf{T}}\mathbf{Q}(\widehat{\boldsymbol{\beta}} - (1/n)\mathbf{1})$
10:         Update $\widehat{\mathbf{c}}_u$, $\widehat{c}_u$ and $\widehat{\mathbf{C}}_u$:
                $\widehat{c}_u = 1/(\widehat{\theta}_u - \widehat{\boldsymbol{\theta}}_u^{\mathsf{T}}\widehat{\mathbf{\Theta}}_u^{-1}\widehat{\boldsymbol{\theta}}_u)$   $\widehat{\mathbf{c}}_u = \widehat{\mathbf{\Theta}}_u^{-1}\widehat{\boldsymbol{\theta}}_u/\widehat{c}_u$
                $\widehat{\mathbf{C}}_u = \widehat{\mathbf{\Theta}}_u^{-1} + \widehat{\mathbf{c}}_u\widehat{\mathbf{c}}_u^{\mathsf{T}}/\widehat{c}_u$
11:         Rearrange $\widehat{\mathbf{\Theta}}$ and $\widehat{\mathbf{C}}$ using $\mathbf{P}$ for $u$ as in (2.28)
12:     end for
13:     for $i = 1$ to $n$ do
14:         if $(\widehat{\mathbf{\Theta}}\mathbf{1})_i - 1 \neq 0$ then $\nu = -(\widehat{\mathbf{\Theta}}\mathbf{1})_i + 1$
15:         Set $(\widehat{\mathbf{\Theta}})_{ii} = (\widehat{\mathbf{\Theta}})_{ii} + \nu$ and update $\widehat{\mathbf{C}}$ using (2.33)
16:         end if
17:     end for
18: until criterion$(\widehat{\mathbf{\Theta}}, \widehat{\mathbf{\Theta}}_{\text{pre}}) \leq \epsilon$
19: return $\mathbf{\Theta} = \widehat{\mathbf{\Theta}} - \mathbf{J}$ and $\mathbf{C} = \widehat{\mathbf{C}} - \mathbf{J}$
```

for $i, j = 1, \dots, n$. Based on the Lagrangian stated in (2.50), the necessary and sufficient optimality conditions for the problem in (2.9) are

$$\begin{aligned} -\widetilde{\mathbf{\Theta}}^{-1} + \widetilde{\mathbf{K}} + \mathbf{M}_1 + \mathbf{M}_2 + \mathbf{M}_4 &= \mathbf{O} \\ (\widetilde{\mathbf{\Theta}})_{ij} \leq 1/n \;\text{ if } (\mathbf{A})_{ij} = 1,\; i \neq j \\ (\widetilde{\mathbf{\Theta}})_{ij} = 1/n \;\text{ if } (\mathbf{A})_{ij} = 0,\; i \neq j \\ \widetilde{\mathbf{\Theta}} \succeq 0 \quad \widetilde{\mathbf{\Theta}}\mathbf{1} = \mathbf{1} \quad (\mathbf{M}_1)_{ij}\,((\widetilde{\mathbf{\Theta}})_{ij} - 1/n) = 0 \end{aligned} \tag{2.52}$$

where $\widetilde{\mathbf{\Theta}} = \mathbf{\Theta} + \mathbf{J}$, $\widetilde{\mathbf{C}} = (\mathbf{\Theta} + \mathbf{J})^{-1}$ and $\widetilde{\mathbf{K}} = \mathbf{K} + \mathbf{J}$. The matrices $\mathbf{M}_1$, $\mathbf{M}_2$ and $\mathbf{M}_4$ are defined as in (2.35), (2.36) and (2.51), respectively. For the u-th row/column of $\widetilde{\mathbf{\Theta}}$, the first optimality condition in (2.52) reduces to

$$-\widetilde{\mathbf{c}}_u + \widetilde{\mathbf{k}}_u + \mathbf{m}_u = \mathbf{0} \tag{2.53}$$

$$-\widetilde{c}_u + \widetilde{k}_u + m_u = 0 \tag{2.54}$$

where the condition in (2.53) can also be stated using the relations in (2.30) and (2.31) as

$$\widetilde{\mathbf{\Theta}}_u^{-1}\widetilde{\boldsymbol{\theta}}_u\widetilde{c}_u + \widetilde{\mathbf{k}}_u + \mathbf{m}_u = \mathbf{0}. \tag{2.55}$$

Subproblem for block-coordinate descent updates. Based on the optimality conditions stated in (2.52) and (2.55), we derive the following quadratic program solved for updating u-th row/column of $\widetilde{\mathbf{\Theta}}$,

$$\begin{aligned} \underset{\widetilde{\boldsymbol{\theta}}_u}{\text{minimize}} \quad & \frac{1}{2}\widetilde{c}_u^2\widetilde{\boldsymbol{\theta}}_u^\intercal\widetilde{\mathbf{\Theta}}_u^{-1}\widetilde{\boldsymbol{\theta}}_u + \widetilde{c}_u\widetilde{\boldsymbol{\theta}}_u^\intercal\widetilde{\mathbf{k}}_u \\ \text{subject to} \quad & (\widetilde{\boldsymbol{\theta}}_u)_i \leq 1/n \;\; \text{if } (\mathbf{a}_u)_i = 1 \\ & (\widetilde{\boldsymbol{\theta}}_u)_i = 1/n \;\; \text{if } (\mathbf{a}_u)_i = 0 \\ & -(\widetilde{\boldsymbol{\theta}}_u - (1/n)\mathbf{1})^\intercal\mathbf{1} = \widetilde{\theta}_u - (1/n) \end{aligned} \tag{2.56}$$

By changing variables $\widetilde{\boldsymbol{\theta}}_u = \boldsymbol{\theta}_u + (1/n)\mathbf{1}$, $\widetilde{\theta}_u = \theta_u + (1/n)$ and dividing the objective function with $\widetilde{c}_u^2$, we rewrite (2.56) as a quadratic program of the standard form,

$$\begin{aligned} \underset{\boldsymbol{\theta}_u}{\text{minimize}} \quad & \frac{1}{2}\boldsymbol{\theta}_u^\intercal\widetilde{\mathbf{\Theta}}_u^{-1}\boldsymbol{\theta}_u + \boldsymbol{\theta}_u^\intercal\left(\frac{\widetilde{\mathbf{k}}_u}{\widetilde{c}_u} + \frac{1}{n}\widetilde{\mathbf{\Theta}}_u^{-1}\mathbf{1}\right) \\ \text{subject to} \quad & (\boldsymbol{\theta}_u)_i \leq 0 \;\; \text{if } (\mathbf{a}_u)_i = 1 \\ & (\boldsymbol{\theta}_u)_i = 0 \;\; \text{if } (\mathbf{a}_u)_i = 0 \\ & -\boldsymbol{\theta}_u^\intercal\mathbf{1} = \theta_u \end{aligned} \tag{2.57}$$

which can be simplified by eliminating the equality constraints as follows,

$$\begin{aligned} \underset{\boldsymbol{\beta}}{\text{minimize}} \quad & \frac{1}{2}\boldsymbol{\beta}^\intercal\mathbf{Q}\boldsymbol{\beta} - \boldsymbol{\beta}^\intercal\mathbf{p} \\ \text{subject to} \quad & \boldsymbol{\beta} \geq \mathbf{0} \end{aligned} \tag{2.58}$$

where

$$\begin{aligned} \boldsymbol{\beta} = -(\boldsymbol{\theta}_u)_{\mathcal{S}} \quad & \mathbf{p} = (\widetilde{\mathbf{k}}_u/\widetilde{k}_u + (1/n)\widetilde{\mathbf{\Theta}}_u^{-1}\mathbf{1})_{\mathcal{S}} \\ \mathbf{Q} = (\widetilde{\mathbf{\Theta}}_u^{-1})_{\mathcal{S}\mathcal{S}} \quad & \mathcal{S} = \{i \,|\, (\mathbf{a}_u)_i = 1\}. \end{aligned} \tag{2.59}$$

In order to solve (2.57) for all u, we first iteratively update each row/column by solving the nonnegative quadratic program in (2.58). After each cycle of n row/column updates, the diagonal entries of the resulting matrix ($\widetilde{\mathbf{\Theta}}$) are modified to satisfy the combinatorial Laplacian constraints $-\boldsymbol{\theta}_u^\intercal\mathbf{1}=\theta_u$ for $u=1,\dots,n$ in (2.57). The diagonal update parameters are optimized by solving the following projection problem for given $\widetilde{\mathbf{\Theta}}$

$$\begin{aligned} \underset{\boldsymbol{\nu}}{\text{minimize}} \quad & \|\widetilde{\mathbf{\Theta}} - \widehat{\mathbf{\Theta}}\|_F^2 \\ \text{subject to} \quad & \widehat{\mathbf{\Theta}} = \widetilde{\mathbf{\Theta}} + \text{diag}(\boldsymbol{\nu}) \quad (\widehat{\mathbf{\Theta}} - \mathbf{J}) \in \mathcal{L}_c \end{aligned} \tag{2.60}$$

where $\boldsymbol{\nu}$ is the vector of update parameters, $\mathcal{L}_c$ denotes the set of combinatorial Laplacian matrices and $\mathbf{J} = (1/n)\mathbf{1}\mathbf{1}^{\mathsf{T}}$.

Proposed Algorithm. Algorithm 2 is proposed to solve Problem 3 for a given connectivity matrix $\mathbf{A}$ and regularization matrix $\mathbf{H}$. 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., $\boldsymbol{\Theta}$). Instead, it iteratively solves for the matrix $\widetilde{\boldsymbol{\Theta}} = \boldsymbol{\Theta} + \mathbf{J}$ whose entries are shifted by $(1/n)$. Secondly, 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 $\widetilde{\boldsymbol{\Theta}}$ leads to a CGL matrix (via the $\widehat{\boldsymbol{\Theta}} - \mathbf{J}$ transformation) by solving the projection problem in (2.60).

In Algorithm 2, we propose to estimate the shifted matrix $\widetilde{\boldsymbol{\Theta}}$ 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 (2.8) by solving the equivalent problem in (2.9). Based on this result, we derive optimality conditions for (2.9), and develop Algorithm 2 which iteratively estimates $\widetilde{\boldsymbol{\Theta}}$ until the convergence criterion has been satisfied. Then, the optimal $\boldsymbol{\Theta}$ is recovered by subtracting $\mathbf{J}$ from the estimated $\widetilde{\boldsymbol{\Theta}}$ as in line 19 of Algorithm 2.

2.4.4 Convergence and Complexity Analysis of Algorithms

Convergence. The following proposition addresses the convergence of our proposed algorithms based on the results from [58, 67, 69].

Proposition 6. *Algorithms 1 and 2 guarantee convergence to the global optimal solutions of Problems 1, 2 and 3.*

Proof. 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 [67, 58]. As stated in Proposition 2, all of our problems of interest are convex. Also, the objective functions are differentiable (see in Section 2.4). Moreover, convergence conditions (see in [67]) 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 $\mathbf{Q}$ matrix in (2.46) and (2.58) is positive definite throughout all iterations. To prove this, we use the following Schur complement condition on partitions of $\boldsymbol{\Theta}$ as in (2.28),

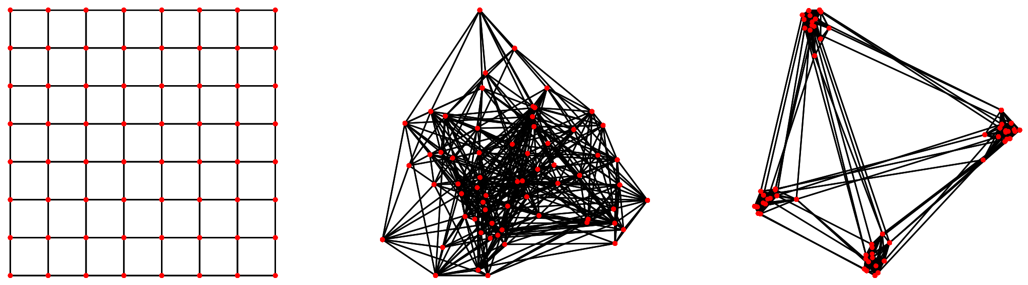
$$\boldsymbol{\Theta} \succ 0 \iff \boldsymbol{\Theta}_u \succ 0 \text{ and } \theta_u - \boldsymbol{\theta}_u^{\mathsf{T}} \boldsymbol{\Theta}_u^{-1} \boldsymbol{\theta}_u > 0. \tag{2.61}$$

where $\boldsymbol{\Theta}_u$ (and therefore, $\boldsymbol{\Theta}_u^{-1}$) is fixed and positive definite from the previous iteration. Noting that both algorithms initialize $\mathbf{\Theta}$ and $\mathbf{C}$ as (diagonal) positive definite matrices, $\theta_u - \boldsymbol{\theta}_u^\mathsf{T}\mathbf{\Theta}_u^{-1}\boldsymbol{\theta}_u = 1/c_u > 0$ by (2.31), then the updated $\mathbf{\Theta}$ is also positive definite. Since both algorithms solve for the optimal row/column updates at each iteration, based on the result in [67], 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 [69]. Also, it is trivial to show that additional updates on diagonal elements (see in (2.33)) maintain positive definiteness of $\mathbf{\Theta}$ and $\mathbf{C}$. Therefore, both algorithms guarantee convergence to the optimal solution. $\square$

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 $\widehat{\mathbf{\Theta}}_u^{-1}$. After a cycle of n row/column updates, updating a diagonal entry of $\widehat{\mathbf{\Theta}}$ and its inverse, $\widehat{\mathbf{C}}$, 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 $\mathbf{\Theta}$). Moreover, both proposed algorithms use diagonal matrices to initialize $\widehat{\mathbf{\Theta}}$ and $\widehat{\mathbf{C}}$. In practice, better initializations (i.e., "warm starts") and randomized implementations [58] can be exploited to reduce the algorithm runtime. To solve the subproblems in (2.46) and (2.58), which are specifically nonnegative quadratic programs, we employ an extension of Lawson-Hanson algorithm [70] with a block principal pivoting method [71]. 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 2.5, we present empirical complexity results for the proposed algorithms.

2.5 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) [30]. Algorithm 2 (CGL) is compared to the methods proposed for estimating shifted CGL matrices (SCGL) [33] and learning graphs from smooth signals (GLS) [31, 32], as well as the graph topology inference approach (GTI) proposed in [34]. 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



(a) $\mathcal{G}_{\text{grid}}^{(64)}$: 8×8 grid graph (b) $\mathcal{G}_{\text{ER}}^{(64,0.2)}$: Erdos-Renyi graph (c) $\mathcal{G}_{\text{M}}^{(64,0.2,0.9)}$: Modular graph with 4 modules

Figure 2.2: Examples of different graph connectivity models, (a) grid (b) Erdos-Renyi (c) modular graphs used in our experiments. All graphs have 64 vertices.

connectivity constraints on graph Laplacian estimation.

2.5.1 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, $\mathbf{L}$, is used to generate independent data samples from the distribution $\mathsf{N}(\mathbf{0}, \mathbf{L}^{\dagger})$. A graph (i.e., $\mathbf{L}$) 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 Figure 2.2):

- Grid graph, $\mathcal{G}_{\text{grid}}^{(n)}$, consisting of n vertices attached to their four nearest neighbors (except the vertices at boundaries).
- Random Erdos-Renyi graph, $\mathcal{G}_{\text{ER}}^{(n,p)}$, with n vertices attached to other vertices with probability p.
- Random modular graph (also known as stochastic block model), $\mathcal{G}_{\text{M}}^{(n,p_1,p_2)}$ with n vertices and four modules (subgraphs) 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 $\mathsf{U}(0.1, 3)$. 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 $\mathsf{U}(0.1, 3)$, 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 [33, 34] 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 $\epsilon = 10^{-4}$,

$$\mathsf{criterion}(\widehat{\mathbf{\Theta}}, \widehat{\mathbf{\Theta}}_{\text{pre}}) = \frac{\|\widehat{\mathbf{\Theta}} - \widehat{\mathbf{\Theta}}_{\text{pre}}\|_F}{\|\widehat{\mathbf{\Theta}}_{\text{pre}}\|_F} \leq \epsilon \,, \tag{2.62}$$

where $\widehat{\mathbf{\Theta}}$ and $\widehat{\mathbf{\Theta}}_{\text{pre}}$ denote the graph parameters from current and previous steps, respectively (see Algorithms in 1 and 2).

In order to measure graph learning performance, we use two different metrics,

$$\mathsf{RE}(\widehat{\mathbf{\Theta}}, \mathbf{\Theta}^*) = \frac{\|\widehat{\mathbf{\Theta}} - \mathbf{\Theta}^*\|_F}{\|\mathbf{\Theta}^*\|_F} \tag{2.63}$$

which is the relative error between the ground truth graph ($\mathbf{\Theta}^*$) and estimated graph parameters ($\widehat{\mathbf{\Theta}}$), and

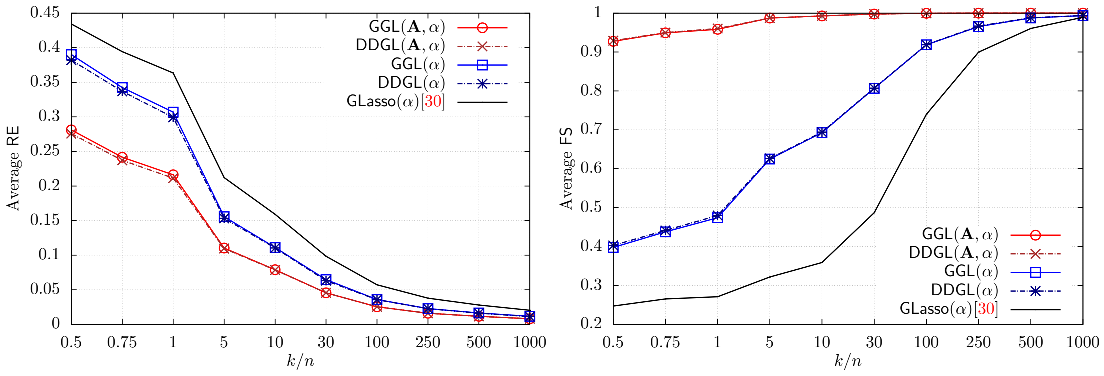
$$\mathsf{FS}(\widehat{\mathbf{\Theta}}, \mathbf{\Theta}^*) = \frac{2\,\mathsf{tp}}{2\,\mathsf{tp} + \mathsf{fn} + \mathsf{fp}} \tag{2.64}$$

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

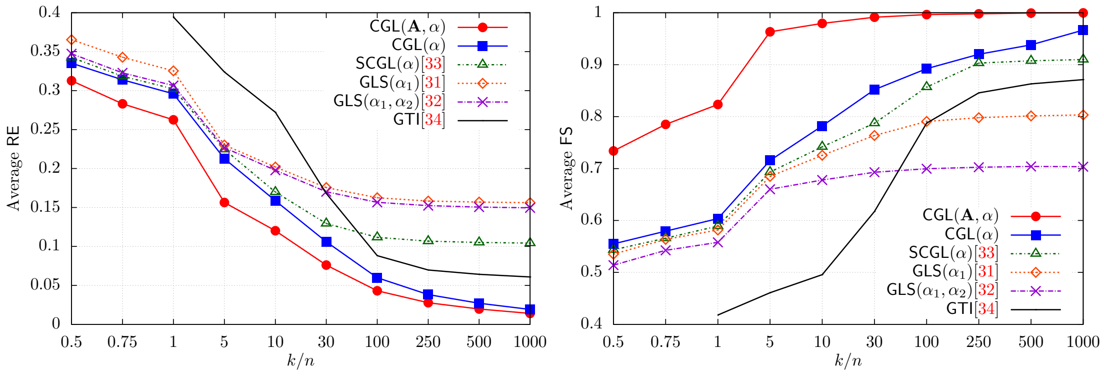
In our experiments, since SCGL and GLasso methods employ $\alpha\|\mathbf{\Theta}\|_1$ for regularization, we use the same regularization in our proposed methods (i.e., the matrix $\mathbf{H}$ is selected as in (2.3) for Problems 1, 2 and 3) to fairly compare all methods. Monte-Carlo simulations are performed for each proposed/baseline method, by successively solving the associated graph learning problem with different regularization parameters (i.e., α, α_1 and α_2) to find the (best) regularization minimizing RE. The corresponding graph estimate is also used to calculate FS. Particularly, for the GGL, DDGL, CGL, SCGL and GLasso methods, α is selected from the following set:

$$\{0\} \cup \{0.75^r (s_{\max}\sqrt{\log(n)/k}) \,|\, r = 1, 2, 3, \ldots, 14\}, \tag{2.65}$$

where $s_{\max} = \max_{i \neq j} |(\mathbf{S})_{ij}|$ is the maximum off-diagonal entry of $\mathbf{S}$ 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 [72, 73]. For both of the GLS methods [31, 32] addressing the problems in (2.15) and (2.16), α_1 is selected from $\{0\} \cup \{0.75^r s_{\max} \,|\, r = 1, 2, 3, \ldots, 14\}$, and for the problem in (2.16) the parameter α_2 is selected by fine tuning. For GLS [31], the $\mathrm{Tr}(\mathbf{\Theta}) = n$ constraint in (2.15) is set as $\mathrm{Tr}(\mathbf{\Theta}) = \mathrm{Tr}(\mathbf{\Theta}^*)$. Since GLS [32] and GTI [34] approaches generally result in severely biased solutions with respect to the ground truth $\mathbf{\Theta}^*$ (based on our observations from the experiments), their RE values are calculated after normalizing the estimated solution $\mathbf{\Theta}$ as $\widehat{\mathbf{\Theta}} = (\mathrm{Tr}(\mathbf{\Theta}^*)/\mathrm{Tr}(\mathbf{\Theta}))\mathbf{\Theta}$.



(a) Average performance results for learning generalized graph Laplacian matrices: The proposed methods GGL(α) and DDGL(α) outperform GLasso(α). The degree of improvement achieved by GGL($\mathbf{A}, \alpha$) and DDGL($\mathbf{A}, \alpha$), incorporating the connectivity constraints, is also demonstrated.



(b) Average performance results for learning combinatorial graph Laplacian matrices: The proposed CGL(α) method outperforms all baseline approaches, and the difference increases as gets k/n larger. Incorporating the connectivity constraints (i.e., CGL($\mathbf{A}, \alpha$)) naturally improves the performance.

Figure 2.3: Comparison of the (a) GGL and (b) CGL estimation methods: The algorithms are tested on grid ($\mathcal{G}^{(n)}_{\text{grid}}$) and random graphs ($\mathcal{G}^{(n,0.1)}_{\text{ER}}$ and $\mathcal{G}^{(n,0.1,0.3)}_{\text{M}}$).

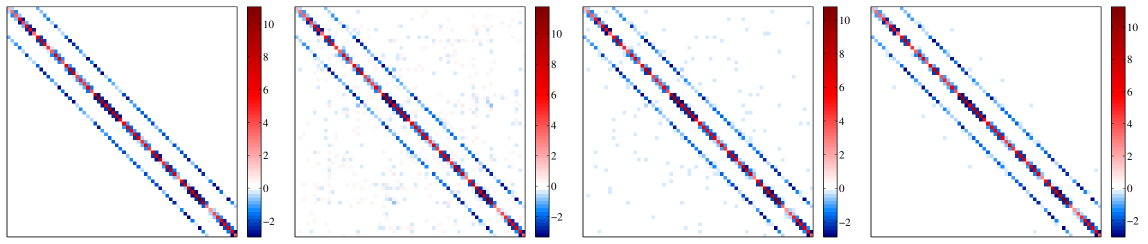
Results. Figure 2.3 demonstrates graph learning performances of different methods (in terms of average RE and FS) with respect to the number of data samples, used to calculate sample covariance $\mathbf{S}$, per number of vertices (i.e., k/n). In our results, GGL($\mathbf{A}, \alpha$), DDGL($\mathbf{A}, \alpha$) and CGL($\mathbf{A}, \alpha$) 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(α), DDGL(α) and CGL(α) refer to solving the problems with ℓ_1-regularization only (i.e., without connectivity constraints).

As shown in Figure 2.3, our proposed methods outperform all baseline approaches (namely GLasso [30], SCGL [33], GLS [31, 32] and GTI [34]) in terms of both RE and FS metrics. Naturally, incorporating the connectivity constraints (e.g., in GGL$(\mathbf{A},\alpha)$ and CGL$(\mathbf{A},\alpha)$) 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 $\mathbf{S}$ 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 ($\mathbf{A}$). As an example, for fixed $k/n=30$, Table 2.1 compares average RE results for different graph connectivity models and number of vertices (n for 64 and 256). Also, Figures 2.4 and 2.5 illustrate sample GGL and CGL estimation results and compare different methods.

Table 2.1: Average relative errors for different graph connectivity models and number of vertices with fixed $k/n = 30$

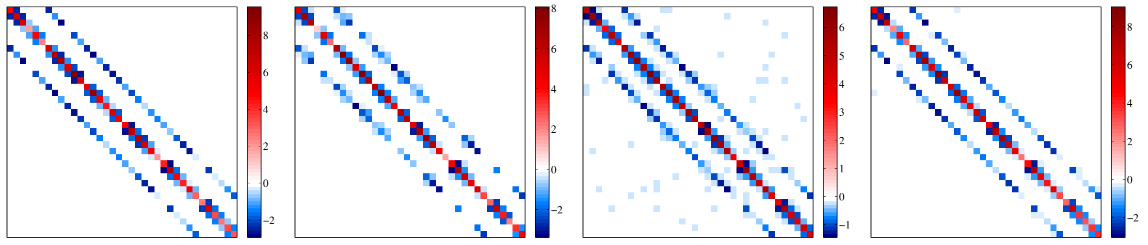
Connectivity models	Average RE$(n=64)$ $\mid$ Average RE$(n=256)$		
	GLasso(α)	GGL(α)	GGL$(\mathbf{A},\alpha)$
$\mathcal{G}^{(n)}_{\text{grid}}$	0.079 $\mid$ 0.078	0.053 $\mid$ 0.037	0.040 $\mid$ 0.027
$\mathcal{G}^{(n,0.1)}_{\text{ER}}$	0.105 $\mid$ 0.112	0.077 $\mid$ 0.082	0.053 $\mid$ 0.053
$\mathcal{G}^{(n,0.1,0.3)}_{\text{M}}$	0.102 $\mid$ 0.124	0.075 $\mid$ 0.081	0.051 $\mid$ 0.053

For estimation of GGL matrices, Figure 2.3a demonstrates that the best set of results are obtained by solving the DDGL problem, DDGL$(\mathbf{A},\alpha)$ and DDGL(α). 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$(\mathbf{A},\alpha)$ and GGL(α), also provides a good performance, where the difference between GGL and DDGL is often negligible. Moreover, both of the proposed GGL(α) and DDGL(α) 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, Figure 2.3b shows that CGL (i.e., Algorithm 2) provides significantly better performance with increasing number of data samples (k/n) as compared to the SCGL, GLS 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 [33] optimizes a shifted CGL instead of directly estimating a CGL matrix as stated in (2.13). The GLS and GTI methods solve the problems in (2.15), (2.16) and (2.18), whose objective functions are derived without taking multivariate Gaussian distributions into



(a) The ground truth GGL ($\mathbf{\Theta}^*$) (b) $\mathbf{S}^{-1}$: (RE,FS) = (0.171,0.20) (c) GLasso [30]: (RE,FS) = (0.078,0.64) (d) GGL: (RE,FS) = (0.054,0.89)

Figure 2.4: Illustration of precision matrices (whose entries are color coded) estimated from $\mathbf{S}$ for $k/n = 30$: In this example, (a) the ground truth is a GGL associated with a grid graph having $n=64$ vertices. From $\mathbf{S}$, the matrices are estimated by (b) inverting $\mathbf{S}$, (c) GLasso(α) and (d) GGL(α). The proposed method provides the best estimation in terms of RE and FS, and the resulting matrix is visually the most similar to the ground truth $\mathbf{\Theta}^*$.



(a) The ground truth CGL ($\mathbf{\Theta}^*$) (b) GLS [31]: (RE,FS) = (0.187,0.75) (c) GTI [34]: (RE,FS) = (0.322,0.39) (d) CGL: (RE,FS) = (0.085,0.82)

Figure 2.5: Illustration of precision matrices (whose entries are color coded) estimated from $\mathbf{S}$ for $k/n = 30$: In this example, (a) the ground truth is a CGL associated with a grid graph having $n=36$ vertices. From $\mathbf{S}$, the matrices are estimated by (b) GLS(α_1), (c) GTI and (d) CGL(α). The proposed method provides the best estimation in terms of RE and FS, and the resulting matrix is visually the most similar to the ground truth $\mathbf{\Theta}^*$.

account. Specifically, the objective in (2.16) serves as a lower-bound for the maximum-likelihood criterion in (2.24) (see Proposition 4). Besides, the performance difference against GTI is substantial [34] across different k/n, and GTI generally does not converge if $\mathbf{S}$ has low-rank (i.e., $k/n < 1$), so those results are not available.

2.5.2 Empirical Results for Computational Complexity

Figure 2.6 compares the computational speedups achieved by our proposed algorithms over GLasso, which is implemented according to the P-GLasso algorithm presented in [49]. In our experiments, the algorithms are tested on Erdos-Renyi graphs with $n = 64$ vertices ($\mathcal{G}_{\text{ER}}^{(64,p)}$). 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

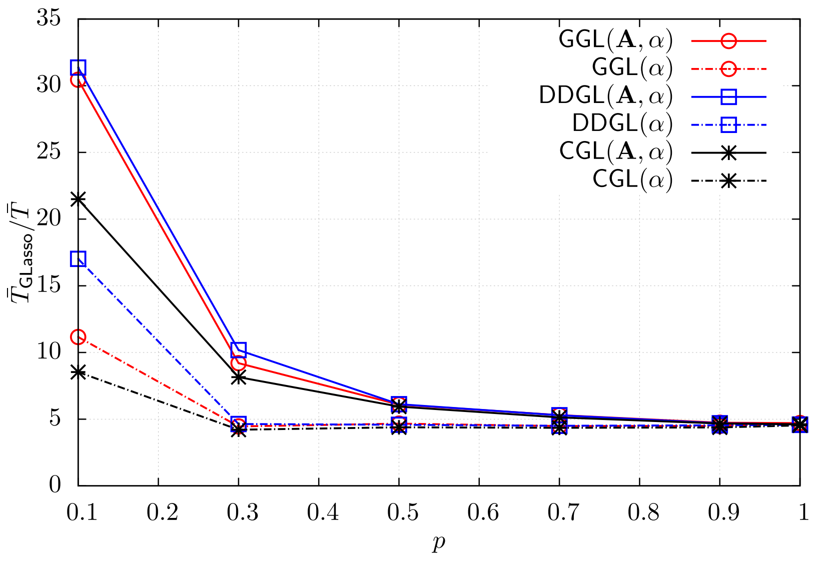


Figure 2.6: Computational speedup as compared to GLasso ($\bar{T}_{\mathsf{GLasso}}/\bar{T}$).

in the previous section. The speedup values are calculated using $\bar{T}_{\mathsf{GLasso}}/\bar{T}$, where $\bar{T}$ and $\bar{T}_{\mathsf{GLasso}}$ 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 [31, 32] on average, and the other methods [33, 34] do not have efficient implementations, we only use GLasso as the baseline algorithm in this experiment.

As shown in Figure 2.6, 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 ℓ_1-regularized quadratic program [30, 49] having a nonsmooth objective function, and it is generally harder to solve compared to the (smooth) nonnegative quadratic program in (2.46).

2.5.3 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* [74] 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 $\mathbf{S}$ is calculated as

$\mathbf{S} = (1/d)\,\mathbf{X}\mathbf{X}^{\mathsf{T}} + (1/3)\,\mathbf{I}$ where $\mathbf{X}$ is the mean removed $n \times d$ data matrix. The $(1/3)\,\mathbf{I}$ term is added based on the variational Bayesian approximation result in [42] for binary data. Then, $\mathbf{S}$ is used as input to the GLasso, GGL and CGL methods. Here, we only compare methods minimizing the objective function in (2.1) which is shown to be a suitable metric for binary data in [42, 43].

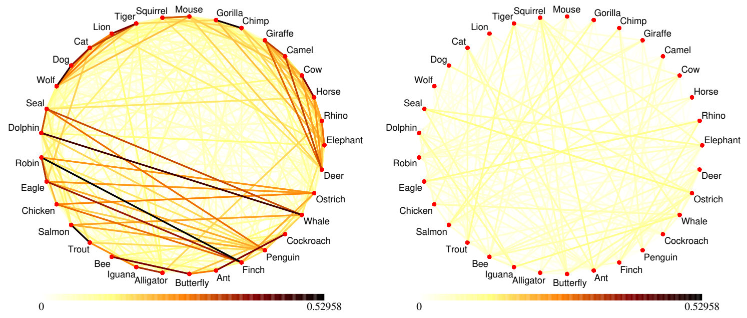
Figure 2.7 illustrates the graphs constructed by using GLasso, GGL and CGL with different regularization parameters. The positive and negative edge weights estimated by GLasso are shown side-by-side in Figures 2.7a and 2.7d, where the magnitudes of negative weights are substantially smaller than most 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 Figures 2.7a and 2.7d. The results follow the intuition that larger positive weights should be assigned between animals considered to be similar. Such pairs of animals include (gorilla,chimp), (dolphin,whale), (dog,wolf) and (robin,finch).

2.5.4 Graph Laplacian Estimation under Model Mismatch

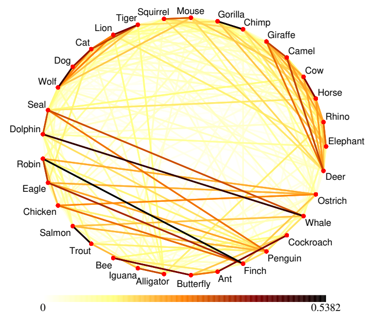
In the case of a model mismatch (i.e., when data is not generated by a GMRF with a graph Laplacian as its precision matrix), the proposed methods allow us to find the closest graph Laplacian fit with respect to the original model in the log-determinant Bregman divergence sense. Figure 2.8 illustrates two examples (simulating sparse and dense mismatches) where the ground truth precision matrices $\mathbf{\Omega}$ have both negative and positive edge weights (Figure 2.8a). From their true covariances $\mathbf{\Sigma} = \mathbf{\Omega}^{-1}$, GLasso recovers the original model by allowing both positive and negative edge weights (Figure 2.8b). On the other hand, the proposed GGL finds the closest graph Laplacian with respect to the ground truth (Figure 2.8c). 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 $\mathbf{\Omega}$. 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 [54].

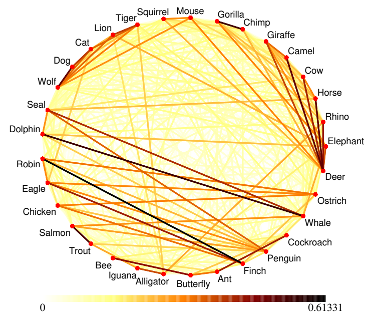
2.5.5 Graph Laplacian Estimation under Connectivity Mismatch

We now evaluate empirically the effect inaccurate selection of connectivity matrices ($\mathbf{A}$), used to determine the connectivity constraints in our problems. For this purpose, in our experiments, we randomly construct multiple 64×64 GGL matrices based on Erdos-Renyi graphs with $p = 0.1$ (i.e., $\mathcal{G}_{\mathrm{ER}}^{(64,0.1)}$) and generate datasets associated with the GGLs, as discussed in Section 2.5.1. Then, the proposed GGL method is employed to estimate graphs from generated data using different connectivity constraints, whose corresponding connectivity matrices are obtained by randomly swapping

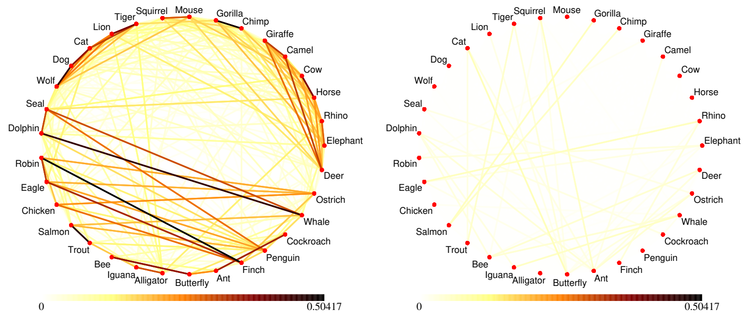


(a) $\mathsf{GLasso}(\alpha = 0)$: (Left) Positive and (Right) absolute negative weights

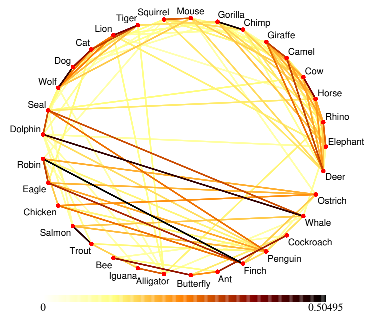


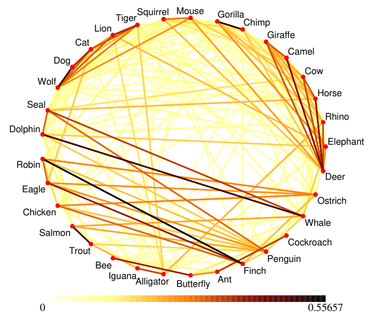


(b) $\mathsf{GGL}(\alpha = 0)$ (c) $\mathsf{CGL}(\alpha = 0)$



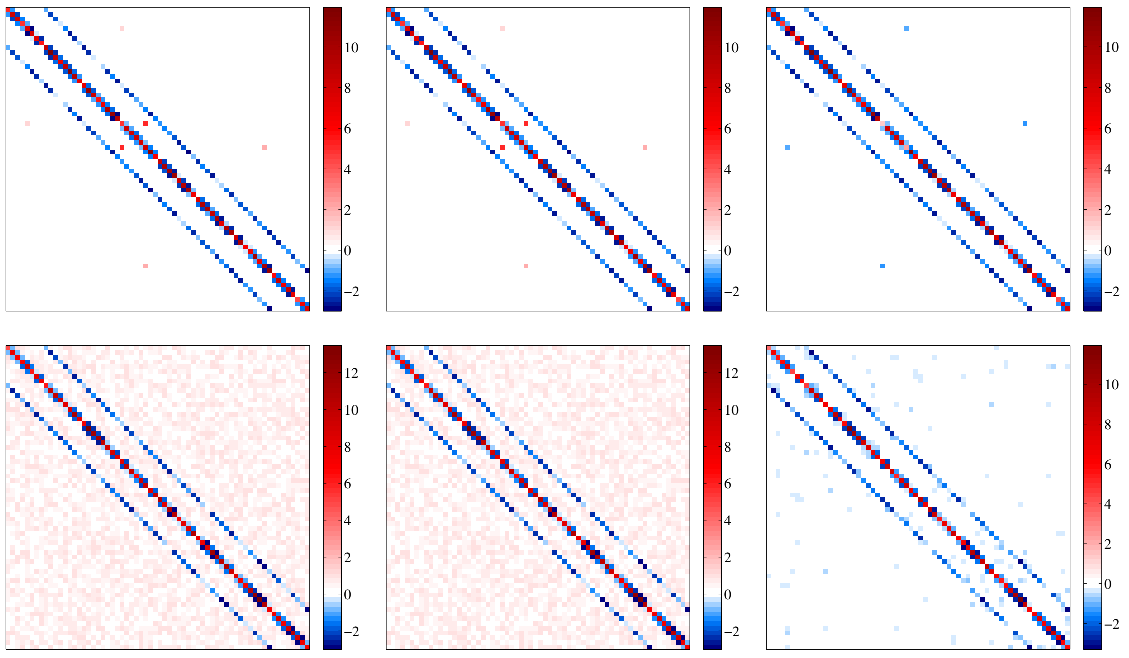
(d) $\mathsf{GLasso}(\alpha = 0.02)$: (Left) Positive and (Right) absolute negative weights





(e) $\mathsf{GGL}(\alpha = 0.02)$ (f) $\mathsf{CGL}(\alpha = 0.02)$

Figure 2.7: The graphs learned from Animals dataset using methods $\mathsf{GLasso}(\alpha)$, $\mathsf{GGL}(\alpha)$ and $\mathsf{CGL}(\alpha)$: $\mathsf{GGL}(\alpha)$ leads to sparser graphs compared to the others. The results follow the intuition that larger positive weights should be assigned between similar animals, although the dataset is categorical (non-Gaussian).



(a) Ground Truth ($\mathbf{\Omega}$) (b) $\mathsf{GLasso}(\alpha = 0)$ (c) $\mathsf{GGL}(\alpha = 0)$

Figure 2.8: Illustration of precision matrices whose entries are color coded, where negative (resp. positive) entries correspond to positive (resp. negative) edge weights of a graph: (a) Ground truth precision matrices ($\mathbf{\Omega}$) are randomly generated with (top) sparse and (bottom) dense positive entries (i.e., negative edge weights). From the corresponding true covariances ($\mathbf{\Sigma}$), the precision matrices are estimated by (b) $\mathsf{GLasso}(\alpha = 0)$ and (c) $\mathsf{GGL}(\alpha = 0)$ methods without ℓ_1-regularization.

the entries of the true connectivity matrix to simulate connectivity mismatches. Specifically, a 5% mismatch is obtained by randomly exchanging 5% of the ones in $\mathbf{A}$ with zero entries.

Figure 2.9 compares the accuracy of GGL estimation with connectivity constraints under different levels (5%, 10%, 15% and 25%) of connectivity mismatch against the GGL estimation with true connectivity constraints (i.e., $\mathsf{GGL}(\mathbf{A})$) and without using any connectivity constraints (i.e., $\mathsf{GGL}(\mathbf{A}_{\text{full}})$). 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 $\mathsf{GGL}(\mathbf{A})$ and can outperform $\mathsf{GGL}(\mathbf{A}_{\text{full}})$, even though there is a 25% mismatch ($\mathsf{GGL}(\mathbf{A}_{25\%})$). However, the performance difference with respect to $\mathsf{GGL}(\mathbf{A})$ increases as the number of data samples increases (e.g., $k/n > 10$), where both $\mathsf{GGL}(\mathbf{A})$ and $\mathsf{GGL}(\mathbf{A}_{\text{full}})$ performs substantially better. In this case, the graph estimation without connectivity constraints ($\mathsf{GGL}(\mathbf{A}_{\text{full}})$) can be preferred if connectivity information is uncertain.

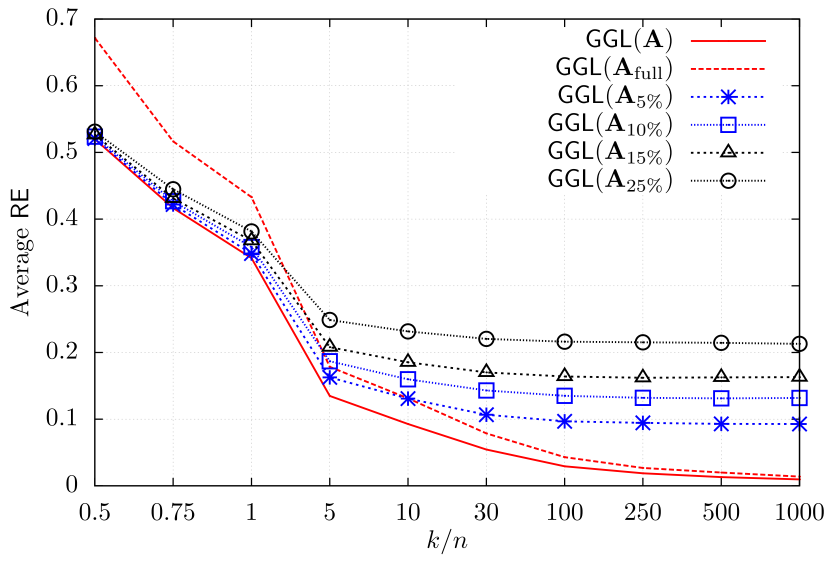


Figure 2.9: Average relative errors for different number of data samples (k/n), where $\mathsf{GGL}(\mathbf{A})$ exploits the true connectivity information in GGL estimation, and $\mathsf{GGL}(\mathbf{A}_{\text{full}})$ refers to the GGL estimation without connectivity constraints. Different levels of connectivity mismatch are tested. For example, $\mathsf{GGL}(\mathbf{A}_{5\%})$ corresponds to the GGL estimation with connectivity constraints having a 5% mismatch. No ℓ_1-regularization is applied in GGL estimation (i.e., $\alpha = 0$).

2.6 Conclusion

In this chapter, 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.

Chapter 3

Graph Learning from Filtered Signals: Graph System Identification

In Chapter 2, we have proposed algorithms for learning three different types of Laplacian matrices, where models of interest are defined by graph Laplacians only. The present chapter extends Chapter 2 by (i) introducing more general graph-based models based on *graph systems*, defined by a graph (i.e., graph Laplacian) and a graph-based filter (i.e., a matrix function of a graph Laplacian) as illustrated in Figure 3.1, and (ii) proposing an algorithm to learn parameters of a graph system from multiple signal/data observations.

For graph-based modeling, graph systems provide a general abstraction where graphs represent pairwise relations between the samples, and graph-based filters (GBFs)[a] determine the signal smoothness. With this construction, we formulate the graph-based modeling as a *graph system identification (GSI)* problem with an ℓ_1-regularized maximum likelihood (ML) criterion, where the goal is to jointly identify a graph Laplacian $\mathbf{L}$ and a GBF h from signals/data. In order to solve the GSI problem, we propose an alternating optimization algorithm that first optimizes the graph by fixing the GBF, and then designs the GBF by fixing the graph. The proposed algorithm involves three main steps:

- *Graph-based filter (GBF) identification*: For a given graph Laplacian $\mathbf{L}$, the purpose of this step is to design a GBF h, whose inverse function (i.e., h^{-1}) is used for the prefiltering step discussed next.

A version of the work presented in this chapter will be submitted for publication subsequently [75].

[a]A formal definition for graph-based filters is given in Section 1.3 (see Definition 6).

- *Prefiltering*: We perform an inverse filtering operation (h^{-1}) on observed signals, then the prefiltered signals are used in graph Laplacian estimation. We show that this step of our algorithm significantly improves the accuracy of graph estimation.

- *Graph Laplacian Estimation*: To learn graphs from prefiltered signals, we employ the CGL estimation algorithm introduced in Chapter 2. Although the present chapter focuses on graphs associated with CGL matrices, the algorithm can be simply extended for different types of graph Laplacian matrices.

The proposed algorithm is developed under the general assumption that h is a one-to-one function, hence its inverse function h^{-1} exists. Based on this condition, we specifically consider the parametric GBFs listed in Table 3.1, which are one-to-one functions and have useful applications discussed in Section 3.2.3. The first three GBF types in the table define basic scaling and shifting operations, and the exponential decay and β-hop localized GBFs provide diffusion-based models. These GBFs can also be used for modeling different classes of smooth graph signals satisfying that most of the signal energy is concentrated in the low graph frequencies[a], since they yield larger filter responses (h_β) in lower graph frequencies (λ) as illustrated in Figure 3.2.

In order to accommodate the GBFs in Table 3.1 into our algorithm (i.e., the GBF identification step), we propose specific methods to find the filter parameter β that fully characterizes the GBF. For a selected GBF type, our proposed algorithm guarantees optimal identification of β and $\mathbf{L}$ in an ℓ_1-regularized ML sense for frequency shifting, variance shifting and β-hop localized GBFs. However, for frequency scaling and exponential decay GBFs, our algorithm cannot find the optimal β in general, but it guarantees that the estimated $\mathbf{L}$ is optimal up to a constant factor. In practice, the type of GBF and its parameter β can also be selected based on the prior knowledge available about the problem and the application. In this case, different GBFs and their parameters can be tested until the estimated graph and GBF pair achieves the desired performance (e.g., in terms of mean square error, likelihood or sparsity) where the parameter β serves as a regularization parameter, which can be used for tuning smoothness of the signal for example.

Moreover, our algorithm can be extended to support different types of GBFs with more than one filter parameter by introducing specific methods to identify their parameters. As long as a specified GBF (h) has an inverse function (h^{-1}), the prefiltering and graph Laplacian estimation steps can be directly utilized to learn graphs from signals/data.

This chapter is organized as follows. Section 3.1 presents the related work. We formulate the graph system identification problem and discuss its variations in Section 3.2. In Section 3.3, we derive optimality conditions and develop methods for the graph system identification. Experimental results are presented in Section 3.4 and Section 3.5 draws some conclusions.

[a]Graph signals satisfying this condition are analogous to low-pass signals in classical signal processing.

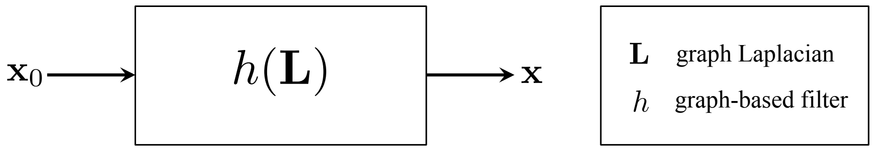
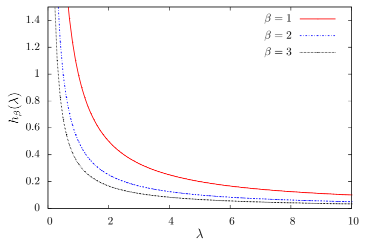
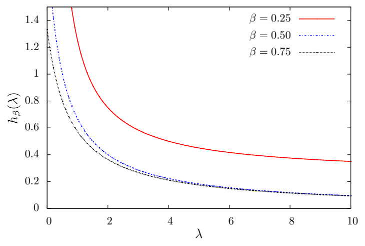


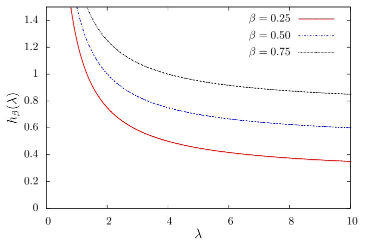
Figure 3.1: Input-output relation of a graph system defined by a graph Laplacian ($\mathbf{L}$) and a graph-based filter (h). In this work, we focus on joint identification of $\mathbf{L}$ and h.

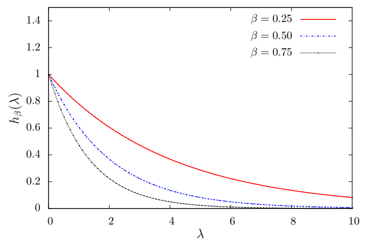
Table 3.1: Graph-based filter (GBF) functions with parameter β and corresponding inverse functions. Note that $\lambda^\dagger$ denotes the pseudoinverse of λ, that is, $\lambda^\dagger = 1/\lambda$ for $\lambda \neq 0$ and $\lambda^\dagger = 0$ for $\lambda = 0$.

Filter Name	$h_\beta(\lambda)$	$h_\beta^{-1}(s)$
Frequency scaling	$\begin{cases} \frac{1}{\beta\lambda} & \lambda > 0 \\ 0 & \lambda = 0 \end{cases}$	$\begin{cases} \frac{1}{\beta s} & s > 0 \\ 0 & s = 0 \end{cases}$
Frequency shifting	$(\lambda + \beta)^\dagger$	$\frac{1}{s} - \beta$
Variance shifting	$\lambda^\dagger + \beta$	$(s - \beta)^\dagger$
Exponential decay	$\exp(-\beta\lambda)$	$-\log(s)/\beta$
β-hop localized	$\begin{cases} \frac{1}{\lambda^\beta} & \lambda > 0 \\ 0 & \lambda = 0 \end{cases}$	$\begin{cases} \left(\frac{1}{s}\right)^{\frac{1}{\beta}} & s > 0 \\ 0 & s = 0 \end{cases}$


(a) Frequency scaling GBF


(b) Frequency shifting GBF


(c) Variance shifting GBF


(d) Exponential decay GBF

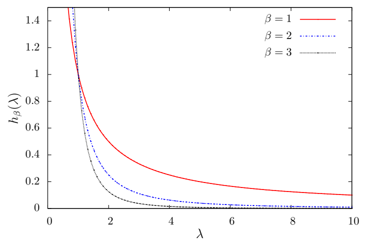

(e) β-hop localized GBF

Figure 3.2: Graph-based filters with different β parameters as a function of graph frequency $h_\beta(\lambda)$.

3.1 Related Work and Contributions

Some of the related studies have already discussed in the context of graph Laplacian estimation (in Chapter 2). In this section, we revisit them by pointing out their main differences with respect to our work in the present chapter. Table 3.2 summarizes the related studies by comparing against our work in terms of target variables, underlying assumptions and optimization criteria.

Table 3.2: An overview of the related work on learning graphs from smooth/diffused signals (NA: No assumptions, L: Localized, 1-1: one-to-one function).

References	Target Variable(s)	Assumptions			Optimization Criterion
		Graph	Filter	Signal	
[31, 32]	graph Laplacian	NA	NA	NA	regularized Laplacian quadratic form
[34]	shift operator	NA	NA	NA	ℓ_1-norm of shift operator
[56]	adjacency matrix	NA	NA	NA	trace of adjacency
[76]	GBF source locations	Given	NA	L	least squares
[77]	multiple heat kernels source locations	NA	Heat kernel	L	regularized least squares
[78]	polynomials of adjacency	Sparse	Poly-nomial	NA	least squares
This work	graph Laplacian GBF (Table 3.1)	NA	1-1	NA	regularized maximum likelihood

As discussed in Chapter 2, the problems in (2.15) and (2.16) are solved to estimate CGL matrices from smooth signals [31, 32], yet GBFs are not considered in their formulations. In [34, 56], the authors focus on learning graph shift/diffusion operators (such as adjacency and graph Laplacian matrices) from a complete set of eigenvectors that has to be computed from observed data. Particularly, Segarra *et al.* [34] solve the sparse recovery problem in (2.18) to infer the topology of a graph, and Pasdeloup *et al.* [56] focus on the estimation of an adjacency matrix by solving a trace minimization problem. In fact, the problems in [34] and [56] are equivalent if the target matrix is constrained to be a CGL matrix, since the problems of minimizing $\|\mathbf{L}\|_1$ and $\mathrm{Tr}(\mathbf{L})$ over the set of CGL matrices (i.e., $\mathbf{L} \in \mathcal{L}_c$) lead to the same solution. Although our approach also requires a set eigenvectors to be computed, they are not directly used to estimate a graph. Instead, the computed eigenvectors are used for the prefiltering step, then a graph is estimated from the prefiltered signals by minimizing a regularized ML criterion. Segarra *et al.* [76] discuss the joint identification of a GBF and a sparse input (localized sources) from a set of observed signals for a given graph. Thanou *et al.* [77] further address the estimation of a graph in a similar setting with localized sources and propose a dictionary-based method where a graph estimation problem is solved to construct a dictionary consisting of multiple diffusion kernels, and the resulting dictionary is used for identifying the localized sources in the diffusion process. Although the present chapter focuses on the GSI problem without locality assumptions on diffused signals, when a single diffusion kernel is used in the dictionary and no locality assumptions are imposed, the problem in [77] becomes analogous to our problem of interest and specifically reduces to the CGL estimation problem in (2.15) originally proposed by Dong *et al.* [31]. Yet, in our algorithm, we solve a different problem with a regularized ML criterion (i.e., Problem 3 introduced in Chapter 2) to estimate graphs from prefiltered signals. The proposed algorithm can be used as an alternative method to construct a dictionary as in [77] for identifying localized sources, which is out of the scope of the present work. Moreover, Mei and Moura [78] address the estimation of polynomials of adjacency matrices by solving a regularized least-squares problem. As their counterparts, polynomials of graph Laplacians can be used to approximate the GBFs in Table 3.1 as well as many other types filters such as bandlimited GBFs [57]. Since polynomial filters provide more degrees of freedom in designing GBFs, they can be considered as more general compared to our GBFs of interest. However, they are not one-to-one functions in general, so the proposed algorithm (i.e., the prefiltering step) cannot be applied to polynomial filters. Additionally, the algorithm proposed in [78] can only guarantee the optimality of solutions in mean-square sense under a very restrictive set of assumptions which require the polynomials of adjacency matrices to be sparse[a]. Our proposed algorithm provides stronger optimality guarantees in a regularized ML sense without imposing any assumptions on the sparsity of graphs, yet the GBFs of interest are more restrictive than polynomial filters. The identification of graph systems with polynomial filters will be studied as part of our future work.

To the best of our knowledge, this is the first work that (i) formulates the graph-based modeling problem as identification of graph systems (with the types of GBFs in Table 3.1) under a regularized ML criterion and (ii) proposes an algorithm based on prefiltering to jointly identify a graph and a GBF. The existing related studies consider optimization of different criteria (see Table 3.2), and none of them propose a prefiltering-based procedure for graph estimation. Since diffusion kernels [15] are special cases of graph systems, which are equivalent to the graph systems with an exponential decay filter, our proposed method can be used to learn diffusion kernels from signals/data, which are popular in many applications [21, 17, 79].

3.2 Problem Formulation: Graph System Identification

Our formulation is based on the following general assumption on a GBF h, which holds for the GBFs in Table 3.1.

[a]In practice, powers of adjacency matrices lead to dense matrices.

Assumption 1. We assume that a graph-based filter $h(\lambda)$ is a nonnegative and one-to-one function of λ.

3.2.1 Filtered Signal Model

We formulate the graph system identification problem in a probabilistic setting by assuming that the observed (filtered) signals have been sampled from a zero mean n-variate Gaussian distribution $\mathbf{x} \sim \mathsf{N}(\mathbf{0}, \mathbf{\Sigma} = h(\mathbf{L}))$,

$$p(\mathbf{x}|h(\mathbf{L})) = \frac{1}{(2\pi)^{n/2}|h(\mathbf{L})|^{1/2}} \exp\left(-\frac{1}{2}\mathbf{x}^{\mathsf{T}}h(\mathbf{L})^{\dagger}\mathbf{x}\right), \tag{3.1}$$

with the covariance $\mathbf{\Sigma} = h(\mathbf{L})$ defined based on a graph Laplacian $\mathbf{L}$ and a GBF h. The signal model in Chapter 2 is a special case of (3.1), which can be shown by choosing the GBF h in (3.1) as

$$h(\lambda) = \lambda^{\dagger} = \begin{cases} 1/\lambda & \lambda > 0 \\ 0 & \lambda = 0 \end{cases} \text{ that is } h(\mathbf{L}) = \mathbf{L}^{\dagger}. \tag{3.2}$$

Then, replacing the $h(\mathbf{L})$ term in (3.1) with $\mathbf{L}^{\dagger}$ leads to the GMRF model in (2.19), whose precision (inverse covariance) is $\mathbf{\Omega} = h(\mathbf{L})^{\dagger} = \mathbf{L}$. Obviously, for a general GBF, $h(\mathbf{L})^{\dagger}$ does not always have a graph Laplacian form.

From the graph signal processing perspective, the random signal $\mathbf{x}$ in (3.1) can be considered as the output of a graph system (see in Figure 3.1 and Definition 7) defined by a graph Laplacian matrix $\mathbf{L}$ and a GBF $\bar{h}(\mathbf{L})$ for the input $\mathbf{x}_0 \sim \mathsf{N}(\mathbf{0}, \mathbf{\Sigma}_0)$. Without loss of generality, we assume that the input is a white Gaussian noise $\mathbf{x}_0 \sim \mathsf{N}(\mathbf{0}, \mathbf{I})$ and choose $\bar{h}(\mathbf{L}) = \sqrt{h(\mathbf{L})}$, so the covariance of the output $\mathbf{x} = \bar{h}(\mathbf{L})\mathbf{x}_0$ is

$$\mathbf{\Sigma} = \mathsf{E}[\mathbf{x}\mathbf{x}^{\mathsf{T}}] = \sqrt{h(\mathbf{L})}\mathsf{E}[\mathbf{x}_0\mathbf{x}_0^{\mathsf{T}}]\sqrt{h(\mathbf{L})}^{\mathsf{T}} = \sqrt{h(\mathbf{L})}\sqrt{h(\mathbf{L})}^{\mathsf{T}} = h(\mathbf{L}) \tag{3.3}$$

as in (3.1), since $\mathsf{E}[\mathbf{x}_0\mathbf{x}_0^{\mathsf{T}}] = \mathbf{I}$. Note that for a more general (colored) input model, $\mathbf{x}_0 \sim \mathsf{N}(\mathbf{0}, \mathbf{\Sigma}_0)$, the whitening transform $\mathbf{T} = \mathbf{\Sigma}_0^{-1/2}$ can be used to write $\mathbf{\Sigma}$ as

$$\mathbf{\Sigma} = \mathsf{E}[\mathbf{x}\mathbf{x}^{\mathsf{T}}] = \sqrt{h(\mathbf{L})}\mathbf{T}\mathsf{E}[\mathbf{x}_0\mathbf{x}_0^{\mathsf{T}}]\mathbf{T}\sqrt{h(\mathbf{L})}^{\mathsf{T}} = h(\mathbf{L}). \tag{3.4}$$

Moreover, for modeling smooth graph signals, it is reasonable to choose $h(\lambda)$ to be a monotonically decreasing function[a] satisfying $h(\lambda_1) \geq h(\lambda_2) \geq \cdots \geq h(\lambda_n) > 0$, where the graph frequencies (i.e., eigenvalues of $\mathbf{L}$) are ordered as $0 = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$. Thus, the corresponding covariance matrix $\mathbf{\Sigma} = h(\mathbf{L})$ represent graph signals whose energy is larger in lower graph frequencies. On the other

[a]All of the GBFs in Table 3.1 are monotonically decreasing functions on the interval $\lambda \in (0, \infty)$. The exponential decay and frequency shifting GBFs further satisfies $h(\lambda_1) \geq h(\lambda_2)$ at the zero frequency (i.e., $0 = \lambda_1 \leq \lambda_2$), while for the other GBFs, we have $h(\lambda_2) \geq h(\lambda_1)$.

hand, the inverse covariance $h(\mathbf{L})^\dagger$ leads to a variation operator that is similar to a graph Laplacian[a] $\mathbf{L}$, which is essentially a special case of $h(\mathbf{L})^\dagger$ if the GBF is chosen as in (3.2). The quadratic term $\mathbf{x}^\intercal h(\mathbf{L})^\dagger\mathbf{x}$ in the exponent of (3.1) can be compactly written as

$$\mathbf{x}^\intercal h(\mathbf{L})^\dagger\mathbf{x} = \mathbf{x}^\intercal\mathbf{U}h(\mathbf{\Lambda}_\lambda)^\dagger\mathbf{U}^\intercal\mathbf{x} = \mathbf{x}^\intercal\mathbf{U}\mathbf{\Lambda}_\sigma^\dagger\mathbf{U}^\intercal\mathbf{x} \tag{3.5}$$

by using the GBT $\mathbf{U}$, which jointly diagonalizes $\mathbf{L} = \mathbf{U}\mathbf{\Lambda}_\lambda\mathbf{U}^\intercal$ and $h(\mathbf{L}) = \mathbf{U}h(\mathbf{\Lambda}_\lambda)\mathbf{U}^\intercal = \mathbf{U}\mathbf{\Lambda}_\sigma\mathbf{U}^\intercal$, where the diagonal matrices $\mathbf{\Lambda}_\lambda = \operatorname{diag}([\lambda_1\,\lambda_2\cdots\lambda_n]^\intercal)$ and $\mathbf{\Lambda}_\sigma = \operatorname{diag}([\sigma_1^2\,\sigma_2^2\cdots\sigma_n^2]^\intercal)$ are composed of the graph frequencies and GBF responses, respectively. Note that the nonnegativity condition in Assumption 1, that is, $\sigma_i^2 = h(\lambda_i) \geq 0$ for $i = 1,\dots,n$, ensures that the covariance $\mathbf{\Sigma} = h(\mathbf{L})$ is a positive semidefinite matrix. In terms of the GBF responses (i.e., $h(\lambda_i) = \sigma_i^2$ for $i = 1,2,\dots,n$), we can rewrite (3.5) as

$$\mathbf{x}^\intercal h(\mathbf{L})^\dagger\mathbf{x} = \mathbf{x}^\intercal\mathbf{U}h(\mathbf{\Lambda}_\lambda)^\dagger\mathbf{U}^\intercal\mathbf{x} = \sum_{i=1}^{n}\frac{1}{h(\lambda_i)}\widehat{x}_i^2 = \sum_{i=1}^{n}\frac{1}{\sigma_i^2}\widehat{x}_i^2. \tag{3.6}$$

where $\widehat{x}_i = \mathbf{u}_i^\intercal\mathbf{x}$ and $\mathbf{u}_i$ denotes the i-th column of $\mathbf{U}$ associated with the graph frequency λ_i. Since the ordering of the inverse GBF responses, $1/\sigma_1^2 \leq 1/\sigma_2^2 \leq \cdots \leq 1/\sigma_n^2$, is consistent with the ordering of graph frequencies $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$, $\mathbf{x}^\intercal h(\mathbf{L})^\dagger\mathbf{x}$ provides a variation measure such that a smaller $\mathbf{x}^\intercal h(\mathbf{L})^\dagger\mathbf{x}$ means a smoother signal $\mathbf{x}$. As a special case, choosing $h(\lambda) = \lambda^\dagger$ reduces (3.6) to the quadratic graph Laplacian form in (1.6).

3.2.2 Problem Formulation

Our goal is to estimate the graph system parameters $\mathbf{L}$ and h based on k random samples, $\mathbf{x}_i$ for $i = 1,\dots,k$, obtained from the signal model $p(\mathbf{x}|\mathbf{\Sigma})$ in (3.1) by maximizing the likelihood of $\mathbf{\Sigma} = h(\mathbf{L})$, that is

$$\prod_{i=1}^{k} p(\mathbf{x}_i|\mathbf{\Sigma}) = (2\pi)^{-\frac{kn}{2}}|\mathbf{\Sigma}|^{-\frac{k}{2}}\prod_{i=1}^{k}\exp\left(-\frac{1}{2}\mathbf{x}_i{}^\intercal\mathbf{\Sigma}^\dagger\mathbf{x}_i\right). \tag{3.7}$$

The maximization of (3.7) can be equivalently stated as minimizing its negative log-likelihood, which leads to the following problem for estimating the graph system parameters $\mathbf{L}$ and h,

$$\begin{aligned}(\widehat{\mathbf{L}},\widehat{h}) &= \underset{\mathbf{L},h}{\operatorname{argmin}}\left\{\frac{1}{2}\sum_{i=1}^{k}\operatorname{Tr}\left(\mathbf{x}_i{}^\intercal h(\mathbf{L})^\dagger\mathbf{x}_i\right) - \frac{k}{2}\log|h(\mathbf{L})^\dagger|\right\} \\ &= \underset{\mathbf{L},h}{\operatorname{argmin}}\left\{\operatorname{Tr}\left(h(\mathbf{L})^\dagger\mathbf{S}\right) - \log|h(\mathbf{L})^\dagger|\right\}\end{aligned} \tag{3.8}$$

[a] In Chapter 1, the graph Laplacian quadratic form $\mathbf{x}^\intercal\mathbf{L}\mathbf{x}$ in (1.6) is used to quantify the variation of a signal $\mathbf{x}$ on the graph associated with $\mathbf{L}$.

where $\mathbf{S}$ is the sample covariance calculated using k samples, $\mathbf{x}_i$ for $i = 1, 2, \dots, k$. In our problem formulation, we additionally introduce a sparsity promoting weighted ℓ_1-regularization, $\|\mathbf{L} \odot \mathbf{H}\|_1$ for robust graph estimation, where $\mathbf{H}$ is symmetric regularization matrix and $\odot$ denotes element-wise multiplication. Thus, the proposed graph system identification problem is formulated as follows:

$$\begin{aligned} \underset{\mathbf{L}\succeq 0,\beta}{\text{minimize}} \quad & \mathrm{Tr}(h_\beta(\mathbf{L})^\dagger \mathbf{S}) - \log|h_\beta(\mathbf{L})^\dagger| + \|\mathbf{L} \odot \mathbf{H}\|_1 \\ \text{subject to} \quad & \mathbf{L}\,\mathbf{1} = \mathbf{0}, \quad (\mathbf{L})_{ij} \leq 0 \quad i \neq j \end{aligned} \tag{3.9}$$

where β is the parameter for a given type of filter h_β from Table 3.1, and $\mathbf{H}$ denotes the regularization matrix. The constraints ensure that $\mathbf{L}$ is a CGL matrix[a]. In this work, we particularly choose $\mathbf{H} = \alpha(2\mathbf{I} - \mathbf{1}\mathbf{1}^\intercal)$ so that the regularization term in (3.9) reduces to the standard ℓ_1-regularization $\alpha\|\mathbf{L}\|_1$ with parameter α.

3.2.3 Special Cases of Graph System Identification Problem and Applications of Graph-based Filters

Graph Learning. The problem in (3.9) can be reduced to the CGL problem proposed in Chapter 2 by choosing the filter h_β to be the frequency scaling or β-hop localized GBFs with $\beta = 1$, so that we obtain $h_\beta(\lambda) = \lambda^\dagger$ and $h_\beta(\mathbf{L}) = \mathbf{L}^\dagger$ as in (3.2). Since $h_\beta(\mathbf{L})^\dagger = \mathbf{L}$, we can rewrite the objective function in (3.9) as

$$\mathrm{Tr}(\mathbf{L}\mathbf{S}) - \log|\mathbf{L}| + \|\mathbf{L} \odot \mathbf{H}\|_1 \tag{3.10}$$

which is the objective function of graph learning problems discussed in Chapter 2.

Graph Learning from Noisy Signals. The variance shifting filter corresponds to a noisy signal model with variance $\beta = \sigma^2$. Specifically, assuming that the noisy signal is modeled as $\widehat{\mathbf{x}} = \mathbf{x} + \mathbf{e}$, where $\mathbf{x} \sim \mathsf{N}(\mathbf{0}, \mathbf{L}^\dagger)$ denotes the original signal, and $\mathbf{e} \sim \mathsf{N}(\mathbf{0}, \sigma^2\mathbf{I})$ is the additive white Gaussian noise vector independent of $\mathbf{x}$ with variance σ^2. Therefore, the noisy signal $\widehat{\mathbf{x}}$ is distributed as $\widehat{\mathbf{x}} \sim \mathsf{N}(\mathbf{0}, \widehat{\mathbf{\Sigma}} = \mathbf{L}^\dagger + \sigma^2\mathbf{I})$. The same model is obtained by variance shifting filter with $\beta = \sigma^2$ so that

$$h_\beta(\lambda) = \lambda^\dagger + \beta \text{ and } h_\beta(\mathbf{L}) = \mathbf{L}^\dagger + \beta\mathbf{I} = \mathbf{L}^\dagger + \sigma^2\mathbf{I}. \tag{3.11}$$

Graph Learning from Frequency Shifted Signals. For the shifted frequency filter with parameter β, we have

$$h_\beta(\lambda) = (\lambda + \beta)^\dagger \text{ and } h_\beta(\mathbf{L}) = (\mathbf{L} + \beta\mathbf{I})^\dagger. \tag{3.12}$$

[a]The formulation can be trivially extended for different types of graph Laplacians (e.g., generalized graph Laplacian) discussed in Chapter 2.

By substituting $h_\beta(\mathbf{L})$ with $(\mathbf{L}+\beta\mathbf{I})^\dagger$, the problem in (3.9) can be written as

$$\begin{array}{ll} \underset{\widetilde{\mathbf{L}}\succeq 0,\beta\geq 0}{\text{minimize}} & \operatorname{Tr}(\widetilde{\mathbf{L}}\mathbf{S})-\log|\widetilde{\mathbf{L}}|+\|\widetilde{\mathbf{L}}\odot\mathbf{H}\|_1 \\ \text{subject to} & \widetilde{\mathbf{L}}=\mathbf{L}+\beta\mathbf{I}, \quad \mathbf{L}\mathbf{1}=\mathbf{0}, \quad (\mathbf{L})_{ij}\leq 0 \;\; i\neq j \end{array} \tag{3.13}$$

which is the shifted combinatorial Laplacian (SCGL) estimation problem in (2.13) that is originally proposed in [33].

Diffusion Kernel Learning. The diffusion kernel on a graph $\mathcal{G}$ [15] is the matrix exponential of $\mathbf{L}$, that is

$$\exp(-\beta\mathbf{L}) = \lim_{t\to\infty}\left(\mathbf{I}-\frac{\beta\mathbf{L}}{t}\right)^t \quad t\in\mathbb{N}, \tag{3.14}$$

where $\mathbf{L}$ is the graph Laplacian associated with $\mathcal{G}$ and the parameter β is a real number called diffusion bandwidth.

To learn diffusion kernels, defined in (3.14), our proposed problem in (3.9) can be modified by choosing $h_\beta(\lambda)$ as the exponential decay filter, so we get

$$h_\beta(\mathbf{L})=\mathbf{U}h_\beta(\mathbf{\Lambda})\mathbf{U}^\intercal=\mathbf{U}\exp(-\beta\mathbf{\Lambda})\mathbf{U}^\intercal=\exp(-\beta\mathbf{L}) \tag{3.15}$$

based on Definitions 5 and 6.

Diffusion kernels are used to model a basic class of random diffusion processes [15]. Suppose that the random process is obtained by diffusion of the initial random vector $\mathbf{x}(0)$ whose entries are independent, zero-mean random variables, denoted as $(\mathbf{x}(0))_i$ for $i=1,2,\dots,n$, with variance σ^2, the random diffusion over a graph $\mathcal{G}$ is formulated recursively as

$$\mathbf{x}(t+1)=\mathbf{x}(t)-r\mathbf{L}\mathbf{x}(t)=(\mathbf{I}-r\mathbf{L})\mathbf{x}(t) \quad t\in\{0,1,\dots\} \tag{3.16}$$

where t represents the time index, $\mathbf{L}$ is the graph Laplacian of $\mathcal{G}$ and the real number $r\in(0,1)$ is the diffusion rate of the process. On i-th vertex of the graph $\mathcal{G}$, we can write the random process in (3.16) as

$$(\mathbf{x}(t+1))_i=(\mathbf{x}(t))_i+r\sum_{j\in\mathcal{N}_i}(\mathbf{W})_{ij}\left((\mathbf{x}(t))_j-(\mathbf{x}(t))_i\right) \tag{3.17}$$

for all i where $\mathbf{W}$ is the adjacency matrix of the graph $\mathcal{G}$, and $\mathcal{N}_i$ is the set of vertex indices neighboring i-th vertex (v_i). More compactly, (3.16) can be written as

$$\mathbf{x}(t)=\mathbf{G}(t)\mathbf{x}(0) \text{ where } \mathbf{G}(t)=(\mathbf{I}-r\mathbf{L})^t\,. \tag{3.18}$$

The covariance of $\mathbf{x}(t)$ is

$$\begin{aligned}(\boldsymbol{\Sigma}(t))_{ij} &= \mathsf{E}\left[(\mathbf{G}(t)\mathbf{x}(0))_i(\mathbf{G}(t)\mathbf{x}(0))_j\right] \\ &= \mathsf{E}\left[\left(\sum_{k=1}^{n}(\mathbf{G}(t))_{ik}(\mathbf{x}(0))_k\right)\left(\sum_{l=1}^{n}(\mathbf{G}(t))_{jl}(\mathbf{x}(0))_l\right)\right],\end{aligned} \tag{3.19}$$

which simplifies by using independence of $\mathbf{x}(0)$ as

$$(\boldsymbol{\Sigma}(t))_{ij} = \sigma^2\sum_{k=1}^{n}((\mathbf{G}(t))_{ik}(\mathbf{G}(t))_{kj}) = \sigma^2(\mathbf{G}(t)^2)_{ij}. \tag{3.20}$$

Therefore, the covariance matrix leads to

$$\boldsymbol{\Sigma}(t) = \mathbf{G}(t)^2 = \sigma^2\left(\mathbf{I}-r\mathbf{L}\right)^{2t}. \tag{3.21}$$

To adjust the time resolution of the process, we replace t with $t/\Delta t$ and r with $r\Delta t$ in $\mathbf{G}(t)$ so that

$$\mathbf{G}(t) = \sigma\left(\mathbf{I}-\frac{r\mathbf{L}}{1/\Delta t}\right)^{t/\Delta t}. \tag{3.22}$$

For arbitrarily small Δt, $\mathbf{G}(t)$ converges to a matrix exponential function of $\mathbf{L}$,

$$\widetilde{\boldsymbol{\Sigma}}(t) = \sigma^2\lim_{\Delta t\to 0}\left(\mathbf{I}-\frac{r\mathbf{L}}{1/\Delta t}\right)^{2t/\Delta t} = \sigma^2\exp(-2rt\mathbf{L}) \tag{3.23}$$

which is equivalent to exponential decay filtering operation in Table 3.1 with $\beta(t) = -2rt$. For arbitrarily large t (i.e., when the diffusion reaches steady-state), the covariance is

$$\lim_{t\to\infty}\widetilde{\boldsymbol{\Sigma}}(t) = \lim_{t\to\infty}\sum_{i=1}^{n}\exp(-2rt\lambda_i)\mathbf{u}_i\mathbf{u}_i^{\mathsf{T}} = \frac{\sigma^2}{n}\mathbf{1}\mathbf{1}^{\mathsf{T}}, \tag{3.24}$$

whose all entries have the same value, since we have $\lambda_1 = 0$ for CGLs. Thus, the statistical properties eventually become spatially flat across all vertices as intuitively expected for a diffusion process. Yet, for a nonsingular $\mathbf{L}$ (e.g., a GGL), the process bahaves differently so that the signal energy vanishes, since the above limit converges to the $n\times n$ zero matrix as t gets larger.

Graph Learning from β-hop Localized Signals. To learn graphs from β-hop localized signals, where β is a positive integer, the GBF can be selected as $h(\lambda) = (\lambda^{\dagger})^{\beta}$ so that the corresponding inverse covariance (precision) matrix in (3.9) is $h(\mathbf{L})^{\dagger} = \mathbf{L}^{\beta}$, which is generally not a graph Laplacian matrix due to the exponent β. However, it defines a β-hop localized operation on a graph associated with $\mathbf{L}$, and the corresponding signal model leads to β-hop localized signals, in which each sample depends on samples located within the neighborhood at most β-hops away.

Algorithm 3 Graph System Identification

```
Input: Sample covariance $\mathbf{S}$, graph-based filter type $h_\beta$
Output: Graph Laplacian $\mathbf{L}$ and $\beta$ filter parameter
1: Obtain $\mathbf{U}$ and $\mathbf{\Lambda}_s$ via eigendecomposition $\mathbf{S}=\mathbf{U}\mathbf{\Lambda}_s\mathbf{U}^\mathsf{T}$
2: Initialize parameter $\widehat{\beta}$: Apply the initialization method in Section 3.3.3 for variance/frequency
   shifting GBFs. For the other types of GBFs, apply a positive random initialization.
3: repeat
4:     Prefilter the sample covariance $\mathbf{S}$:
           $\mathbf{S}_{\text{pf}} = (h_{\widehat{\beta}}^{-1}(\mathbf{S}))^\dagger = \mathbf{U}(h_{\widehat{\beta}}^{-1}(\mathbf{\Lambda}_s))^\dagger\mathbf{U}^\mathsf{T}$    (3.25)
5:     Estimate $\mathbf{L}$ from prefiltered data ($\mathbf{S}_{\text{pf}}$):
           $\widehat{\mathbf{L}} \leftarrow$ Run Algorithm 2 to solve the problem in (3.33)
6:     Update filter parameter $\widehat{\beta}$ or skip if $\widehat{\beta}$ is optimal:
           $\widehat{\beta} \leftarrow$ Apply filter-specific update discussed in Section 3.3.3
7: until convergence has been achieved
8: return Graph system parameters $\widehat{\mathbf{L}}$ and $\widehat{\beta}$
```

3.3 Proposed Solution

Algorithm 3 is proposed to solve the graph system identification problem in (3.9) for a given sample covariance $\mathbf{S}$ and a selected type of graph-based filter h_β. After obtaining $\mathbf{U}$ and $\mathbf{\Lambda}_s$ via eigendecomposition of $\mathbf{S}$ and initialization of the parameter β (see lines 1 and 2), the algorithm performs three main steps to find the optimal graph Laplacian $\mathbf{L}$ and the filter parameter β:

1. *Prefiltering step* applies an inverse filtering on the sample covariance $\mathbf{S}$ to reverse the effect of filter h in the graph system. Without the prefiltering step, it may be impossible to effectively recover $\mathbf{L}$ from $\mathbf{S}$, since $\mathbf{\Sigma}^\dagger = h(\mathbf{L})^\dagger$ is generally not a graph Laplacian. The proposed prefiltering allows us to effectively estimate the original eigenvalues of $\mathbf{L}$ (i.e., $\mathbf{\Lambda}_\lambda$) from the prefiltered covariance $\mathbf{S}_{\text{pf}}$ in (3.25).

2. *Graph Laplacian estimation step* uses Algorithm 2 developed for estimating the CGL $\widehat{\mathbf{L}}$ from prefiltered covariance $\mathbf{S}_{\text{pf}}$.

3. *Filter parameter selection step* finds the best matching β for the graph system. Depending on the type of GBF, we propose different methods for parameter selection.

In the rest of this section, we derive the optimality conditions and discuss prefiltering, graph Laplacian estimation and filter parameter selection in Sections 3.3.1, 3.3.2 and 3.3.3, respectively. The optimality conditions are derived based on the following assumption on the number of data samples used to compute the sample covariance $\mathbf{S}$ (i.e., k).

Assumption 2. We assume that the number of data samples (k) used for estimating the sample covariance $\mathbf{S}$ is asymptotically large (i.e., $k \to \infty$), so the sample covariance $\mathbf{S}$ converges to the actual covariance matrix $\mathbf{\Sigma}$ almost surely by the strong law of large numbers. Thus, we have $\mathbf{S} = \mathbf{\Sigma}$ by the assumption.

Note that this is a theoretical assumption used to derive the optimality conditions for the graph system identification problem. In practice, the proposed algorithm (Algorithm 3) works regardless of the number of data samples used to obtain the sample covariance $\mathbf{S}$.

3.3.1 Optimal Prefiltering

Based on Assumption 2, we have a graph-based transform matrix $\mathbf{U}$ satisfying $\mathbf{L}=\mathbf{U}\mathbf{\Lambda}_\lambda\mathbf{U}^\intercal$, $h(\mathbf{L})^\dagger = \mathbf{U}h(\mathbf{\Lambda}_\lambda)^\dagger\mathbf{U}^\intercal$ and $\mathbf{S}=\mathbf{\Sigma}=\mathbf{U}\mathbf{\Lambda}_\sigma\mathbf{U}^\intercal$. By change of variables, the objective function in (3.8) becomes

$$\mathcal{J}(\mathbf{U}, h(\mathbf{\Lambda}_\lambda)) = \mathrm{Tr}(\mathbf{U}h(\mathbf{\Lambda}_\lambda)^\dagger\mathbf{U}^\intercal\mathbf{U}\mathbf{\Lambda}_\sigma\mathbf{U}^\intercal) - \log|\mathbf{U}h(\mathbf{\Lambda}_\lambda)^\dagger\mathbf{U}^\intercal| \tag{3.26}$$

which is simplified using properties of $\mathrm{Tr}(\cdot)$ and $|\cdot|$ as

$$\mathcal{J}(h(\mathbf{\Lambda}_\lambda)) = \mathrm{Tr}(h(\mathbf{\Lambda}_\lambda)^\dagger\mathbf{\Lambda}_\sigma) - \log|h(\mathbf{\Lambda}_\lambda)^\dagger| \tag{3.27}$$

where the graph-based filter h and the diagonal matrix of graph frequencies $\mathbf{\Lambda}_\lambda$ are unknown. By letting $\phi_i = h(\lambda_i)^\dagger = (h(\mathbf{\Lambda}_\lambda)^\dagger)_{ii}$ and $\sigma_i^2 = (\mathbf{\Lambda}_\sigma)_{ii}$ for $i=1,2,\dots,n$, we can write (3.27) as

$$\mathcal{J}(\boldsymbol{\phi}) = \sum_{i=1}^{n} \left(\phi_i\sigma_i^2 - \log(\phi_i)\right), \tag{3.28}$$

where $\boldsymbol{\phi} = [\phi_1\,\phi_2\,\cdots\,\phi_n]^\intercal$. In minimization of the convex function (3.28), the optimal solution satisfies the following necessary and sufficient conditions [67] obtained by taking the derivative of (3.28) with respect to ϕ_i and equating to zero,

$$\frac{\partial\mathcal{J}(\boldsymbol{\phi})}{\partial\phi_i} = \frac{1}{\phi_i} - \sigma_i^2 = 0. \tag{3.29}$$

By change of variables, the optimality conditions can be stated as

$$h(\lambda_i) = (h(\mathbf{\Lambda}_\lambda))_{ii} = (\mathbf{\Lambda}_\sigma)_{ii} = \sigma_i^2 \quad \forall i. \tag{3.30}$$

Based on Assumption 1, we can also write (3.30) as

$$h^{-1}(h(\lambda_i)) = \lambda_i = h^{-1}(\sigma_i^2) \quad \forall i, \tag{3.31}$$

where h^{-1} is inverse function of h. By using the matrix notation, we can state (3.31) more compactly as

$$h^{-1}(\mathbf{S}) = \mathbf{U}h^{-1}(\mathbf{\Lambda}_{\sigma})\mathbf{U}^{\mathsf{T}} = \mathbf{U}\mathbf{\Lambda}_{\lambda}\mathbf{U}^{\mathsf{T}} = \mathbf{L}. \tag{3.32}$$

This condition shows that we can find the optimal Laplacian $\mathbf{L}$ via inverse filtering (inverse prefiltering) $h^{-1}(\mathbf{S})$. Yet, (3.32) is satisfied only if Assumption 2 holds, which requires infinitely many samples (i.e., $k \to \infty$) to estimate the sample covariance $\mathbf{S}$. Thus, using (3.32) does not lead to a Laplacian matrix in practice. In order to address this problem, Algorithm 3 first estimates the prefiltered sample covariance $\mathbf{S}_{\text{pf}}$ as in (3.25), then employs Algorithm 2 to find the best graph Laplacian from $\mathbf{S}_{\text{pf}}$ by minimizing the criterion in (3.10).

3.3.2 Optimal Graph Laplacian Estimation

For a graph-based filter h (or h_β) satisfying the optimal prefiltering condition in (3.31), the graph system identification problem in (3.9) can be written as the following graph learning problem,

$$\begin{aligned} \underset{\mathbf{L}\succeq 0}{\text{minimize}} \quad & \text{Tr}(\mathbf{L}\mathbf{S}_{\text{pf}}) - \log|\mathbf{L}| + \|\mathbf{L}\odot\mathbf{H}\|_1 \\ \text{subject to} \quad & \mathbf{L}\,\mathbf{1} = \mathbf{0}, \quad (\mathbf{L})_{ij} \leq 0 \quad i \neq j \end{aligned} \tag{3.33}$$

where $\mathbf{S}_{\text{pf}} = (h^{-1}(\mathbf{S}))^{\dagger}$ is obtained by prefiltering the covariance $\mathbf{S}$. This problem is discussed in detail in Chapter 2 where we have developed Algorithm 2 (CGL) to solve (3.33).

3.3.3 Filter Parameter Selection

Depending on the type of GBF (in Table 3.1), we propose different methods for choosing the filter parameter β.

Initialization for Variance/Frequency Shifting Filters. Assuming that the optimal prefiltering condition in (3.31) holds, for both variance and frequency shifting GBFs, the optimal β is found by calculating $\sigma_1^2 = \mathbf{u}_1{}^{\mathsf{T}}\mathbf{S}\mathbf{u}_1$ where $\mathbf{u}_1$ is the eigenvector associated with the zero eigenvalue of the Laplacian $((\mathbf{\Lambda}_\lambda)_{11} = \lambda_1 = 0)$. Specifically,

- if $h_\beta(\lambda)$ is a variance shifting filter, we get

$$h_\beta(\lambda_1) = \sigma_1^2 = \mathbf{u}_1{}^{\mathsf{T}}\mathbf{S}\mathbf{u}_1 = \lambda_1^{\dagger} + \beta, \tag{3.34}$$

  since $\lambda_1 = \lambda_1^{\dagger} = 0$, the optimal β satisfies

$$\beta = \mathbf{u}_1{}^{\mathsf{T}}\mathbf{S}\mathbf{u}_1 = \sigma_1^2. \tag{3.35}$$

- if $h_\beta(\lambda)$ is a frequency shifting filter, we obtain

$$h_\beta(\lambda_1) = \sigma_1^2 = \mathbf{u_1}^\mathsf{T}\mathbf{S}\mathbf{u}_1 = 1/(\lambda_1+\beta) = 1/\beta, \tag{3.36}$$

so the optimal β satisfies

$$\beta = 1/(\mathbf{u_1}^\mathsf{T}\mathbf{S}\mathbf{u}_1) = 1/\sigma_1^2. \tag{3.37}$$

Since the optimal β can be directly estimated from $\mathbf{S}$ as in (3.35) and (3.37), Algorithm 3 uses the optimized initial β (in line 2) for prefiltering, and then estimates $\mathbf{L}$, so that the filter parameter update (line 6) is skipped for graph systems with frequency and variance shifting filters.
Exponential decay and Frequency Scaling Filters. Assuming that the condition in (3.31) holds and $\widehat{\mathbf{L}}$ is the solution of (3.33) where $\mathbf{S}_{\rm pf}$ is obtained by prefiltering with parameter $\widehat{\beta}$.

- If $h_\beta(\lambda)$ is a frequency scaling filter, the prefiltering gives

$$h_{\widehat{\beta}}^{-1}(\sigma_i^2) : \lambda_1 = \sigma_1^2 = 0, \;\; \frac{\beta}{\widehat{\beta}}\lambda_i = \frac{1}{\widehat{\beta}\sigma_i^2} \quad i=2,\dots,n. \tag{3.38}$$

- If $h_\beta(\lambda)$ is an exponential decay filter, we have

$$h_{\widehat{\beta}}^{-1}(\sigma_i^2) = -\frac{\log(\sigma_i^2)}{\widehat{\beta}} = -\frac{\log(\exp(-\beta\lambda_i))}{\widehat{\beta}} = \frac{\beta}{\widehat{\beta}}\lambda_i. \tag{3.39}$$

Based on (3.38) and (3.39), for any selected $\widehat{\beta}$, the resulting graph Laplacian satisfies $\widehat{\mathbf{L}} = (\beta/\widehat{\beta})\mathbf{L}$ where $\mathbf{L}$ and β denote the original graph system parameters. So, Algorithm 3 finds the optimal combinatorial Laplacian $\mathbf{L}$ up to a constant scale factor $\beta/\widehat{\beta}$, and the parameter $\widehat{\beta}$ can be tuned so that the desired normalization (scaling factor) for $\mathbf{L}$ is achieved.
β-hop Localized Filter. For estimation of the optimal hop count β in Algorithm 3, prefiltering with $\widehat{\beta}$ gives,

$$h_{\widehat{\beta}}^{-1}(\sigma_i^2) : \lambda_1 = \sigma_1^2 = 0, \;\; \lambda_i^{\beta/\widehat{\beta}} = \left(\frac{1}{\sigma_i^2}\right)^{1/\widehat{\beta}} \quad i=2,\dots,n. \tag{3.40}$$

Since this requires the graph learning step to estimate $\mathbf{L}^{\beta/\widehat{\beta}}$, which is not a graph Laplacian in general, Algorithm 3 cannot guarantee optimal graph system identification unless $\beta = \widehat{\beta}$. In order to find the optimal β, we perform a line search for given range of integers optimizing the following:

$$\widehat{\beta} = \underset{\widehat{\beta}\in\mathbb{N}}{\operatorname{argmin}} ||h_{\widehat{\beta}}(\widehat{\mathbf{L}}) - \mathbf{S}||_F. \tag{3.41}$$

3.4 Results

3.4.1 Graph Learning from Diffusion Signals/Data

We evaluate the performance of our proposed graph system identification (GSI) method (i.e., Algorithm 3) by benchmarking against the current state-of-the-art approaches proposed for learning graph from smooth signals (GLS) [31, 32] as well as the graph topology inference (GTI) in [34]. The proposed GSI is also compared against the CGL estimation algorithm (CGL [36]) in Chapter 2 (i.e., using Algorithm 2 without prefiltering) and the inverse prefiltering (IPF) approach, which estimates a graph Laplacian matrix by inverting the prefiltered covariance, $\mathbf{S}_{\text{pf}}$ in (3.25), so that $\widehat{\mathbf{L}} = h_\beta^{-1}(\mathbf{S}) = \mathbf{S}_{\text{pf}}^{\dagger}$. For this purpose, we generate several artificial datasets based on the signal model in (3.1), defined by a graph Laplacian ($\mathbf{L}$) and a GBF (h_β) where the dataset entries are generated by random sampling from the distribution $\mathsf{N}(\mathbf{0}, h_\beta(\mathbf{L}))$. Then, the generated data is used in the proposed and benchmark algorithms to recover the corresponding graph Laplacian $\mathbf{L}$. We repeat our experiments for different $\mathbf{L}$ and h_β where graphs are constructed by using three different graph connectivity models:

- Grid graph, $\mathcal{G}_{\text{grid}}^{(n)}$, consisting n vertices attached to their four nearest neighbors (except the vertices at boundaries).
- Random Erdos-Renyi graph, $\mathcal{G}_{\text{ER}}^{(n,p)}$, with n vertices attached to other vertices with probability $p = 0.2$.
- Random modular graph (also known as stochastic block model), $\mathcal{G}_{\text{M}}^{(n,p_1,p_2)}$ with n vertices and four modules (subgraphs) where the vertex attachment probabilities across modules and within modules are $p_1 = 0.1$ and $p_2 = 0.2$, respectively.

Then, the edge weights of a graph are randomly selected from the uniform distribution $\mathsf{U}(0.1, 3)$, on the interval $[0.1, 3]$. For each $\mathbf{L}$ and h_β pair, we perform Monte-Carlo simulations to test average performance of proposed and benchmark methods with varying number of data samples (k) and fixed number of vertices ($n = 36$)[a]. To measure the estimation performance, we use the following two metrics:

$$\mathsf{RE}(\widehat{\mathbf{L}}, \mathbf{L}^*) = \frac{\|\widehat{\mathbf{L}} - \mathbf{L}^*\|_F}{\|\mathbf{L}^*\|_F} \tag{3.42}$$

which is the relative error between the ground truth graph ($\mathbf{L}^*$) and estimated graph parameters ($\widehat{\mathbf{L}}$), and

$$\mathsf{FS}(\widehat{\mathbf{L}}, \mathbf{L}^*) = \frac{2\,\mathsf{tp}}{2\,\mathsf{tp} + \mathsf{fn} + \mathsf{fp}} \tag{3.43}$$

[a]Methods are evaluated on small graphs (with $n = 36$ vertices), since GTI [34] is implemented using CVX [64] and do not currently have an efficient and scalable implementation. The proposed and the other benchmark methods are more efficient and can support larger graphs.

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

In our experiments, for the proposed GSI, the regularization parameter α in (3.33) is selected from the following set:

$$\{0\} \cup \{0.75^r (s_{\max}\sqrt{\log(n)/k}) \,|\, r = 1, 2, 3, \dots, 14\}, \tag{3.44}$$

where $s_{\max} = \max_{i \neq j} |(\mathbf{S})_{ij}|$ is the maximum off-diagonal entry of $\mathbf{S}$ 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 [72, 73]. Monte-Carlo simulations are performed for each proposed/baseline method, by successively solving the associated problem with different regularization parameters to find the best regularization that minimizes RE. The corresponding graph estimate is also used to calculate FS. For all baseline methods [31, 32, 34], the required parameters are selected by fine tuning. Since CGL [36], GLS [31, 32] and GTI [34] approaches generally result in severely biased solutions with respect to the ground truth $\mathbf{L}^*$ (based on our observations from the experiments), RE values are calculated after normalizing the estimated solution $\mathbf{L}$ as $\widehat{\mathbf{L}} = (\mathrm{Tr}(\mathbf{L}^*)/\mathrm{Tr}(\mathbf{L}))\mathbf{L}$. Note that, this normalization also resolves the ambiguity in identification of graph systems with exponential decay and frequency scaling filters up to a scale factor (discussed in Section 3.3.3).

Figures 3.3 and 3.4 depict the performance of different methods applied for estimating graphs from signals modeled based on exponential decay filters (diffusion kernels) and β-hop localized filters. As shown in Figures 3.3 and 3.4, the proposed GSI significantly outperforms all baseline methods, including the state-of-the-art GLS [31, 32] and GTI [34], in terms of average RE and FS metrics. The performance difference between GSI and CGL [36] demonstrates the impact of the prefiltering step, which substantially improves the graph learning accuracy. Similarly, the performance gap between GSI and IPF shows the advantage of Algorithm 3 compared to the direct prefiltering of input covariance ($\mathbf{S}$) as in (3.32), where GSI provide better graph estimation especially when number of data samples (i.e., k/n) is small. Besides, Figures 3.5 and 3.6 illustrate two examples from the experiments with grid graphs for the case of $k/n = 30$, where the proposed GSI constructs graphs that are the most similar to the ground truth ($\mathbf{L}^*$).

3.4.2 Graph Learning from Variance/Frequency Shifted Signals

In this subsection, we compare the CGL estimation performance of GSI, CGL [36] and SCGL [33] methods from signals modeled based on variance and frequency shifting GBFs. As discussed in Section 3.2.3, the covariance matrices for signals modeled based on these GBFs with parameter β are

- $\mathbf{\Sigma} = \mathbf{L}^{\dagger} + \beta\mathbf{I}$ for variance shifting,
- $\mathbf{\Sigma} = (\mathbf{L} + \beta\mathbf{I})^{\dagger}$ for frequency shifting.

where $\mathbf{L}$ denotes the associated combinatorial Laplacian.

In our experiments, we construct 10 random Erdos-Renyi graphs ($\mathcal{G}_{\mathrm{ER}}^{(n,p)}$) with $n = 36$ vertices and $p = 0.2$, then generate $\mathbf{\Sigma}$ for each GBF by varying β between 0 and 1. To evaluate the effect of β only, we use actual covariance matrices instead of sample covariances as input to the algorithms. So, GSI, CGL and SCGL estimate a graph Laplacian $\mathbf{L}$ from $\mathbf{\Sigma}$. The average RE results are presented in Tables 3.3 and 3.4 for various β.

Table 3.3 shows that the proposed GSI significantly outperforms CGL for $\beta > 0$, and the average RE difference increases as β gets larger. This is because the variance shifting GBF leads to the noisy signal model with the covariance in (3.11) where β represents the variance of the noise (σ^2), and the prefiltering step allows GSI to perfectly estimate the parameter β from $\mathbf{\Sigma}$ by using (3.35) so that

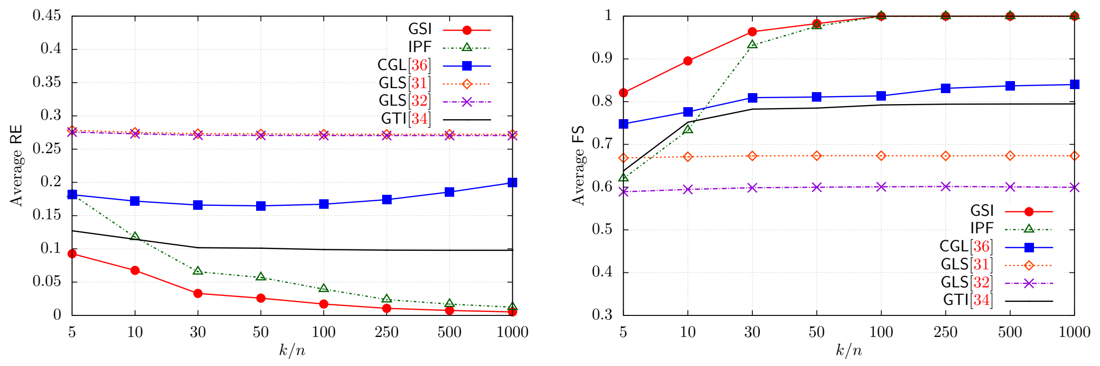


Figure 3.3: Average RE and FS results for graph estimation from signals modeled based on exponential decay GBFs tested with $\beta = \{0.5, 0.75\}$ on 10 different grid, Erdos-Renyi and modular graphs (30 graphs in total). The proposed GSI outperforms all baseline methods in terms of both RE and FS.

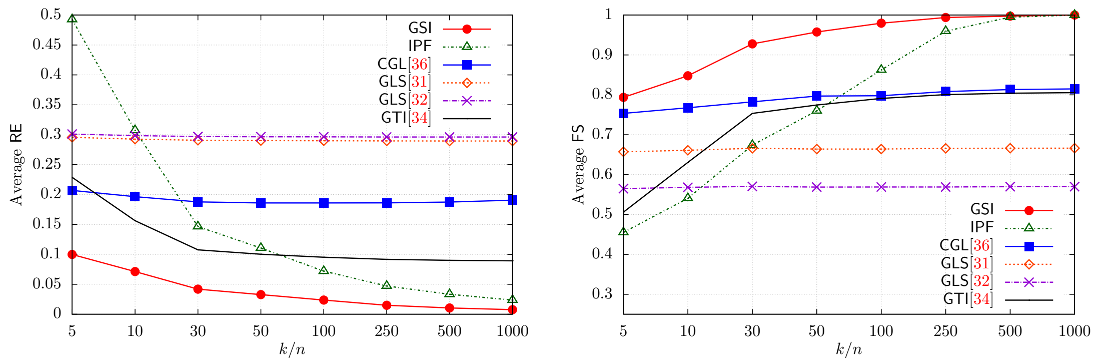
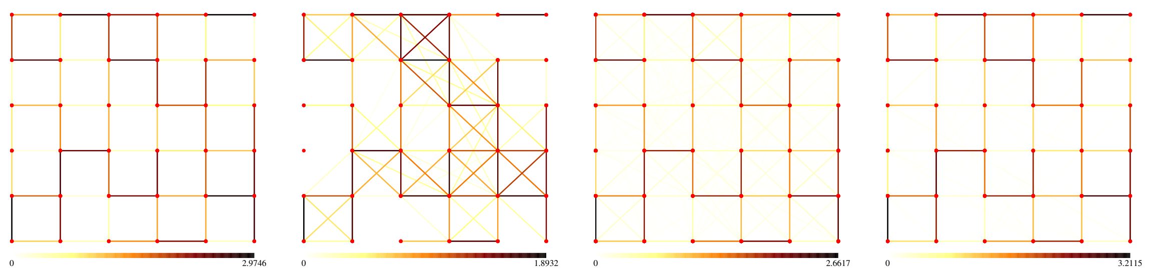
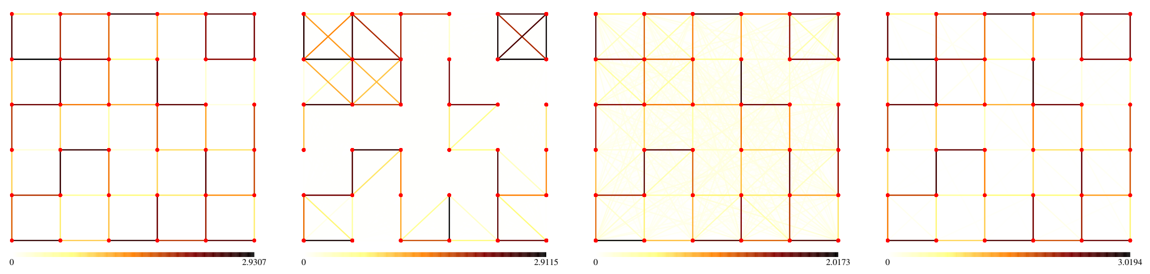


Figure 3.4: Average RE and FS results for graph estimation from signals modeled based on β-hop localized GBFs tested with $\beta = \{2, 3\}$ on 10 different grid, Erdos-Renyi and modular graphs (30 graphs in total). The proposed GSI outperforms all baseline methods in terms of both RE and FS.



(a) The ground truth CGL ($\mathbf{L}^*$) (b) GLS [31]: (RE,FS) = (0.4019,0.6172) (c) GTI [34]: (RE,FS) = (0.1201,0.7229) (d) GSI: (RE,FS) = (0.0340,0.9917)

Figure 3.5: A sample illustration of graph estimation results (for $k/n = 30$) from signals modeled using the exponential decay GBF with $\beta = 0.75$ and $\mathbf{L}^*$ is derived from the grid graph in (a). The edge weights are color coded where darker colors indicate larger weights. The proposed GSI leads to the graph that is the most similar to the ground truth.



(a) The ground truth CGL ($\mathbf{L}^*$) (b) GLS [31]: (RE,FS) = (0.4045,0.6890) (c) GTI [34]: (RE,FS) = (0.2118,0.6030) (d) GSI: (RE,FS) = (0.0274,0.9833)

Figure 3.6: A sample illustration of graph estimation results (for $k/n = 30$) from signals modeled using the β-hop localized GBF with $\beta = 2$ and $\mathbf{L}^*$ is derived from the grid graph in (a). The edge weights are color coded where darker colors indicate larger weights. The proposed GSI leads to the graph that is the most similar to the ground truth.

the covariance is prefiltered as in (3.25) based on the optimal β. The prefiltering step can also be considered as a *denoising operation* (reversing the effect of variance shifting GBFs) on the signal covariance before the graph estimation step, while CGL work with noisy (i.e., shifted) covariances, which diminish the CGL estimation performance. For $\beta = 0$ (i.e., $\mathbf{\Sigma}$ is noise-free), the problem (3.9) reduces to the CGL estimation problem in [36], so both GSI and CGL lead to the same average RE.

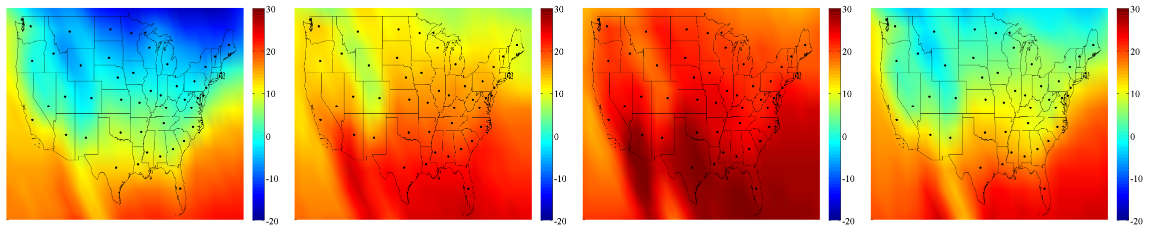
For the frequency shifting GBFs with $\beta > 0$, GSI performs slightly better than SCGL, since SCGL is implemented using a general purpose solver CVX [64], which generally produces less accurate solutions compared to our algorithm. Moreover, our algorithm is approximately 90 times faster than SCGL on average, and significantly outperforms SCGL for $\beta = 0$, since SCGL method is developed for shifted covariance matrices (i.e., $\mathbf{L} + \beta\mathbf{I}$) where β needs to be strictly positive.

Table 3.3: Average Relative Errors for Variance Shifting GBF

	Filter parameter (β)					
Method	0	0.1	0.3	0.5	0.7	0.9
CGL [36]	2×10^{-4}	0.60	0.79	0.85	0.88	0.89
GSI	2×10^{-4}					

Table 3.4: Average Relative Errors for Frequency Shifting GBF

	Filter parameter (β)			
Method	0	0.1	0.5	0.9
SCGL [33]	0.2354	6.7×10^{-4}	6.6×10^{-4}	6.2×10^{-4}
GSI	1.6×10^{-4}			



(a) 45th day (Winter) (b) 135th day (Spring) (c) 225th day (Summer) (d) 315th day (Autumn)

Figure 3.7: Average air temperatures (in degree Celsius) for (a) 45th, (b) 135th, (c) 225th and (d) 315th days over 16 years (2000-2015). Black dots denote 45 states.

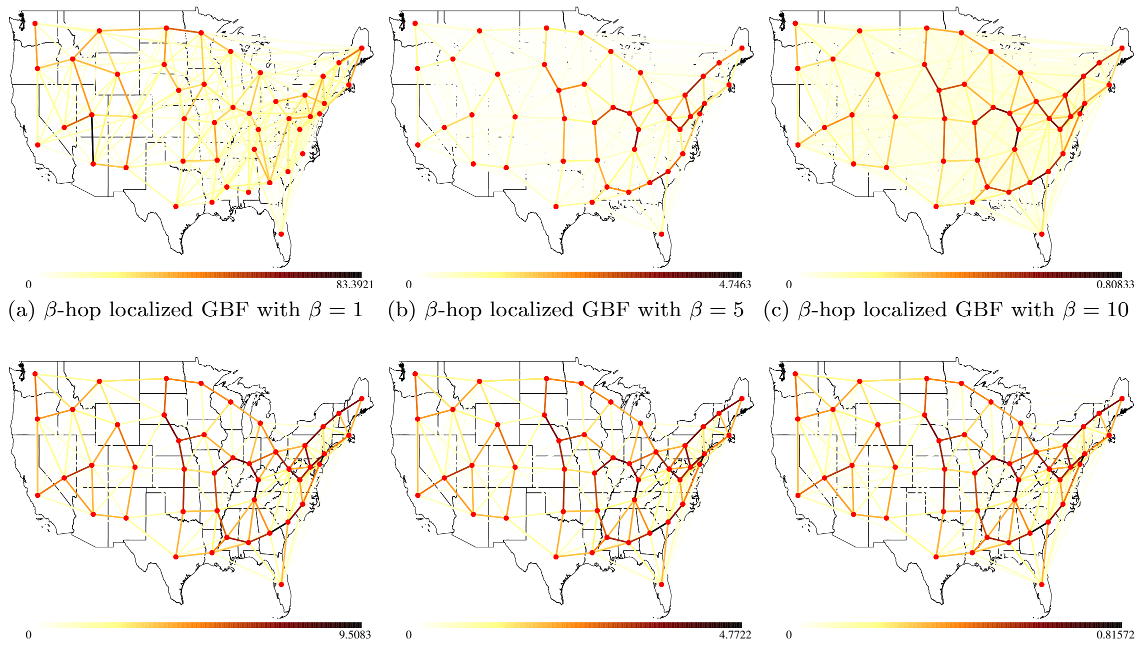
3.4.3 Illustrative Results on Temperature Data

In this experiment, we apply our proposed method on a real (climate) dataset[a] consisting of air temperature measurements [80]. We specifically use the average daily temperature measurements collected from 45 states in the US over 16 years (2000-2015), so that in total there are $k = 5844$ samples for each of the $n = 45$ states. Figure 3.7 shows samples of average temperature signals, which are spatially smooth across different states. Also, *the Rocky Mountains* region has lower average temperature values as compared to the other regions in the western US[b].

The goal of this experiment is to learn graphs (with 45 vertices) associated with exponential decay and β-hop localized filters from temperature data and investigate the effect of filter type and parameter β on the resulting graphs, representing the similarity of temperature conditions between the 45 states. For modeling temperature signals, a diffusion kernel is a good candidate, because it is

[a]NCEP Reanalysis data provided by the NOAA/OAR/ESRL PSD, Boulder, Colorado, USA, from their website at http://www.esrl.noaa.gov/psd/

[b]The Rocky Mountains cross through the states of Idaho, Montana, Wyoming, Utah, Colorado and New Mexico. For example, in Figure 3.7b, the areas with temperature values between 0 and 10 degrees Celsius (colored in green) correspond to the Rocky Mountains region approximately.



(a) β-hop localized GBF with $\beta = 1$ (b) β-hop localized GBF with $\beta = 5$ (c) β-hop localized GBF with $\beta = 10$

(d) Exponential decay with $\beta = 0.25$ (e) Exponential decay with $\beta = 0.5$ (f) Exponential decay with $\beta = 3$

Figure 3.8: Graphs learned from temperature data using the GSI method with exponential decay and β-hop localized GBFs for fixed β parameters where no regularization is applied (i.e., $\mathbf{H}$ is set to the zero matrix). The edge weights are color coded so that darker colors represent larger weights (i.e., more similarities). The graphs associated with exponential decay GBF leads to sparser structures compared to the graphs corresponding to β-hop localized GBFs.

a fundamental solution of the *heat equation*[a], which describes the distribution of heat in a physical environment [81].

Figure 3.8 illustrates the graphs estimated using the GSI method without ℓ_1-regularization (i.e., $\mathbf{H}$ in (3.9) is set to the zero matrix). As shown in the figure, larger edge weights are assigned between vertices (i.e., states) that are closer to each other in general, since temperature values are mostly similar between states within close proximity. However, the distance between states is obviously not the only factor effecting the similarity of temperature values. For example, in Figures 3.8b–3.8f, the weights are considerably small between the states in the Rocky Mountains region and their neighboring states in the east (e.g., Nebraska and Kansas) due to the large differences in altitude. Note also that different choices of GBFs can lead to substantially different similarity graphs. Especially for the β-hop localized GBF with $\beta = 1$ (corresponding to the CGL method), the resulting graph is significantly different than the results in Figures 3.8b and 3.8c, since the 1-hop localized filter does not lead to a diffusion model. The graphs associated with exponential decay GBF leads to sparser graphs, better revealing the structure of the signal, compared to the graphs in

[a]This is the reason that diffusion kernels are also known as heat kernels in the literature.

Figures 3.8a–3.8c. The structure of the graphs in Figures 3.8d–3.8f are similar for different β because of the relation in (3.39) for the exponential decay filter. For example, increasing β from 0.25 to 0.5 approximately halves edge weights, as shown in Figures 3.8d and 3.8e. Besides, the distribution of edge weights for β-hop localized GBFs becomes more similar to the ones in Figures 3.8d–3.8f as β gets larger.

3.5 Conclusion

We have introduced a novel graph-based modeling framework by (i) formulating the modeling problem as the graph system identification from signals/data and (ii) proposing an algorithm that jointly identifies a graph and a graph-based filter. The proposed framework supports various types of graph-based filters which include diffusion kernels as a special case. Our experimental results have demonstrated that the proposed method outperform the state-of-the-art approaches in terms of modeling accuracy.

Chapter 4

Graph Learning from Multiple Graphs: Multigraph Combining

In the previous two chapters, we have studied methods to learn graphs from data under two different model assumptions based on attractive GMRFs and graph systems. The present chapter focuses on a different graph learning problem, called multigraph combining, where the goal is to learn a single graph from multiple graphs (i.e., the observed data consist of a list of graphs). We particularly consider combining simple weighted graphs (i.e., graphs with no self-loops) associated with combinatorial graph Laplacians (CGLs)[a] and propose a three-step formulation based on a maximum likelihood (ML) criterion. In the first two steps, the *common graph-based transform (CGBT)* and *common graph frequency (CGF)* estimation problems are proposed to estimate a GBT ($\widehat{\mathbf{U}}$) and a diagonal matrix ($\widehat{\mathbf{\Lambda}}$) consisting of graph frequencies. By exploiting the optimality conditions of the problems, we propose a method that estimates the best CGBT and CGFs in an ML sense. The third step involves estimating a CGL matrix based on the optimized $\widehat{\mathbf{U}}$ and $\widehat{\mathbf{\Lambda}}$, where we employ Algorithm 2 described in Chapter 2 to learn a CGL matrix.

This chapter is organized as follows. Section 4.1 presents the related work. Section 4.2 first formulates the multigraph combining problem. In Section 4.3, we introduce the proposed solution. Experimental results are presented in Section 4.4, and Section 4.5 draws some conclusions.

4.1 Related Work

In the literature, there is only a limited number of studies on graph combining. The authors in [83, 84] propose graph combining to improve spectral clustering and semi-supervised learning with multiple graphs, respectively. However, their approaches are based on weighted averaging of graphs,

An earlier version of the work in this chapter is published in [82].

[a]The work in this chapter can be trivially extended to learn other types of graph Laplacians.

which have no theoretical guarantees. Additionally, our CGBT estimation problem is closely related to *common principal component* (CPC) estimation originally introduced in [85], where the goal is finding an orthogonal matrix that jointly diagonalizes multiple covariance matrices in an ML sense, and an iterative algorithm for this problem is developed in [86]. Our CGBT problem can be viewed as a variation of the CPC problem with graph Laplacian matrices. Another variation of the CPC problem was also studied in blind source separation [87], and the JADE algorithm [88] was introduced for joint diagonalization with a Frobenius-norm criterion. Algorithmically, our algorithm for the CGBT estimation problem is very similar to the ones in [86, 88], which are all based on Jacobi-like iterations [89]. Yet, our algorithm iteratively updates an orthogonal matrix (i.e., a CGBT) based on pairwise projections on graph Laplacian matrices, while [86] and [88] use covariance matrices. To the best of our knowledge, none of the prior studies address the CGF estimation problem. Note that the graph topology inference problem [34] stated in (2.18) applies the same type of decomposition on the target variable (i.e., $\mathbf{\Theta} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^{\intercal}$) to estimate graph frequencies (i.e., $\mathbf{\Lambda}$) for a given GBT (i.e., $\mathbf{U}$). Since the graph combining problem is not considered in [34], $\mathbf{U}$ is first obtained by the eigendecomposition of a single covariance matrix, and then $\mathbf{\Lambda}$ is estimated from $\mathbf{U}$ by solving an ℓ_1-minimization problem, while we sequentially solve CGBT and CGF estimation problems to optimize $\mathbf{U}$ and $\mathbf{\Lambda}$ from multiple graphs based on an ML criterion.

4.2 Problem Formulation for Multigraph Combining

Our formulation is based on the following two basic assumptions on graphs:

- (*Number of vertices*) All of the given graphs have the same vertex set $\mathcal{V}$ with n vertices.
- (*Connected graphs*) Each of the given graphs is a connected graph.

The proposed three-step formulation is derived from the following general optimization problem for given positive semidefinite matrices $\mathbf{K}_1, \dots, \mathbf{K}_L$ and positive weights $k_1, \dots, k_L$:

$$\begin{array}{ll} \underset{\mathbf{U},\mathbf{\Lambda}}{\text{minimize}} & \displaystyle\sum_{l=1}^{L} k_l \left(\text{Tr}\left(\mathbf{\Theta}\mathbf{K}_l\right) - \log|\mathbf{\Theta}|\right) \\ \text{subject to} & \mathbf{\Theta} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^{\intercal} \quad \mathbf{U}^{\intercal}\mathbf{U} = \mathbf{I} \\ & \mathbf{\Theta} \in \mathcal{L}_c \end{array} \tag{4.1}$$

where the orthogonal matrix $\mathbf{U}$ and the diagonal matrix $\mathbf{\Lambda}$ are the target variables, and $\mathcal{L}_c$ denotes the set of CGLs as stated in (1.3). The weights $k_1, \dots, k_L$ can be selected to adjust the contribution of each $\mathbf{K}_1, \dots, \mathbf{K}_L$ in the problem, so that choosing a larger weight k_l increases the contribution of $\mathbf{K}_l$ in the minimization. Depending on the choices/types of $\mathbf{K}_1, \mathbf{K}_2, \dots, \mathbf{K}_L$, the problem has two variants:

- *(Statistical)* For given L sample covariance matrices $\mathbf{S}_1, \dots, \mathbf{S}_L$ corresponding to L groups (clusters) of data, the problem in (4.1) can be solved to estimate a graph Laplacian from L groups of data by setting $\mathbf{K}_l = \mathbf{S}_l$ for $l = 1, \dots, L$.
- *(Deterministic)* For given L graph Laplacian matrices (e.g., CGLs) $\mathbf{L}_1, \dots, \mathbf{L}_L$, setting $\mathbf{K}_l = \mathbf{L}_l^\dagger$ in (4.1) for $l = 1, \dots, L$ leads to the multigraph combining problem, which is the problem of interest in this chapter.

Note that (4.1) is nonconvex due to the orthogonality constraint (i.e., $\mathbf{U}^\intercal\mathbf{U} = \mathbf{I}$), which makes it hard to solve for $\mathbf{U}$ and $\mathbf{\Lambda}$ directly. Thus, we present a three-step formulation for the multigraph combining problem by decomposing (4.1) into the following three subproblems:

- *Common graph-based transform (CGBT) problem* is formulated with the goal of estimating a CGBT $\mathbf{U}$ and multiple diagonal matrices $\mathbf{\Lambda}_1, \dots, \mathbf{\Lambda}_L$ from given $\mathbf{K}_1, \dots, \mathbf{K}_L$ as follows:

$$\begin{aligned} \underset{\mathbf{U}, \mathbf{\Lambda}_1, \dots, \mathbf{\Lambda}_L}{\text{minimize}} \quad & \sum_{l=1}^{L} k_l \left(\operatorname{Tr}\left(\mathbf{\Lambda}_l \mathbf{U}^\intercal \mathbf{K}_l \mathbf{U}\right) - \log|\mathbf{\Lambda}_l| \right) \\ \text{subject to} \quad & \mathbf{U}^\intercal\mathbf{U} = \mathbf{I} \end{aligned} \tag{4.2}$$

  The basic purpose of this step is to find the best orthogonal transform $\mathbf{U}$ by minimizing the weighted ML criterion above in (4.2). Then, the resulting diagonal matrices, denoted as $\widehat{\mathbf{\Lambda}}_1, \dots, \widehat{\mathbf{\Lambda}}_L$, are used in the next step to optimize a CGF matrix.

- *Common graph frequency (CGF) problem* is formulated to estimate a CGF matrix $\mathbf{\Lambda}$ from the diagonal matrices $\widehat{\mathbf{\Lambda}}_1, \dots, \widehat{\mathbf{\Lambda}}_L$ obtained by solving (4.2) as:

$$\underset{\mathbf{\Lambda}}{\text{minimize}} \quad \sum_{l=1}^{L} k_l \left(\operatorname{Tr}(\mathbf{\Lambda}\widehat{\mathbf{\Lambda}}_l^\dagger) - \log|\mathbf{\Lambda}| \right) \tag{4.3}$$

- Since the estimated CGBT and CGF matrices (i.e., $\widehat{\mathbf{U}}$ and $\widehat{\mathbf{\Lambda}}$) generally do not lead to a CGL matrix, we propose the following CGL problem, discussed in Chapter 2, to optimize a CGL based on $\widehat{\mathbf{U}}$ and $\widehat{\mathbf{\Lambda}}$:

$$\begin{aligned} \underset{\mathbf{\Theta}}{\text{minimize}} \quad & \operatorname{Tr}(\mathbf{\Theta}\mathbf{S}) - \log|\mathbf{\Theta}| \\ \text{subject to} \quad & \mathbf{\Theta} \in \mathcal{L}_c \end{aligned} \tag{4.4}$$

  where $\mathbf{S} = \widehat{\mathbf{U}}\widehat{\mathbf{\Lambda}}^\dagger\widehat{\mathbf{U}}^\intercal$, and $\mathcal{L}_c$ denotes the set of CGLs. This step can be viewed as a projection for finding the closest CGL matrix with respect to the $\mathbf{S}$ in an ML sense.

As an alternative to the above three-step formulation, the problem in (4.1) can be solved without splitting the target variable $\mathbf{\Theta}$ into $\mathbf{U}$ and $\mathbf{\Lambda}$, so that it reduces to the CGL problem in (4.4) where $\mathbf{S} = 1/k_{\text{sum}} \sum_{l=1}^{L} k_l \, \mathbf{K}_l$ such that $k_{\text{sum}} = \sum_{l=1}^{L} k_l$ is the normalization factor. In this case, the multigraph combining is simply carried out by first computing a weighted summation of input matrices $\mathbf{K}_1, \mathbf{K}_2, \dots, \mathbf{K}_L$ defining $\mathbf{S}$ and then solving the CGL problem with $\mathbf{S}$ as the input. Yet, our formulation introduces CGBT and CGF problems by decomposing the target variable as $\mathbf{\Theta} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\mathsf{T}$, which has two main advantages over directly solving the CGL problem. Firstly, splitting the target variable $\mathbf{\Theta}$ into $\mathbf{U}$ and $\mathbf{\Lambda}$ introduces more degrees of freedom for an algorithm to search for the best solutions, so that solving the CGBT and CGF problems potentially leads to better estimation of $\mathbf{U}$ and $\mathbf{\Lambda}$ in an ML sense. This advantage will be empirically demonstrated later in Section 4.4. Secondly, for a given set of L input CGLs $\mathbf{L}_1, \dots, \mathbf{L}_L$, directly solving the CGL problem requires us to compute the pseudoinverse of each CGL to have $\mathbf{K}_l = \mathbf{L}_l^\dagger$ for $l = 1, \dots, L$ used in calculating the $\mathbf{S}$, which can be infeasible for large L because of the computationally intensive pseudoinverse operation. However, the proposed three-step formulation allows us to work with CGLs directly, since a CGBT ($\mathbf{U}$) is invariant to pseudoinverses of input CGLs, and also corresponding graph frequencies can be simply obtained for a given $\mathbf{U}$ by using $\mathbf{\Lambda}_l = \mathbf{U}^\mathsf{T}\mathbf{L}_l\mathbf{U}$.

In order to justify our three-step formulation, the following section presents the statistical derivations of the proposed problems as well as their optimality conditions used to develop our algorithm.

4.2.1 Statistical Formulation and Optimality Conditions

Let $\mathbf{x}_l$ be an n-variate Gaussian random vector $\mathbf{x}_l \sim \mathsf{N}(\mathbf{0}, \mathbf{L}_l^\dagger)$ for $l = 1, 2, \dots, L$. Given $\widetilde{k}_l = k_l + 1$ independent random vectors the random data matrix $\mathbf{X}_l = [\mathbf{x}_l^{(1)}\, \mathbf{x}_l^{(2)} \cdots \mathbf{x}_l^{(\widetilde{k}_l)}]$ leads to the random empirical covariance matrix $\mathbf{S}_l = (1/k_l)\mathbf{X}_l\mathbf{X}_l^\mathsf{T}$. Then, the corresponding scatter matrix $\widetilde{\mathbf{S}}_l = k_l\mathbf{S}_l$ has a Wishart distribution, $\widetilde{\mathbf{S}}_l \sim \mathsf{W}(\mathbf{L}_l^\dagger, k_l)$, which is a common data model used in covariance estimation. Since $\widetilde{\mathbf{S}}_l$ are independent for $l = 1, 2, \dots, L$, the likelihood function of $\mathbf{L}_1, \mathbf{L}_2, \dots, \mathbf{L}_L$ is

$$\mathsf{p}(\{\widetilde{\mathbf{S}}_l\}_{l=1}^L | \{\mathbf{L}_l\}_{l=1}^L) = \prod_{l=1}^{L} C_l |\mathbf{L}_l^\dagger|^{-\frac{k_l}{2}} \exp\left(-\frac{k_l}{2}\mathrm{Tr}\left(\mathbf{L}_l\mathbf{S}_l\right)\right) \tag{4.5}$$

where C_l is a constant that does not depend on $\mathbf{L}_l$. Thus, we can write negative log-likelihood objective function as follows,

$$\mathcal{J}_{\mathrm{NLL}}(\mathbf{L}_1, \mathbf{L}_2, \dots, \mathbf{L}_L) = \sum_{l=1}^{L} k_l \left(-\log|\mathbf{L}_l| + \mathrm{Tr}\left(\mathbf{L}_l\mathbf{S}_l\right)\right), \tag{4.6}$$

which leads to the objective function of the proposed problem in (4.1) for $\mathbf{\Theta} = \mathbf{L}_1 = \cdots = \mathbf{L}_L$ where $\mathbf{\Theta}$ is the target variable that combines $\mathbf{L}_1, \dots, \mathbf{L}_L$. Without loss of generality, the weights $k_1, \dots, k_L$ can be normalized by dividing (4.6) with the constant $k_{\mathrm{sum}} = \sum_{l=1}^{L} k_l$, so that $\sum_{l=1}^{L} k_l/k_{\mathrm{sum}} = 1$.

Common Graph-based Transform (CGBT) Estimation. In order to find the best CGBT matrix, we introduce the following constraint on CGL matrices,

$$\mathcal{C}_{\text{CGBT}}: \ \mathbf{L}_l = \mathbf{U}\boldsymbol{\Lambda}_l\mathbf{U}^\intercal \qquad \text{for } l = 1, 2, \dots, L, \tag{4.7}$$

where $\mathbf{U}$ denotes the orthogonal CGBT matrix that we seek to obtain, which is supposed to (approximately) jointly diagonalize $\mathbf{L}_1, \dots, \mathbf{L}_L$, and $\{\boldsymbol{\Lambda}_l\}_{l=1}^L$ is the corresponding set of diagonal matrices consisting of graph frequencies. Based on (4.7), we can rewrite $\mathcal{J}_{\text{NLL}}$ objective function in (4.6) as

$$\mathcal{J}_{\text{NLL}}(\mathbf{U}, \{\boldsymbol{\Lambda}_l\}_{l=1}^L) = \sum_{l=1}^{L} k_l \left(-\text{log}|\boldsymbol{\Lambda}_l| + \text{Tr}\left(\boldsymbol{\Lambda}_l\mathbf{U}^\intercal\mathbf{S}_l\mathbf{U}\right)\right) \tag{4.8}$$

Since $\lambda_1^{(l)} = 0$ for $l = 1, .., L$ and $\mathbf{u}_1 = (1/\sqrt{n})\mathbf{1}$ by properties of CGL matrices, we can simplify (4.8) as follows,

$$\mathcal{J}_{\text{NLL}}(\widetilde{\mathbf{U}}, \widetilde{\boldsymbol{\lambda}}) = \sum_{l=1}^{L} k_l \sum_{j=2}^{n} \left(-\text{log}(\lambda_j^{(l)}) + \lambda_j^{(l)}\mathbf{u}_j^\intercal\mathbf{S}_l\mathbf{u}_j\right), \tag{4.9}$$

where $\mathbf{u}_j$ is the j-th column of $\mathbf{U}$ and $\lambda_j^{(l)} = (\boldsymbol{\Lambda}_l)_{jj}$. Also, the variables $\{\mathbf{u}_j\}_{j=2}^n$ and $\{\lambda_2^{(l)}, \dots, \lambda_n^{(l)}\}_{l=1}^L$ are compactly denoted by $\widetilde{\mathbf{U}}$ and $\widetilde{\boldsymbol{\lambda}}$, respectively. Thus, the minimization of the negative log-likelihood in (4.6) under the constraint $\mathcal{C}_{\text{CGBT}}$ leads to the following problem,

$$\begin{aligned} \underset{\widetilde{\mathbf{U}}, \widetilde{\boldsymbol{\lambda}}}{\text{minimize}} \quad & \sum_{l=1}^{L} k_l \sum_{j=2}^{n} \left(-\text{log}(\lambda_j^{(l)}) + \lambda_j^{(l)}\mathbf{u}_j^\intercal\mathbf{S}_l\mathbf{u}_j\right) \\ \text{subject to} \quad & \mathbf{u}_i^\intercal\mathbf{u}_j = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases} \end{aligned} \tag{4.10}$$

which is equivalent to the CGBT problem in (4.2). The optimization problem in (4.10) is nonconvex due to its orthogonality constraints. Thus, we derive necessary conditions for local optimality using the Lagrange multiplier theorem [67]. The Lagrangian function associated to (4.10) is

$$\mathcal{J}_{\text{LAG}}(\widetilde{\mathbf{U}}, \widetilde{\boldsymbol{\lambda}}, \boldsymbol{\mu}) = \mathcal{J}_{\text{NLL}}(\widetilde{\mathbf{U}}, \widetilde{\boldsymbol{\lambda}}) + \sum_{i=2}^{n} \mu_i(\mathbf{u}_i^\intercal\mathbf{u}_i - 1) + 2\sum_{2\leq i<j\leq n} \mu_{ij}\mathbf{u}_i^\intercal\mathbf{u}_j, \tag{4.11}$$

where $\boldsymbol{\mu}$ denotes Lagrange multipliers $\{\mu_i\}_{i=2}^n$ and $\{\mu_{ij}\}_{2\leq i<j\leq n}$ associated with the equality constraints in (4.10). Note that the last summation term of (4.11) is simplified by exploiting the symmetry between multipliers ($\mu_{ij} = \mu_{ji}$). Taking partial derivatives with respect to primal variables $(\widetilde{\mathbf{U}}, \widetilde{\boldsymbol{\lambda}})$ and equating to zero, we obtain following system of equations:

$$\frac{\partial\mathcal{J}_{\text{LAG}}(\widetilde{\mathbf{U}}, \widetilde{\boldsymbol{\lambda}}, \boldsymbol{\mu})}{\partial\lambda_j^{(l)}} = \sum_{l=1}^{L} k_l \left(-\frac{1}{\lambda_j^{(l)}} + \mathbf{u}_j^\intercal\mathbf{S}_l\mathbf{u}_j\right) = 0, \tag{4.12}$$

$$\frac{\partial \mathcal{J}_{\text{LAG}}(\widetilde{\mathbf{U}}, \widetilde{\boldsymbol{\lambda}}, \boldsymbol{\mu})}{\partial \mathbf{u}_j} = \sum_{l=1}^{L} 2k_l \lambda_j^{(l)} \mathbf{S}_l \mathbf{u}_j + 2\mu_j \mathbf{u}_j + 2 \sum_{2 \le i < j \le n} \mu_{ij} \mathbf{u}_i = \mathbf{0}. \tag{4.13}$$

Since k_l, $\lambda_j^{(l)}$ and $\mathbf{u}_j^{\intercal}\mathbf{S}_l\mathbf{u}_j$ are all nonnegative, (4.12) can be simplified as

$$1/\lambda_j^{(l)} = \mathbf{u}_j^{\intercal}\mathbf{S}_l\mathbf{u}_j. \tag{4.14}$$

By multiplying (4.13) with $\mathbf{u}_j^{\intercal}$ from the left, we get

$$\sum_{l=1}^{L} 2k_l \lambda_j^{(l)} \mathbf{u}_j^{\intercal}\mathbf{S}_l\mathbf{u}_j + 2\mu_j = 0. \tag{4.15}$$

Then, replacing $\mathbf{u}_j^{\intercal}\mathbf{S}_l\mathbf{u}_j$ with $1/\lambda_j^{(l)}$, as stated in (4.14), leads to

$$\mu_j = -\sum_{l=1}^{L} k_l \quad \text{for } j = 2, \dots, n. \tag{4.16}$$

By multiplying (4.13) with $(1/2)\mathbf{u}_h^{\intercal}$ from the left, we get

$$\sum_{l=1}^{L} k_l \lambda_j^{(l)} \mathbf{u}_h^{\intercal}\mathbf{S}_l\mathbf{u}_j + \mu_{hj} = 0 \quad j = 2, \dots, n, \; h \ne j. \tag{4.17}$$

Then, switching h and j indexes leads to

$$\sum_{l=1}^{L} k_l \lambda_h^{(l)} \mathbf{u}_j^{\intercal}\mathbf{S}_l\mathbf{u}_h + \mu_{jh} = 0 \quad h = 2, \dots, n, \; j \ne h. \tag{4.18}$$

Subtracting (4.18) from (4.17) results in the following equation,

$$\mathbf{u}_h^{\intercal} \left(\sum_{l=1}^{L} k_l (\lambda_j^{(l)} - \lambda_h^{(l)}) \mathbf{S}_l \right) \mathbf{u}_j = 0 \quad j, h = 2, \dots, n, \; j \ne h, \tag{4.19}$$

where $\lambda_j^{(l)} = 1/(\mathbf{u}_j^{\intercal}\mathbf{S}_l\mathbf{u}_j)$ and $\lambda_h^{(l)} = 1/(\mathbf{u}_h^{\intercal}\mathbf{S}_l\mathbf{u}_h)$ by (4.14). Based on (4.14) and (4.19), the necessary optimality conditions for the CGBT are

$$\mathbf{u}_h^{\intercal} \left(\sum_{l=1}^{L} k_l (\lambda_j^{(l)} - \lambda_h^{(l)}) \mathbf{S}_l \right) \mathbf{u}_j = 0 \quad \text{for } j, h = 2, \dots, n, \; j \ne h, \tag{4.20}$$

$$\mathbf{u}_j^{\intercal}\mathbf{S}_l\mathbf{u}_j = 1/\lambda_j^{(l)} \quad \text{for } j = 2, \dots, n, \tag{4.21}$$

both of which obviously hold if $\mathbf{U}$ jointly diagonalizes $\mathbf{S}_1, \dots, \mathbf{S}_L$, since each $\mathbf{u}_h^{\intercal}\mathbf{S}_l\mathbf{u}_j$ term would be equal to zero for $h \neq j$ and $1/\lambda_j^{(l)}$ for $h = j \geq 2$. However, for a given $\mathbf{U}$ that cannot jointly diagonalize the empirical covariances, the weighted combinations of $\mathbf{S}_1, \dots, \mathbf{S}_L$ in (4.20) measure the deviation from diagonality for each $\mathbf{u}_j$ and $\mathbf{u}_h$ pair, and it is used in our algorithm to update columns of $\mathbf{U}$ (discussed in Section 4.3). Note that the overall effect of $\mathbf{S}_l$ on the deviation depends not only on the number of data samples (k_l) but also on graph frequencies ($\lambda_j^{(l)}$ and $\lambda_h^{(l)}$) where the corresponding weight is $k_l(\lambda_j^{(l)} - \lambda_h^{(l)})$. The other condition in (4.21) is used in CGF estimation discussed next.

Common Graph Frequency (CGF) Estimation. To find the best CGF from multiple graph frequency matrices $\mathbf{\Lambda}_1, \dots, \mathbf{\Lambda}_L$, we introduce the following constraint on graph Laplacians,

$$\mathcal{C}_{\mathrm{CGF}} : \; \mathbf{L}_l = \widehat{\mathbf{U}} \mathbf{\Lambda} \widehat{\mathbf{U}}^{\mathsf{T}} \qquad \text{for } l = 1, 2, \dots, L \tag{4.22}$$

where $\widehat{\mathbf{U}}$ is an optimal solution to the CGBT problem in (4.10), and $\mathbf{\Lambda} = \operatorname{diag}([\lambda_1 \, \lambda_2 \, \cdots \, \lambda_n]^{\mathsf{T}})$ is the diagonal matrix we want to estimate. By using (4.22), we can rewrite $\mathcal{J}_{\mathrm{NLL}}(\mathbf{L}_1, \mathbf{L}_2, \dots, \mathbf{L}_L) = \mathcal{J}_{\mathrm{NLL}}(\mathbf{\Lambda})$ in (4.6) as,

$$\mathcal{J}_{\mathrm{NLL}}(\mathbf{\Lambda}) = \sum_{l=1}^{L} k_l \left(-\log|\mathbf{\Lambda}| + \operatorname{Tr}\left(\mathbf{\Lambda} \widehat{\mathbf{U}}^{\mathsf{T}} \mathbf{S}_l \widehat{\mathbf{U}} \right) \right). \tag{4.23}$$

Since the first eigenvalue of a CGL matrix is zero, we simplify the above equation as,

$$\mathcal{J}_{\mathrm{NLL}}(\lambda_2, \dots, \lambda_n) = \sum_{l=1}^{L} k_l \sum_{j=2}^{n} \left(-\log(\lambda_j) + \lambda_j \widehat{\mathbf{u}}_j^{\mathsf{T}} \mathbf{S}_l \widehat{\mathbf{u}}_j \right) \tag{4.24}$$

where $\widehat{\mathbf{u}}_j$ is the j-th column of $\widehat{\mathbf{U}}$ and $\lambda_j = (\mathbf{\Lambda})_{jj}$. By using the optimality condition $1/\lambda_j^{(l)} = \widehat{\mathbf{u}}_j^{\mathsf{T}} \mathbf{S}_l \widehat{\mathbf{u}}_j$ in (4.21), we get

$$\mathcal{J}_{\mathrm{NLL}}(\lambda_2, \dots, \lambda_n) = \sum_{l=1}^{L} k_l \sum_{j=2}^{n} \left(-\log(\lambda_j) + \lambda_j / \lambda_j^{(l)} \right) \tag{4.25}$$

whose minimization leads to the CGF problem in (4.3). By taking the derivative of (4.25) with respect to λ_j and equating it to zero, we obtain

$$\frac{\partial \mathcal{J}_{\mathrm{NLL}}(\lambda_2, \dots, \lambda_n)}{\partial \lambda_j} = \sum_{l=1}^{L} k_l \left(-\frac{1}{\lambda_j} + \frac{1}{\lambda_j^{(l)}} \right) = 0. \tag{4.26}$$

Since (4.25) is a convex function, (4.26) is the necessary and sufficient optimality condition for the CGF estimation, which can be compactly written as

$$\frac{1}{\lambda_j} = \frac{\sum_{l=1}^{L} k_l / \lambda_j^{(l)}}{\sum_{l=1}^{L} k_l} = \frac{\sum_{l=1}^{L} k_l / \lambda_j^{(l)}}{k_{\mathrm{sum}}} \qquad j = 2, 3, \dots, n. \tag{4.27}$$

Therefore, the optimal common graph frequency λ_j is a weighted sum of $1/\lambda_j^{(1)}, \dots, 1/\lambda_j^{(L)}$, where

Algorithm 4 Multigraph Combining Algorithm

```
Input: CGLs $\{\mathbf{L}_l\}_{l=1}^L$, positive weights $\{k_l\}_{l=1}^L$, initial matrix $\mathbf{U}_{\text{init}}$ and error tolerance $\epsilon$.
Output: Combined graph Laplacian $\widehat{\mathbf{L}}$
1: $\widehat{\mathbf{U}} \leftarrow \texttt{CGBT}(\{\mathbf{L}_l\}_{l=1}^L, \{k_l\}_{l=1}^L, \mathbf{U}_{\text{init}}, \epsilon)$    (i.e., apply Algorithm 5 to estimate a CGBT)
2: $\{s_j\}_{j=2}^n = \left\{\frac{1}{k_{\text{sum}}}\sum_{l=1}^L (k_l/\widehat{\mathbf{u}}_j^\intercal \mathbf{L}_l \widehat{\mathbf{u}}_j)\right\}_{j=2}^n$
3: $\lambda_1 = 0 \qquad \{\lambda_j\}_{j=2}^n = \{1/s_j\}_{j=2}^n$
4: $\{(\widehat{\mathbf{\Lambda}})_{jj}\}_{j=1}^n = \{\lambda_j\}_{j=1}^n$
5: $\widehat{\mathbf{L}} \leftarrow$ Solve the problem in (4.4) with $\widehat{\mathbf{S}} = \widehat{\mathbf{U}}\widehat{\mathbf{\Lambda}}^\dagger\widehat{\mathbf{U}}^\intercal$ to estimate $\widehat{\mathbf{L}}$.
6: return $\widehat{\mathbf{L}}$
```

the weights depend on the number of observations (e.g., k_l) used to calculate the sample covariances (e.g., $\mathbf{S}_l$).

Proposition 7. *If $\widehat{\mathbf{U}}, \widehat{\mathbf{\Lambda}}_1, \dots, \widehat{\mathbf{\Lambda}}_L$ are the optimal solutions of (4.2), then the CGF estimation problem in (4.3) has the following closed form solution, the diagonal matrix $\mathbf{\Lambda}$ with entries*

$$(\mathbf{\Lambda})_{11} = 0 \text{ and } (\mathbf{\Lambda})_{jj} = \frac{\sum_{l=1}^L k_l}{\sum_{l=1}^L k_l/\widehat{\lambda}_j^{(l)}} \text{ for } j = 2, 3, \dots, n, \tag{4.28}$$

where $\widehat{\lambda}_j^{(l)} = (\widehat{\mathbf{\Lambda}}_l)_{jj}$.

Proof. The proof follows from (4.23)–(4.27). □

Combinatorial graph Laplacian (CGL) Estimation. Assuming that CGBT and CGF matrices (i.e., $\widehat{\mathbf{U}}$ and $\widehat{\mathbf{\Lambda}}$) are optimal, the best CGL can be simply obtained by $\widehat{\mathbf{L}} = \widehat{\mathbf{U}}\widehat{\mathbf{\Lambda}}\widehat{\mathbf{U}}^\intercal$. However, estimating the optimal $\widehat{\mathbf{U}}$ is generally not feasible in practice, since the associated problem is non-convex so that an estimated $\widehat{\mathbf{U}}$ is typically a local optimal solution. Moreover, for a given set of graph Laplacian matrices, there may be no orthogonal matrix that satisfies $\mathcal{C}_{\text{CGBT}}$ (i.e., that jointly diagonalizes $\mathbf{L}_1, \dots, \mathbf{L}_L$). Thus, in general, $\widehat{\mathbf{L}} = \widehat{\mathbf{U}}\widehat{\mathbf{\Lambda}}\widehat{\mathbf{U}}^\intercal$ is not a graph Laplacian matrix. In order to find the closest CGL to the estimated $\widehat{\mathbf{U}}\widehat{\mathbf{\Lambda}}\widehat{\mathbf{U}}^\intercal$ term, we propose to solve the CGL problem in (4.4) where $\mathbf{S} = \widehat{\mathbf{U}}\widehat{\mathbf{\Lambda}}^\dagger\widehat{\mathbf{U}}^\intercal$ is analogous of the sample covariance in the original CGL problem discussed in Chapter 2.

4.3 Proposed Solution

In order to find the best CGL matrix $\widehat{\mathbf{L}}$ that combines L graph Laplacian matrices $\{\mathbf{L}_l\}_{l=1}^L$, we propose Algorithm 4, where we first solve the problem stated in (4.2) to find the best CGBT $\widehat{\mathbf{U}}$ (see line 1). We then estimate the CGF matrix (see lines 2–4) based on the condition stated in (4.27). Finally, the best CGL is found by solving the optimization problem in (4.4) (see line 5). Note that Algorithm 4 calls Algorithm 5 to find the best CGBT.

Algorithm 5 Common Graph-based Transform Algorithm (CGBT)

```
1: function CGBT($\{\mathbf{L}_l\}_{l=1}^L, \mathbf{U}_{\text{init}}, \{k_l\}_{l=1}^L, \epsilon$)
2:     $\mathbf{U} = \mathbf{U}_{\text{init}}$
3:     do
4:         $\mathbf{U}_{\text{pre}} = \mathbf{U}$
5:         for all $(h, j)$ pairs such that $2 \leq h, j \leq n$ and $h \neq j$ do
6:             $\mathbf{u}_h = (\mathbf{U})_{:,h} \quad \mathbf{u}_j = (\mathbf{U})_{:,j}$
7:             $\{\mathbf{Q}_l\}_{l=1}^L = \left\{ \begin{bmatrix} 1/(\mathbf{u}_h^\intercal \mathbf{L}_l \mathbf{u}_h) & 1/(\mathbf{u}_h^\intercal \mathbf{L}_l \mathbf{u}_j) \\ 1/(\mathbf{u}_j^\intercal \mathbf{L}_l \mathbf{u}_h) & 1/(\mathbf{u}_j^\intercal \mathbf{L}_l \mathbf{u}_j) \end{bmatrix} \right\}_{l=1}^L$
8:             $\mathbf{R} \leftarrow$ Best Rotation($\{\mathbf{Q}_l\}_{l=1}^L$, $\{k_l\}_{l=1}^L$, $\epsilon$)
9:             $\mathbf{V} = [\mathbf{u}_h \ \mathbf{u}_j]\mathbf{R}$
10:            $(\mathbf{U})_{:,h} = (\mathbf{V})_{:,1} \quad (\mathbf{U})_{:,j} = (\mathbf{V})_{:,2}$
11:        end for
12:    while $||\mathbf{U} - \mathbf{U}_{\text{pre}}||_F \geq \epsilon$ (stopping criterion)
13:    return $\mathbf{U}$
14: end function

15: function Best Rotation($\{\mathbf{Q}_l\}_{l=1}^L$, $\{k_l\}_{l=1}^L$, $\epsilon$)
16:    $\mathbf{R} = \mathbf{I}$
17:    do
18:        $\mathbf{R}_{\text{pre}} = \mathbf{R} \quad \mathbf{T} = \mathbf{0}$
19:        $\mathbf{r}_1 = (\mathbf{R})_{:,1} \quad \mathbf{r}_2 = (\mathbf{R})_{:,2}$
20:        for $l = 1$ to $L$ do
21:            $\delta_1^{(l)} = 1/(\mathbf{r}_1^\intercal \mathbf{Q}_l \mathbf{r}_1) \quad \delta_2^{(l)} = 1/(\mathbf{r}_2^\intercal \mathbf{Q}_l \mathbf{r}_2)$
22:            $\mathbf{T} = \mathbf{T} + k_l(\delta_1^{(l)} - \delta_2^{(l)})\mathbf{Q}_l$
23:        end for
24:        $\theta = \frac{1}{2}\text{arctan}\left(\frac{2\,(\mathbf{T})_{1,2}}{(\mathbf{T})_{1,1} - (\mathbf{T})_{2,2}}\right)$
25:        $\mathbf{R} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}$
26:    while $||\mathbf{R} - \mathbf{R}_{\text{pre}}||_F \geq \epsilon$ (stopping criterion)
27:    return $\mathbf{R}$
28: end function
```

In Algorithm 5, we present the CGBT algorithm proposed to solve the nonconvex optimization problem in (4.2). In a nutshell, for a given initial guess $\mathbf{U}_{\text{init}}$, weights $\{k_l\}_{l=1}^L$ and error tolerance ϵ, the CGBT algorithm iteratively solves the equation in (4.20) by pairwise updating the columns of $\mathbf{U}$ (see lines 6–10). Specifically, the `for` loop at line 5 iterates over all (h, j) pairs for $h, j = 2, \ldots, n$ and $h \neq j$ until the ϵ-convergence has been reached (see line 26). At each iterate, an optimized (2×2) rotation matrix $\mathbf{R}$ is used to update $\mathbf{u}_h$ and $\mathbf{u}_j$ (i.e., columns of $\mathbf{U}$) so that (4.20) is satisfied (see lines 8–10). To find the best rotation, another iterative procedure is used as depicted in Algorithm 5 between lines 15–28.

4.4 Results

In this section, we present our experimental results demonstrating the performance of our multigraph combining algorithm which solves (4.1) with $k_1 = k_2 = \cdots = k_L = 1$. For given L graph Laplacian matrices, $\mathbf{L}_1, \dots, \mathbf{L}_L$, we compare the proposed Algorithm 4 by benchmarking against the following two methods:

1. (*Averaging method*) The averaging method combines graphs as

$$\mathbf{L}_{\text{avg}} = \frac{1}{L}\sum_{l=1}^{L}\mathbf{L}_l, \tag{4.29}$$

2. (*CGL method*) The CGL problem in (4.4) is solved to combine graphs by setting the input matrix $\mathbf{S}$ as

$$\mathbf{S} = \frac{1}{\sum_{l=1}^{L} k_l}\sum_{l=1}^{L} k_l\,\mathbf{L}_l^{\dagger} = \frac{1}{L}\sum_{l=1}^{L}\mathbf{L}_l^{\dagger} \tag{4.30}$$

   since $k_1 = k_2 = \cdots = k_L = 1$. The solution (i.e., combined CGL) is denoted as $\mathbf{L}_{\text{cgl}}$.

We use two different metrics to measure the graph combining performance. The first metric is called *coding gain* which is a popular metric used in information theory and compression [90]. This metric is used to measure how well a designed CGBT $\mathbf{U}$ diagonalizes $\mathbf{L}^{\dagger}$ as follows,

$$\mathsf{cg}(\mathbf{U},\mathbf{L}) = \left(\frac{\prod_{i=1}^{n}(\mathbf{L}^{\dagger})_{ii}}{\prod_{i=2}^{n}(\mathbf{U}^{\mathsf{T}}\mathbf{L}^{\dagger}\mathbf{U})_{ii}}\right)^{1/n}. \tag{4.31}$$

The second metric we use is the graph Laplacian quadratic form in (1.5) to measure *average variation* of k signals ($\{\mathbf{y}_i\}_{i=1}^{k}$) with respect to a graph Laplacian $\mathbf{L}$ as,

$$\mathsf{av}(\{\mathbf{y}_i\}_{i=1}^{k},\mathbf{L}) = \frac{1}{k}\sum_{i=1}^{k}\mathbf{y}_i^{\mathsf{T}}\mathbf{L}\mathbf{y}_i. \tag{4.32}$$

In our experiments, for each input graph $\mathbf{L}_1, \dots, \mathbf{L}_L$ we randomly pick $k = 1000\times n$ samples from the distribution $\mathbf{x}_l \sim \mathsf{N}(\mathbf{0}, \mathbf{L}_l^{\dagger})$, and measure the average variation (av) with respect to combined graphs, $\widehat{\mathbf{L}}$, $\mathbf{L}_{\text{avg}}$ and $\mathbf{L}_{\text{cgl}}$. On the other hand, the coding gain is directly calculated for each input graph $\mathbf{L}_1, \dots, \mathbf{L}_L$ using GBTs $\widehat{\mathbf{U}}$, $\mathbf{U}_{\text{avg}}$ and $\mathbf{U}_{\text{cgl}}$ obtained from $\widehat{\mathbf{L}}$, $\mathbf{L}_{\text{avg}}$ and $\mathbf{L}_{\text{cgl}}$, respectively. Figures 4.1 and 4.2 illustrate input graphs with line and mesh structures and their combined graph results, $\widehat{\mathbf{L}}$ and $\mathbf{L}_{\text{avg}}$, respectively. Corresponding coding gain and average variation results are presented in Tables 4.1–4.4. According to these results, proposed graph combining algorithm outperforms both averaging and CGL methods by providing larger coding gain and lower average variation.

Table 4.1: Coding gain (cg) results for the line graphs in Figure 4.1

cg	$\mathbf{L}_1$	$\mathbf{L}_2$	$\mathbf{L}_3$	$\mathbf{L}_4$	$\mathbf{L}_5$	**Average**
$\widehat{\mathbf{U}}$	0.8298	0.8586	0.9066	0.8319	0.8812	**0.8616**
$\mathbf{U}_{\text{avg}}$	0.8216	0.8102	0.8160	0.8125	0.8061	0.8133
$\mathbf{U}_{\text{cgl}}$	0.8554	0.8591	0.8566	0.8565	0.8583	0.8572

Table 4.2: Average variation (av) results for the line graphs in Figure 4.1

av	$\mathbf{L}_1$	$\mathbf{L}_2$	$\mathbf{L}_3$	$\mathbf{L}_4$	$\mathbf{L}_5$	**Average**
$\widehat{\mathbf{L}}$	15.0063	14.5198	13.4222	15.0538	13.9226	**14.3850**
$\mathbf{L}_{\text{avg}}$	21.1885	21.0094	21.1003	21.1065	21.0882	21.0986
$\mathbf{L}_{\text{cgl}}$	15.0628	14.9598	15.0286	15.0051	15.0065	15.0126

Table 4.3: Coding gain (cg) results for the graphs in Figure 4.2

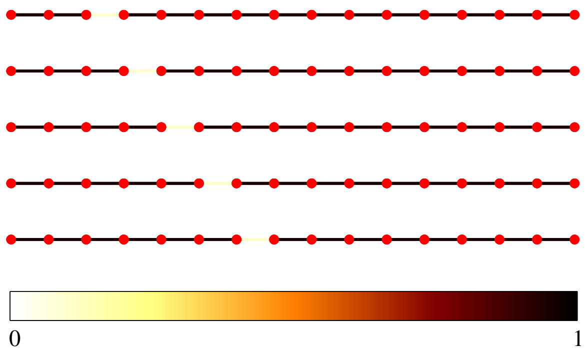
cg	$\mathbf{L}_1$	$\mathbf{L}_2$	$\mathbf{L}_3$	$\mathbf{L}_4$	**Average**
$\widehat{\mathbf{U}}$	0.9791	0.8981	0.9604	0.8799	**0.9294**
$\mathbf{U}_{\text{avg}}$	0.9050	0.8549	0.9521	0.8801	0.8981
$\mathbf{U}_{\text{cgl}}$	0.8673	0.8997	0.9310	0.9184	0.9041

Table 4.4: Average variation (av) results for the graphs in Figure 4.2

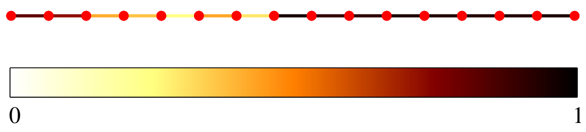
av	$\mathbf{L}_1$	$\mathbf{L}_2$	$\mathbf{L}_3$	$\mathbf{L}_4$	**Average**
$\widehat{\mathbf{L}}$	2.6637	3.4654	1.8542	3.8498	**2.9583**
$\mathbf{L}_{\text{avg}}$	5.6080	7.0555	3.6657	7.6316	5.9902
$\mathbf{L}_{\text{cgl}}$	3.9964	4.5259	2.6581	4.9227	4.0258

4.5 Conclusions

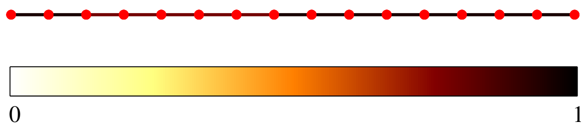
We have introduced a novel framework for graph combining by (i) introducing a new problem formulation with a maximum likelihood criterion and by (ii) proposing a solution involving common graph-based transform estimation and common graph frequency estimation. The experimental results have showed that the proposed multigraph combining method leads to a better model compared to the average graph model in terms of two well-known metrics, coding gain and quadratic Laplacian cost. The applications of the proposed framework to compression, such as designing aggregate models for clusters of signals/data, and to machine learning problems, such as clustering, are possible future directions on this work.



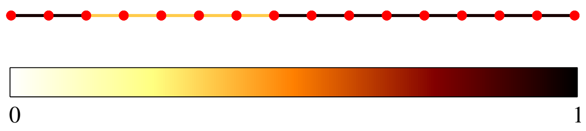
(a) Input graphs: $\mathbf{L}_1, \mathbf{L}_2, \mathbf{L}_3, \mathbf{L}_4, \mathbf{L}_5$



(b) Combined graph $\widehat{\mathbf{L}}$

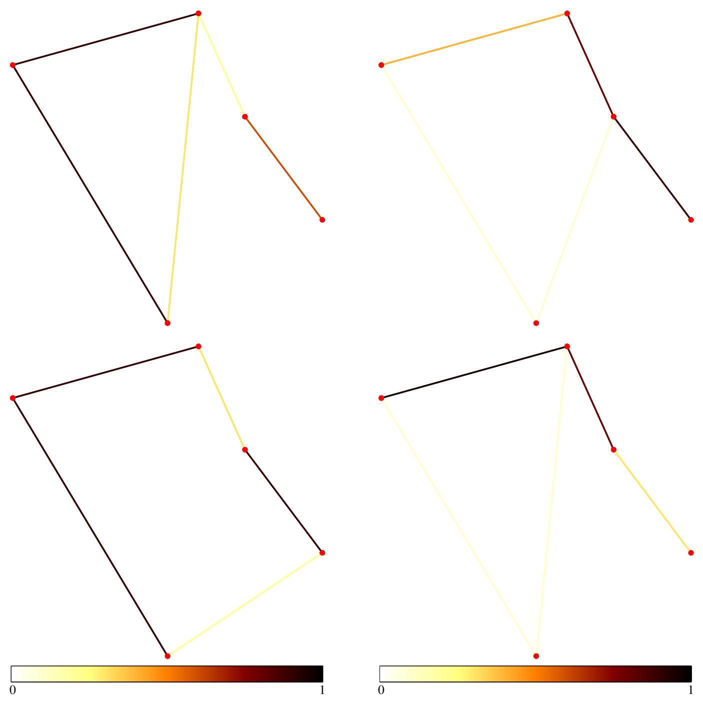


(c) Combined graph $\mathbf{L}_{\text{avg}}$

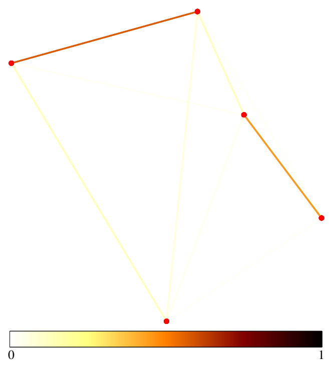


(d) Combined graph $\mathbf{L}_{\text{cgl}}$

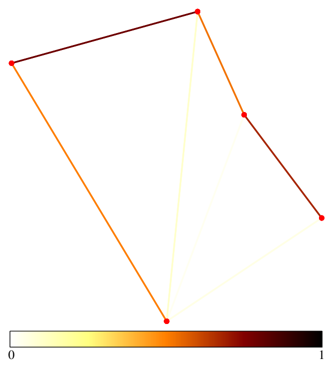
Figure 4.1: Combining $L = 5$ line graphs with $n = 16$ vertices. Edge weights are color coded between 0 and 1. Each input graph have only one weak edge weight equal to 0.1 while all other edges are weighted as 0.95.



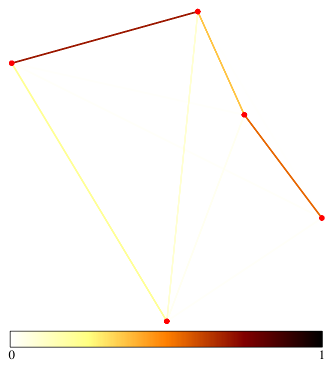
(a) Input graphs: $\mathbf{L}_1, \mathbf{L}_2, \mathbf{L}_3, \mathbf{L}_4$



(b) Combined graph $\widehat{\mathbf{L}}$



(c) Combined graph $\mathbf{L}_{\text{avg}}$



(d) Combined graph $\mathbf{L}_{\text{cgl}}$

Figure 4.2: Combining $L = 4$ mesh-like graphs with $n = 5$ vertices. Edge weights are color coded between 0 and 1.

Chapter 5

Graph-based Transforms for Video Coding

Predictive transform coding is a fundamental compression technique adopted in many block-based image and video compression systems, where block signals are initially predicted from a set of available (already coded) reference pixels, then the resulting residual block signals are transformed (generally by a linear transformation) to decorrelate residual pixel values for effective compression. After prediction and transformation steps, a typical image/video compression system applies quantization and entropy coding to convert transform coefficients into a stream of bits. Figure 5.1 illustrates a representative encoder-decoder architecture comprising three basic components, (i) prediction, (ii) transformation, (iii) quantization and entropy coding, which are also implemented in state-of-the-art compression standards such as JPEG [94], HEVC [95] and VP9 [96]. This chapter focuses mainly on the transformation component of video coding by developing graph-based techniques to design orthogonal transforms adapting to statistical characteristics of block residual signals. We also present theoretical justifications for the proposed techniques.

In predictive transform coding of video, the prediction is typically carried out by choosing among multiple intra and inter prediction modes to exploit spatial and temporal redundancies in block signals. On the other hand, for the transformation, generally a single transform such as the discrete cosine transform (DCT) is separably applied to rows and columns of each residual block. The main problem of using fixed block transforms is the implicit assumption that all residual blocks share the same statistical properties. However, residual blocks can have very diverse statistical characteristics depending on the video content and the prediction mode, as will be demonstrated by some of the experiments in this chapter. The latest video coding standard, HEVC [95], partially addresses this problem by additionally allowing the use of asymmetric discrete sine transform (ADST or DST-7)

Earlier versions of the work in this chapter are published in [8, 91, 91, 92, 7]. A journal version of this work will be submitted for publication subsequently [93].

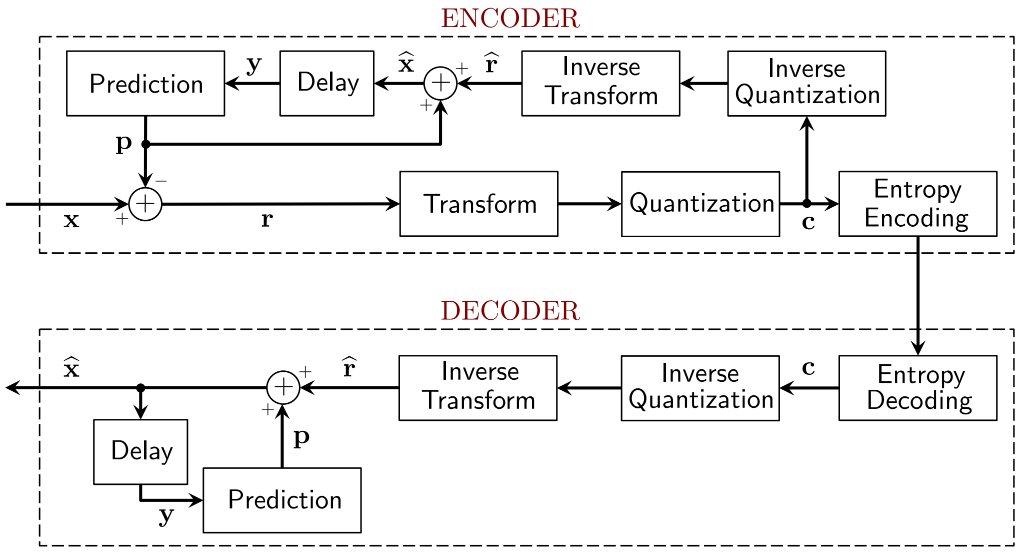


Figure 5.1: Building blocks of a typical video encoder and decoder consisting of three main steps, which are (i) prediction, (ii) transformation, (iii) quantization and entropy coding.

for small (4×4) intra predicted blocks [97]. Yet, it has been shown that better compression can be achieved by using data-driven transform designs that adapt to statistical properties of residual blocks [98, 99, 100, 101, 102, 103, 104, 105]. The majority of prior studies about transforms for video coding focus on developing transforms for intra predicted residuals. Ye and Karczewicz [98] propose the mode-dependent transform (MDT) scheme where a Karhunen-Loeve transform (KLT) is designed for each intra prediction mode. More recently in [99, 100], the MDT scheme is implemented on the HEVC standard, where a single KLT is trained for each intra prediction mode offered in HEVC. Moreover in [101, 102, 103, 104], authors demonstrate considerable coding gains outperforming the MDT method by using the rate-distortion optimized transformation (RDOT) scheme, which suggests designing multiple transforms for each prediction mode. From the predetermined set of transforms, the encoder selects a transform by minimizing a rate-distortion (RD) cost. Since the RDOT scheme allows encoders to flexibly select a transform on a per-block basis, it provides better adaptation to residual blocks with different statistical characteristics as compared to the MDT scheme. However, all of these methods rely on KLTs, which are constructed by the eigendecomposition of sample covariance matrices. In this work, we propose a graph-based modeling framework to design GBTs[a], where the models of interest are defined based on GMRFs whose inverse covariances are graph Laplacian matrices as discussed in Chapter 2. Specifically, in our framework, we propose two different approaches to construct graph Laplacian matrices based on data (i.e., video signals) under specific structural constraints. Our approaches can be interpreted as methods to regularize covariance matrices used in order to derive KLTs by introducing Laplacian and structural constraints, which

[a]GBT is formally defined in Definition 5 (see in Chapter 1).

potentially leads to a better (inverse) covariance estimation (i.e., better models to characterize signals/data) as discussed in Chapter 2.

The following two distinct methods are proposed to develop classes of GBTs for video coding, called GL-GBTs and EA-GBTs:

- *Graph learning for GBT (GL-GBT) design:* The GGL estimation problem (Problem 1 in Chapter 2) is used to estimate a GGL from training data, and the estimated graph Laplacian matrix is used to derive the GBT, called the GL-GBT. The proposed method allows us to impose structural constraints on the graph Laplacian in order to design graphs with desired connectivity. As the KLT, GL-GBT is learned from a sample covariance, but in addition, it incorporates Laplacian and structural constraints reflecting the inherent model assumptions about the video signal. The proposed graph learning approach can be used to design GBTs for MDT and RDOT schemes.

- *Edge-adaptive GBT (EA-GBT) design:* To adapt transforms for block signals with image edges[a], we develop edge-adaptive GBTs (EA-GBTs) which are designed on a per-block basis. These lead to a block-adaptive transform (BAT) scheme, where transforms are derived from graph Laplacians whose weights are modified based on image edges detected in a residual block.

From a statistical learning perspective, the main advantage of GL-GBT over KLT is that GL-GBT requires learning fewer model parameters to be obtained from training data, and thus potentially leads to a more robust transform allowing better compression for the block signals outside of the training dataset. In addition, EA-GBTs provide per-block transform adaptation that is not applicable to KLTs. However in practice, transform coding schemes such as RDOT and BAT require to encode additional side information, called transform signaling overhead, which is used by the decoder to identify the transforms used at the encoder. Thus, it is important for any transform coding scheme to consider the rate-distortion (RD) tradeoff [106]. For example, in RDOT scheme, the number of GL-GBTs/KLTs trained for each mode should be limited so that the designed transforms capture common block characteristics with low signaling overhead. Similarly, for EA-GBTs the encoder should make a decision for each block, i.e., whether to send the detected edge (graph) information or to use a fixed transform such as DCT. In our experiments, we comprehensively evaluate the performance of different transforms considering the rate-distortion tradeoff.

The main contributions of this chapter can be summarized as follows:

- The graph learning techniques proposed in Chapter 2 are used to develop separable and non-separable GBTs, called GBST and GBNT respectively, and their theoretical justifications for coding residual block signals modeled based on GMRFs are presented. In addition to the 1-D GMRF models used to design GBSTs for intra and inter predicted signals in our previous work

[a]We use the term *image edge* to distinguish edges in image/video signals with edges in graphs.

[8], a general 2-D GMRF model is presented for GBNTs, and the optimality of proposed GBTs is analyzed.

- EA-GBTs are applied for intra and inter predicted blocks, while our prior work in [92] focuses only on inter predicted blocks. In addition to the experimental results, we further derive some theoretical results and discuss the cases in which EA-GBTs are useful.

- We present graph-based interpretations of proposed models and statistical characteristics of signals in terms of entries of graph Laplacian matrices. Also, we show that different types of DCTs and DSTs, including the DCT-2 and DST-7 used in HEVC, are specific examples of GBTs.

In the literature, there are a few studies on model-based transform designs for video coding. Most related to our work, Han *et. al.* [97] present a single parameter 1-D GMRF and discuss the cases where ADST (DST-7) is optimal for coding intra predicted signals. Hu *et. al.* [107] extend this model by introducing another parameter to represent piecewise-smooth signals and use that to develop transforms for depth map coding. Although the models introduced in [97, 107] are analogous to our 1-D GMRF introduced for intra predicted signals, our model is defined using multiple parameters (i.e., weights of a line graph) so that it is more general than [97, 107]. In addition, [97] and [107] focus only on separable transforms and do not consider inter predicted signals. In [6, 108], the authors present a graph-based probabilistic framework for predictive video coding, and use it to justify the optimality of DCT. However, optimal graph/transform design is out of their scope. In our previous work [7], we present an extensive comparison of various instances of different graph learning problems for 2-D nonseparable image modeling. This chapter theoretically and empirically validates one of the conclusions in [7], which suggests the use of GBTs derived from GGL matrices. Moreover, several edge-adaptive transform approaches have been proposed. Shen *et. al.* [109] propose edge adaptive transforms (EAT) specifically for depth map compression. Although our proposed EA-GBTs adopt some basic concepts originally introduced in [109], our graph construction method is different (in terms of image edge detection) and provides better compression for residual signals. Hu *et. al.* [110] extends EATs for piecewise-smooth image compression and applies them for depth map coding, while our work focuses on encoding intra and inter predicted blocks.

This chapter is organized as follows. Section 5.1, introduces 1-D and 2-D GMRFs used for modeling the video signals and discusses graph-based interpretations. In Section 5.2, GL-GBT design problem is formulated as a graph Laplacian estimation problem, and solutions for optimal GBNT and GBST construction are proposed. Section 5.3 presents EA-GBTs. Graph-based interpretations of residual block signal characteristics are discussed in Section 5.4 by empirically validating the theoretical observations. The experimental results are presented in Section 5.5 and Section 5.6 draws concluding remarks.

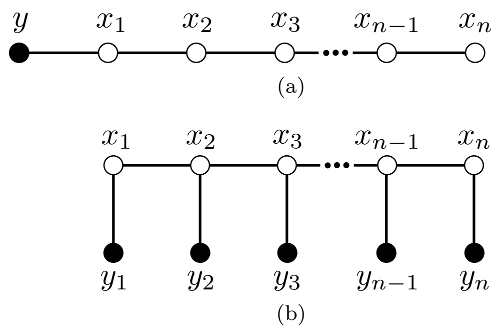


Figure 5.2: 1-D GMRF models for (a) intra and (b) inter predicted signals. Black filled vertices represent the reference pixels and unfilled vertices denote pixels to be predicted and then transform coded.

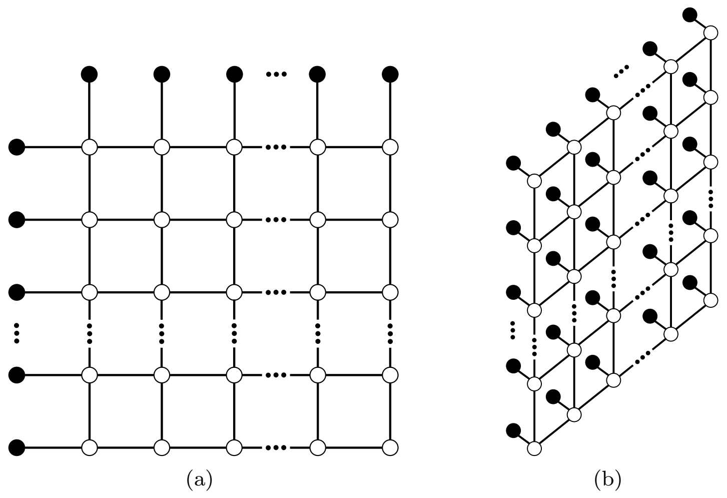


Figure 5.3: 2-D GMRF models for (a) intra and (b) inter predicted signals. Black filled vertices correspond to reference pixels obtained (a) from neighboring blocks and (b) from other frames via motion compensation. Unfilled vertices denote the pixels to be predicted and then transform coded.

5.1 Models for Video Block Signals

For modeling video block signals, we use Gaussian Markov random fields (GMRFs), which provide a probabilistic interpretation for our graph-based framework as presented in Chapter 2.

In statistical modeling of image/video signals, it is generally assumed that adjacent pixel values are positively correlated [4, 5]. The assumption is intuitively reasonable for video signals, since neighboring pixel values are often similar to each other due to spatial and temporal redundancy. With this general assumption, we propose attractive GMRFs to model intra/inter predicted video block signals, where Figures 5.2 and 5.3 illustrate the 1-D and 2-D GMRFs used to design separable and nonseparable GBTs (GBSTs and GBNTs), respectively. We formally define GBST and GBNT as follows.

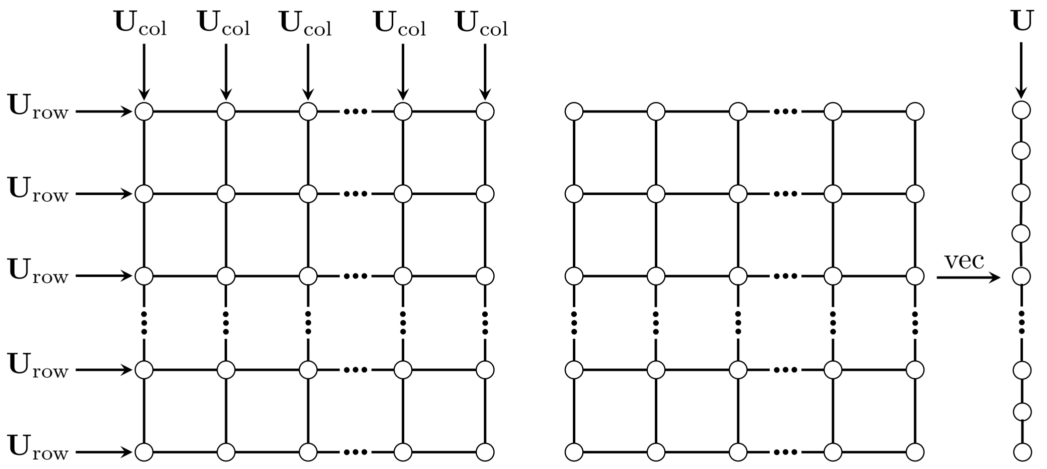


Figure 5.4: Separable and nonseparable transforms: (Left) For an $N{\times}N$ image/video block, separable transforms (e.g., GBSTs) are composed of two (possibly distinct) $N{\times}N$ transforms, $\mathbf{U}_{\mathrm{row}}$ and $\mathbf{U}_{\mathrm{col}}$, applied to rows and columns of the block. (Right) Nonseparable transforms (e.g., GBNTs) apply an $N^2{\times}N^2$ linear transformation using $\mathbf{U}$.

Definition 8 (Graph-based Separable Transform (GBST))**.** Let $\mathbf{U}_{\mathrm{row}}$ and $\mathbf{U}_{\mathrm{col}}$ be $N \times N$ GBTs associated with two line graphs with N vertices, then the GBST of $\mathbf{X}$ is

$$\widehat{\mathbf{X}} = \mathbf{U}_{\mathrm{col}}^{\intercal}\mathbf{X}\mathbf{U}_{\mathrm{row}}, \tag{5.1}$$

where $\mathbf{U}_{\mathrm{row}}$ and $\mathbf{U}_{\mathrm{col}}$ are applied to each row and each column of an $N \times N$ block signal $\mathbf{X}$, respectively.

Definition 9 (Graph-based Nonseparable Transform (GBNT))**.** Let $\mathbf{U}$ be an $N^2{\times}N^2$ GBT associated with a graph with N^2 vertices, then the GBNT of $N{\times}N$ block signal $\mathbf{X}$ is

$$\widehat{\mathbf{X}} = \mathrm{block}(\mathbf{U}^{\intercal}\mathrm{vec}(\mathbf{X})), \tag{5.2}$$

where $\mathbf{U}$ is applied on vectorized signal $\mathbf{x} = \mathrm{vec}(\mathbf{X})$, and the block operator restructures the signal back in block form.

Figure 5.4 shows separable and nonseparable transforms applied on a block signal.

In the rest of this section, we first present three different 1-D GMRFs for intra and inter predicted signals, as well as an edge model. Then, we introduce a 2-D GMRF generalizing these models.

5.1.1 1-D GMRF Models for Residual Signals

In order to model rows/columns of $N{\times}N$ block residual signals, we construct 1-D GMRFs based on first-order autoregressive (AR) processes. Specifically, depending on type of the prediction (i.e., intra/inter), a set of reference samples, denoted as $\mathbf{y}$, is used to predict $\mathbf{x} = [x_1\ x_2\ \cdots\ x_n]^\intercal$ which represents $n = N$ samples in a row/column of a block[a]. Assuming that the optimal causal (MMSE) predictor $\mathbf{p}$ is used for intra/inter prediction, the precision matrix of the resulting residual signal, $\mathbf{r} = \mathbf{x} - \mathbf{p}$, is then derived to define the 1-D GMRF of $\mathbf{r}$.

Intra predicted signal model. For modeling intra predicted signals as a 1-D GMRF, illustrated in Figure 5.2a, we formulate the following stochastic difference equations, which generalize the models in [97, 107] by allowing arbitrary ρ_i correlation parameters, which are fixed in [97, 107], for $i = 0, \dots, n-1$,

$$\begin{aligned} x_1 &= \rho_0(y + d) + e_1 \\ x_2 &= \rho_1 x_1 + e_2 \\ &\vdots \\ x_{n-1} &= \rho_{n-2} x_{n-2} + e_{n-1} \\ x_n &= \rho_{n-1} x_{n-1} + e_n \end{aligned} \tag{5.3}$$

where the reference sample y is used to predict n samples in $\mathbf{x} = [x_1\ x_2\ \cdots\ x_n]^\intercal$. The random variable $d \sim \mathsf{N}(0, \sigma_d^2)$ models the distortion due to compression in the reference sample y, and $e_i \sim \mathsf{N}(0, \sigma_e^2)$ is the noise in x_i with a fixed variance σ_e^2. The spatial correlation coefficients between samples are denoted by $\rho_0, \rho_1, \dots, \rho_{n-1}$, and we also assume that random variables d and e_i are independent for $i = 1, \dots, n$.

The relations in (5.3) can be written more compactly in vector form as $\mathbf{Q}\mathbf{x} = \mathbf{y}_1 + \mathbf{d}_1 + \mathbf{e}$ where $\mathbf{y}_1 = [(\rho_0 y)\ 0 \cdots 0]^\intercal$, $\mathbf{d}_1 = [(\rho_0 d)\ 0 \cdots 0]^\intercal$, $\mathbf{e} = [e_1\ e_2 \cdots e_n]^\intercal$ and

$$\mathbf{Q} = \begin{bmatrix} 1 & 0 & \cdots & \cdots & \cdots & 0 \\ -\rho_1 & 1 & 0 & & & \vdots \\ 0 & -\rho_2 & 1 & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & -\rho_{n-2} & 1 & 0 \\ 0 & \cdots & \cdots & 0 & -\rho_{n-1} & 1 \end{bmatrix}. \tag{5.4}$$

Since $\mathbf{x} = \mathbf{Q}^{-1}\mathbf{y}_1 + \mathbf{Q}^{-1}(\mathbf{d}_1 + \mathbf{e})$ where $\mathbf{p} = \mathbf{Q}^{-1}\mathbf{y}_1$ is the optimal prediction for $\mathbf{x}$, the resulting residual vector is $\mathbf{r} = \mathbf{x} - \mathbf{p}$, and its covariance matrix is

$$\mathbf{\Sigma_r} = \mathbf{Q}^{-1}\mathsf{E}\left[(\mathbf{e} + \mathbf{d}_1)(\mathbf{e} + \mathbf{d}_1)^\intercal\right](\mathbf{Q}^{-1})^\intercal. \tag{5.5}$$

[a]For 1-D models, the number of vertices (n) are equal to the number of pixels (N) in a row/column of an $N \times N$ block.

Inverting the covariance matrix gives us the precision matrix,

$$\begin{aligned}\boldsymbol{\Omega}_{\text{intra}} = {\boldsymbol{\Sigma}_{\mathbf{r}}}^{-1} &= \mathbf{Q}^{\intercal}\left(\mathsf{E}\left[\mathbf{e}\mathbf{e}^{\intercal} + \mathbf{d}_1\mathbf{d}_1^{\intercal}\right]\right)^{-1}\mathbf{Q} \\ &= \frac{1}{\sigma_e^2}\mathbf{Q}^{\intercal}\begin{bmatrix} \frac{1}{1+\beta_d} & 0 & \cdots & 0 \\ 0 & 1 & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \cdots & 0 & 1 \end{bmatrix}\mathbf{Q},\end{aligned} \tag{5.6}$$

which defines our 1-D GMRF model for the residual signal $\mathbf{r}$. This is explicitly stated in (5.9) where $\beta_d = \frac{(\rho_0\sigma_d)^2}{\sigma_e^2}$.

Inter predicted signal model. In modeling inter predicted signals, we have n reference samples, $y_1, \ldots, y_n$, used to predict n samples in $\mathbf{x} = [x_1\ x_2\ \cdots\ x_n]^{\intercal}$, as shown in Figure 5.2b. So, we derive the following difference equations by incorporating multiple reference samples as

$$\begin{aligned} x_1 &= \widetilde{\rho}_1(y_1 + d_1) + e_1 \\ x_2 &= \rho_1 x_1 + \widetilde{\rho}_2(y_2 + d_2) + e_2 \\ &\vdots \\ x_{n-1} &= \rho_{n-2}x_{n-2} + \widetilde{\rho}_{n-1}(y_{n-1} + d_{n-1}) + e_{n-1} \\ x_n &= \rho_{n-1}x_{n-1} + \widetilde{\rho}_n(y_n + d_n) + e_n \end{aligned} \tag{5.7}$$

where $d_i \sim \mathsf{N}(0, \sigma_{d_i}^2)$ denotes the distortion due to compression in the reference sample y_i and $e_i \sim \mathsf{N}(0, \sigma_e^2)$ is the noise in sample x_i. The random variables e_i and d_i are also assumed to be independent. In addition to the spatial correlation coefficients $\rho_1, \ldots, \rho_{n-1}$, our model includes temporal correlation coefficients, $\widetilde{\rho}_1, \ldots, \widetilde{\rho}_n$.

The recursive relations in (5.7) can also be written in vector form, that is $\mathbf{Q}\mathbf{x} = \mathbf{y}_2 + \mathbf{d}_2 + \mathbf{e}$ where $\mathbf{Q}$ is given in (5.4), $\mathbf{y}_2 = [\widetilde{\rho}_1 y_1\ \widetilde{\rho}_2 y_2 \cdots \widetilde{\rho}_n y_n]^{\intercal}$ and $\mathbf{d}_2 = [\widetilde{\rho}_1 d_1\ \widetilde{\rho}_2 d_2 \cdots \widetilde{\rho}_n d_n]^{\intercal}$. We write $\mathbf{x} = \mathbf{Q}^{-1}\mathbf{y}_2 + \mathbf{Q}^{-1}(\mathbf{d}_2 + \mathbf{e})$ where $\mathbf{p} = \mathbf{Q}^{-1}\mathbf{y}_2$ is the optimal prediction for $\mathbf{x}$. So, the resulting residual vector is $\mathbf{r} = \mathbf{x} - \mathbf{p} = \mathbf{Q}^{-1}(\mathbf{d}_2 + \mathbf{e})$ and its covariance is

$$\boldsymbol{\Sigma}_{\mathbf{r}} = \mathbf{Q}^{-1}\mathsf{E}\left[(\mathbf{e} + \mathbf{d}_2)(\mathbf{e} + \mathbf{d}_2)^{\intercal}\right](\mathbf{Q}^{-1})^{\intercal}. \tag{5.8}$$

$$\mathbf{\Omega}_{\text{intra}} = \frac{1}{\sigma_e^2}\begin{bmatrix} \frac{1}{1+\beta_d}+\rho_1^2 & -\rho_1 & 0 & \cdots & \cdots & 0 \\ -\rho_1 & 1+\rho_2^2 & -\rho_2 & 0 & & \vdots \\ 0 & -\rho_2 & 1+\rho_3^2 & -\rho_3 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & & \ddots & -\rho_{n-2} & 1+\rho_{n-1}^2 & -\rho_{n-1} \\ 0 & \cdots & \cdots & 0 & -\rho_{n-1} & 1 \end{bmatrix} \tag{5.9}$$

$$\mathbf{\Omega}_{\text{inter}} = \frac{1}{\sigma_e^2}\begin{bmatrix} \frac{1}{1+\gamma_1}+\frac{\rho_1^2}{1+\gamma_2} & \frac{-\rho_1}{1+\gamma_2} & 0 & \cdots & \cdots & 0 \\ \frac{-\rho_1}{1+\gamma_2} & \frac{1}{1+\gamma_2}+\frac{\rho_2^2}{1+\gamma_3} & \frac{-\rho_2}{1+\gamma_3} & \ddots & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \frac{-\rho_{n-2}}{1+\gamma_{n-1}} & 0 \\ \vdots & \ddots & \ddots & \frac{-\rho_{n-2}}{1+\gamma_{n-1}} & \frac{1}{1+\gamma_{n-1}}+\frac{\rho_{n-1}^2}{1+\gamma_n} & \frac{-\rho_{n-1}}{1+\gamma_n} \\ 0 & \cdots & \cdots & 0 & \frac{-\rho_{n-1}}{1+\gamma_n} & \frac{1}{1+\gamma_n} \end{bmatrix} \tag{5.10}$$

$$\mathbf{\Omega}_{\text{edge}} = \frac{1}{\sigma_e^2}\begin{bmatrix} 1+\rho_1^2 & -\rho_1 & 0 & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & 0 \\ -\rho_1 & 1+\rho_2^2 & -\rho_2 & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & -\rho_{n-2} & 1+\rho_{n-1}^2 & -\rho_{n-1} & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & -\rho_{i-1} & 1+\frac{\rho_i^2}{1+\alpha_t} & -\frac{\rho_i}{1+\alpha_t} & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & -\frac{\rho_i}{1+\alpha_t} & \rho_{i+1}^2+\frac{1}{1+\alpha_t} & -\rho_{i+1} & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & -\rho_{i+1} & 1+\rho_{i+2}^2 & -\rho_{i+2} & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & -\rho_{n-2} & 1+\rho_{n-1}^2 & -\rho_{n-1} \\ 0 & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & 0 & -\rho_{n-1} & 1 \end{bmatrix} \tag{5.11}$$

By inverting the covariance matrix $\mathbf{\Sigma_r}$, we obtain the precision matrix

$$\begin{aligned}\mathbf{\Omega}_{\text{inter}} = \mathbf{\Sigma}_{\mathbf{r}}^{-1} &= \mathbf{Q}^{\intercal} \left(\mathsf{E}\left[\mathbf{e}\mathbf{e}^{\intercal} + \mathbf{d}_2\mathbf{d}_2^{\intercal}\right]\right)^{-1} \mathbf{Q} \\ &= \frac{1}{\sigma_e^2}\mathbf{Q}^{\intercal} \begin{bmatrix} \frac{1}{1+\gamma_1} & 0 & \cdots & 0 \\ 0 & \frac{1}{1+\gamma_2} & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \cdots & 0 & \frac{1}{1+\gamma_n} \end{bmatrix} \mathbf{Q},\end{aligned} \tag{5.12}$$

where $\gamma_i = \frac{(\widetilde{\rho}_i \sigma_{d_i})^2}{\sigma_e^2}$ for $i=1,\dots,n$. The explicit form of the precision $\mathbf{\Omega}_{\text{inter}}$ is stated in (5.10).

Edge-based residual model. In this model, we extend intra/inter predicted signal models by introducing an image edge, which is modeled using a random variable $t \sim \mathsf{N}(0, {\sigma_t}^2)$ representing a sharp signal transition. For example, to extend (5.3) with an image edge at x_i, we have

$$\begin{aligned} x_1 &= \rho_0 y + e_1 \\ x_2 &= \rho_1 x_1 + e_2 \\ &\ \vdots \\ x_{i-1} &= \rho_{i-2} x_{i-2} + e_{i-1} \\ x_i &= \rho_{i-1} x_{i-1} + e_i + t \\ x_{i+1} &= \rho_i x_i + e_{i+1} \\ &\ \vdots \\ x_n &= \rho_{n-1} x_{n-1} + e_n \end{aligned} \tag{5.13}$$

where e_i is defined as in (5.3), and it is statistically independent of t. In vector form, we can write $\mathbf{x} = \mathbf{Q}^{-1}\mathbf{y}_1 + \mathbf{Q}^{-1}(\mathbf{e}+\mathbf{t})$ and $\mathbf{p} = \mathbf{Q}^{-1}\mathbf{y}_1$, where $\mathbf{Q}$ is as stated in (5.4), $\mathbf{y}_1 = [(\rho_0 y)\ 0 \cdots 0]^{\intercal}$, $\mathbf{e} = [e_1\ e_2 \cdots e_n]^{\intercal}$ and $\mathbf{t} = [0\ \cdots 0\ t\ 0\ \cdots 0]^{\intercal}$. Then the residual signal is $\mathbf{r} = \mathbf{x} - \mathbf{p} = \mathbf{Q}^{-1}(\mathbf{e}+\mathbf{t})$. Thus, the covariance of the residual $\mathbf{r}$ is

$$\mathbf{\Sigma_r} = \mathbf{Q}^{-1}\mathsf{E}\left[(\mathbf{e}+\mathbf{t})(\mathbf{e}+\mathbf{t})^{\intercal}\right](\mathbf{Q}^{-1})^{\intercal}, \tag{5.14}$$

and the corresponding precision matrix is

$$\begin{aligned}\boldsymbol{\Omega}_{\text{edge}} &= \boldsymbol{\Sigma}_{\mathbf{r}}^{-1} = \mathbf{Q}^{\intercal}\left(\mathsf{E}\left[\mathbf{e}\mathbf{e}^{\intercal} + \mathbf{t}\mathbf{t}^{\intercal}\right]\right)^{-1}\mathbf{Q} \\ &= \frac{1}{\sigma_e^2}\mathbf{Q}^{\intercal}\begin{bmatrix} 1 & 0 & \cdots & \cdots & \cdots & \cdots & 0 \\ 0 & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & 1 & 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & 0 & \frac{1}{1+\alpha_t} & 0 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & \cdots & \cdots & \cdots & 0 & 1 \end{bmatrix}\mathbf{Q},\end{aligned} \tag{5.15}$$

which is explicitly stated in (5.11) where $\alpha_t = \frac{{\sigma_t}^2}{\sigma_e^2}$. Although this model is derived for intra prediction, a similar edge-based model can be trivially constructed for inter prediction based on (5.7). Also, a special case of this model with $\rho_i = 1$ for $i = 1, \ldots, n$ is considered in [110].

Attractive 1-D GMRFs and DCTs/DSTs as GBTs. The 1-D GMRF models discussed above are attractive if $\rho_1, \rho_2, \ldots, \rho_{n-1}$ are nonnegative. In this case, the corresponding precision matrices in (5.9)–(5.11) are GGL matrices.

Moreover, DCT and DST can be considered as special cases of GBTs associated with line graphs (i.e., 1-D attractive GMRFs). The relation between different types of DCT and graph Laplacians are originally discussed in [111] where DCT-2 (which is the type of DCT used extensively in image/video compression) is shown to be equal to the GBT derived from a uniformly weighted line graph (i.e., $\rho_i = 1$ for $i = 1, \ldots, n$ in 1-D GMRFs presented above) with no self-loops (i.e., all vertex weights are zero). Thus, the corresponding graph Laplacian for the n-point DCT-2 is

$$\mathbf{L}_u = \begin{bmatrix} 1 & -1 & & & 0 \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ 0 & & & -1 & 1 \end{bmatrix} \tag{5.16}$$

which is a CGL matrix. In [8, 107], it has been shown that DST-7 (i.e., ADST used in HEVC standard) is GBT derived from the GGL $\mathbf{L} = \mathbf{L}_u + \mathbf{V}$ where $\mathbf{V} = \operatorname{diag}([1\ 0\ \cdots\ 0]^{\intercal})$, which includes a self-loop at vertex v_1 with weight $f_v(v_1) = 1$. Based on the results in [111, 112], various other types of n-point DCTs and DSTs can be characterized based on GGLs with different vertex weights at v_1 and

v_n. Table 5.1 summarizes the graphs corresponding to different types of DCTs and DSTs, which are GBTs derived from graph Laplacians of the form $\widetilde{\mathbf{L}} = \mathbf{L}_u + \widetilde{\mathbf{V}}$ where $\widetilde{\mathbf{V}} = \operatorname{diag}([f_v(v_1)\ 0\ \cdots\ 0\ f_v(v_n)]^\intercal)$.

Table 5.1: DCTs/DSTs corresponding to $\widetilde{\mathbf{L}}$ with different vertex weights.

Vertex weights	$f_v(v_1)=0$	$f_v(v_1)=1$	$f_v(v_1)=2$
$f_v(v_n)=0$	DCT-2	DST-7	DST-4
$f_v(v_n)=1$	DCT-8	DST-1	DST-6
$f_v(v_n)=2$	DCT-4	DST-5	DST-2

Moreover, we can justify the use of DCT-2 and DST-7 in practice by analyzing the 1-D intra and inter predicted models. For this purpose, we assume that parameters ρ_i for $i = 0, 1, \ldots, n-1$ approach to 1 (i.e., $\rho_i \to 1$), since video pixels are typically highly correlated in practice. Then, based on the precision matrices in (5.9) and (5.10), we can show that the optimal GBT leads to

- DCT-2 if $\sigma_d \gg \sigma_e$ for the intra predicted model (if intra prediction performance is bad),
- DST-7 if $\sigma_d \ll \sigma_e$ for the intra predicted model (if intra prediction performance is good),
- DCT-2 if $\gamma_1 = \cdots = \gamma_n$ for the inter predicted model (if inter prediction performance is similar across pixels).

5.1.2 2-D GMRF Model for Residual Signals

By extending the 1-D models discussed above, we introduce a general 2-D GMRF model for intra and inter predicted $N \times N$ block signals, depicted in Figure 5.3. For this purpose, we derive the precision matrix of the residual signal obtained by predicting the signal $\mathbf{x} = [x_1\ x_2\ \cdots\ x_n]^\intercal$ with $n = N^2$ from n_p reference samples in $\mathbf{y} = [y_1\ y_2\ \cdots\ y_{n_p}]^\intercal$ (i.e., predicting unfilled vertices using black filled vertices in Figure 5.3). In our model, $\mathbf{x}$ and $\mathbf{y}$ are zero-mean and jointly Gaussian with respect to the following attractive 2-D GMRF:

$$\mathsf{p}([\begin{smallmatrix}\mathbf{x}\\ \mathbf{y}\end{smallmatrix}] \,|\, \boldsymbol{\Omega}) = \frac{1}{(2\pi)^{n/2}|\boldsymbol{\Omega}|^{-1/2}} \exp\left(-\frac{1}{2} [\begin{smallmatrix}\mathbf{x}\\ \mathbf{y}\end{smallmatrix}]^\intercal \boldsymbol{\Omega} [\begin{smallmatrix}\mathbf{x}\\ \mathbf{y}\end{smallmatrix}]\right). \tag{5.17}$$

where the precision matrix $\boldsymbol{\Omega}$ and the covariance matrix $\boldsymbol{\Sigma} = \boldsymbol{\Omega}^{-1}$ are partitioned as

$$\boldsymbol{\Omega} = \begin{bmatrix} \boldsymbol{\Omega}_{\mathbf{x}} & \boldsymbol{\Omega}_{\mathbf{xy}} \\ \boldsymbol{\Omega}_{\mathbf{yx}} & \boldsymbol{\Omega}_{\mathbf{y}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\Sigma}_{\mathbf{x}} & \boldsymbol{\Sigma}_{\mathbf{xy}} \\ \boldsymbol{\Sigma}_{\mathbf{yx}} & \boldsymbol{\Sigma}_{\mathbf{y}} \end{bmatrix}^{-1} = \boldsymbol{\Sigma}^{-1}. \tag{5.18}$$

Irrespective of the type of prediction (intra/inter), the optimal (MMSE) prediction of $\mathbf{x}$ from the reference samples in $\mathbf{y}$ is

$$\mathbf{p} = \mathsf{E}[\mathbf{x}|\mathbf{y}] = \boldsymbol{\Sigma}_{\mathbf{xy}}\boldsymbol{\Sigma}_{\mathbf{y}}^{-1}\mathbf{y} = -\boldsymbol{\Omega}_{\mathbf{x}}^{-1}\boldsymbol{\Omega}_{\mathbf{xy}}\mathbf{y}, \tag{5.19}$$

and the resulting residual vector is $\mathbf{r} = \mathbf{x} - \mathbf{p}$ with covariance

$$\begin{aligned}\boldsymbol{\Sigma}_{\mathbf{r}} = \boldsymbol{\Sigma}_{\mathbf{x}|\mathbf{y}} &= \mathsf{E}[\mathbf{r}\mathbf{r}^{\intercal}] = \mathsf{E}[(\mathbf{x}-\mathbf{p})(\mathbf{x}-\mathbf{p})^{\intercal}] \\ &= \mathsf{E}[\mathbf{x}\mathbf{x}^{\intercal} + \mathbf{p}\mathbf{p}^{\intercal} - 2\mathbf{x}\mathbf{p}^{\intercal}] \\ &= \boldsymbol{\Sigma}_{\mathbf{x}} + \boldsymbol{\Sigma}_{\mathbf{xy}}\boldsymbol{\Sigma}_{\mathbf{y}}^{-1}\boldsymbol{\Sigma}_{\mathbf{yx}} - 2\boldsymbol{\Sigma}_{\mathbf{xy}}\boldsymbol{\Sigma}_{\mathbf{y}}^{-1}\boldsymbol{\Sigma}_{\mathbf{yx}} \\ &= \boldsymbol{\Sigma}_{\mathbf{x}} - \boldsymbol{\Sigma}_{\mathbf{xy}}\boldsymbol{\Sigma}_{\mathbf{y}}^{-1}\boldsymbol{\Sigma}_{\mathbf{yx}}.\end{aligned} \tag{5.20}$$

By the matrix inversion lemma [65], the precision matrix of the residual $\mathbf{r}$ is shown to be equal to $\boldsymbol{\Omega}_{\mathbf{x}}$, that is the submatrix in (5.18),

$$\boldsymbol{\Sigma}_{\mathbf{r}}^{-1} = (\boldsymbol{\Sigma}_{\mathbf{x}} - \boldsymbol{\Sigma}_{\mathbf{xy}}\boldsymbol{\Sigma}_{\mathbf{y}}^{-1}\boldsymbol{\Sigma}_{\mathbf{yx}})^{-1} = \boldsymbol{\Omega}_{\mathbf{x}}. \tag{5.21}$$

Since we also have $\boldsymbol{\Sigma}_{\mathbf{x}} = (\boldsymbol{\Omega}_{\mathbf{x}} - \boldsymbol{\Omega}_{\mathbf{xy}}\boldsymbol{\Omega}_{\mathbf{y}}^{-1}\boldsymbol{\Omega}_{\mathbf{yx}})^{-1}$ by [65], the desired precision matrix can also be written as

$$\boldsymbol{\Omega}_{\text{residual}} = \boldsymbol{\Sigma}_{\mathbf{r}}^{-1} = \boldsymbol{\Omega}_{\mathbf{x}} = \boldsymbol{\Sigma}_{\mathbf{x}}^{-1} + \boldsymbol{\Omega}_{\mathbf{xy}}\boldsymbol{\Omega}_{\mathbf{y}}^{-1}\boldsymbol{\Omega}_{\mathbf{yx}}. \tag{5.22}$$

Proposition 8. *Let the signals $\mathbf{x} \in \mathbb{R}^n$ and $\mathbf{y} \in \mathbb{R}^{n_p}$ be distributed based on the attractive GMRF model in (5.17) with precision matrix $\boldsymbol{\Omega}$. If the residual signal $\mathbf{r}$ is estimated by minimum mean square error (MMSE) prediction of $\mathbf{x}$ from $\mathbf{y}$ (i.e, $\mathbf{r} = \mathbf{x} - \mathsf{E}[\mathbf{x}|\mathbf{y}]$), the residual signal $\mathbf{r}$ is distributed as an attractive GMRF whose precision matrix is a generalized graph Laplacian (i.e., $\boldsymbol{\Omega}_{\text{residual}}$ in (5.22)).*

Proof. The proof follows from equations (5.17)–(5.22). □

Proposition 8 also applies to the 1-D models in (5.3), (5.7) and (5.13) which are special cases of (5.17) with precision matrices as stated in (5.9)–(5.11).

5.1.3 Interpretation of Graph Weights for Predictive Coding

For graph-based interpretation of our models, we first present two different formulations of GMRFs, and then investigate the effect of graph weights based on these two forms.

GMRFs in Predictive Form. Let $\mathbf{x} = [x_1\, x_2\, \cdots\, x_n]^{\intercal}$ be a random vector drawn from a GMRF whose precision matrix is $\boldsymbol{\Omega}$, so that the entries of the error vector $\mathbf{e}$ are obtained by the optimal MMSE prediction,

$$e_i = x_i - \sum_{j \in \mathcal{S}\setminus\{i\}} g_{ij} x_j \qquad \text{for } i = 1, \dots, n \tag{5.23}$$

where g_{ij} denotes the optimal MMSE prediction coefficient such that $g_{ii} = 0$ for $i = 1, \dots, n$, and $\mathcal{S} = \{1, \dots, n\}$ is the index set for $\mathbf{x}$. Letting the variance of random variable e_i be

$$\sigma_i^2 = \mathsf{E}\left[e_i^2 \,|\, (\mathbf{x})_{\mathcal{S}\setminus\{i\}}\right], \tag{5.24}$$

we write the following conditional distribution

$$p(x_i|x_j \text{ for } i \neq j) = \frac{1}{\sqrt{2\pi\sigma_i^2}}\exp\left(-\frac{1}{2\sigma_i^2}\left(x_i - \sum_{j\in\mathcal{S}\setminus\{i\}} g_{ij}x_j\right)^2\right) \tag{5.25}$$

from which the entries of the precision matrix can be written in terms of g_{ij} and σ_i^2 as follows:

$$(\mathbf{\Omega})_{ij} = \begin{cases} -\frac{1}{\sigma_i^2}g_{ij} & \text{if } i \neq j \\ \frac{1}{\sigma_i^2} & \text{if } i = j \end{cases}. \tag{5.26}$$

More compactly, the precision matrix $\mathbf{\Omega}$ is given by

$$\mathbf{\Omega} = \mathbf{\Lambda}_\sigma^{-1}(\mathbf{I} - \mathbf{G}) \tag{5.27}$$

where $\mathbf{\Lambda}_\sigma$ is the diagonal matrix whose entries are $(\mathbf{\Lambda}_\sigma)_{ii} = \sigma_i^2$ for $i = 1, \dots, n$, and $\mathbf{G}$ is the prediction coefficient matrix whose entries are $(\mathbf{G})_{ij} = g_{ij}$, so that in matrix form

$$\mathbf{G} = \mathbf{I} - \mathbf{\Lambda}_\sigma\mathbf{\Omega}. \tag{5.28}$$

which is generally not a symmetric matrix, while $\mathbf{\Omega}$ is symmetric by definition. Since $(\mathbf{\Lambda}_\sigma)_{ii} = 1/(\mathbf{\Omega})_{ii}$, the diagonal entries of $\mathbf{G}$ are zero (i.e., $(\mathbf{G})_{ii} = 0$ for all i) by (5.28). Specifically for attractive GMRFs, $(\mathbf{G})_{ij}$ are nonnegative for all i and j.

GMRFs in Laplacian Quadratic Form. Based on Proposition 8, the distribution of residual signals, denoted as $\mathbf{r}$, is defined by the following GMRF whose precision matrix $\mathbf{\Omega}$ is a GGL matrix $\mathbf{L}$,

$$p(\mathbf{r}|\mathbf{L}) = \frac{1}{(2\pi)^{n/2}|\mathbf{L}|^{-1/2}}\exp\left(-\frac{1}{2}\mathbf{r}^\intercal\mathbf{L}\mathbf{r}\right), \tag{5.29}$$

where the quadratic term in the exponent can be decomposed in terms of graph weights as stated in (1.5)

$$\mathbf{r}^\intercal\mathbf{L}\mathbf{r} = \sum_{i=1}^{n}(\mathbf{V})_{ii}\, r_i^2 + \sum_{(i,j)\in\mathcal{I}}(\mathbf{W})_{ij}\,(r_i - r_j)^2 \tag{5.30}$$

where $\mathbf{r} = [r_1 \cdots r_n]^\intercal$, $(\mathbf{W})_{ij} = -(\mathbf{L})_{ij}$, $(\mathbf{V})_{ii} = \sum_{j=1}^{n}(\mathbf{L})_{ij}$ and $\mathcal{I} = \{(i,j)\,|\,(v_i,v_j)\in\mathcal{E}\}$ is the set of index pairs of all vertices associated with the edge set $\mathcal{E}$.

Interpretation of Graph Weights. Based on the Laplacian quadratic form given in (5.29) and (5.30), it is obvious that the distribution of the residual signal $\mathbf{r}$ depends on edge weights ($\mathbf{W}$) and vertex weights ($\mathbf{V}$) where

- assigning larger (resp. smaller) edge weights (e.g., $(\mathbf{W})_{ij}$) increases the probability of having a smaller (resp. larger) squared difference between corresponding residual pixel values (e.g.,

r_i and r_j),

- assigning larger (resp. smaller) vertex weights (e.g., $(\mathbf{V})_{ii}$) increases the probability of pixel values (e.g., r_i) with smaller (resp. larger) magnitude.

In addition, the characterization of the precision matrix (i.e., $\mathbf{L}$) in terms of g_{ij} and σ_i^2 in (5.26) reveals the interplay between prediction and graph weights. Specifically, assuming that the prediction coefficients, g_{ij} for $j = 1, \dots, n$, are given, the variance of the prediction error at vertex v_i, σ_i^2, gets smaller (resp. larger) as either one or both of the following statements are satisfied,

- degree at vertex v_i, $(\mathbf{D})_{ii}$, gets larger (resp. smaller)
- vertex weight $(\mathbf{V})_{ii}$ gets larger (resp. smaller)

which are based on $(\mathbf{L})_{ii} = (\mathbf{D})_{ii} + (\mathbf{V})_{ii} = 1/\sigma_i^2$ by (5.26). Since the degree matrix $\mathbf{D}$ is defined using $\mathbf{W}$, a good prediction may lead to increase in both edge and vertex weights. For the case of video coding, the contribution of edges and vertices depends mainly on statistics of the block signal and the type of the prediction. We experimentally investigate the effect of edge and vertex weights in Section 5.4.

5.2 Graph Learning for Graph-based Transform Design

5.2.1 Graph Learning: Generalized Graph Laplacian Estimation

As justified in Proposition 8, the residual signal $\mathbf{r} \in \mathbb{R}^n$ is modeled as an attractive GMRF, $\mathbf{r} \sim \mathsf{N}(\mathbf{0}, \mathbf{\Sigma} = \mathbf{L}^{\dagger})$, whose precision matrix is a GGL denoted by $\mathbf{L}$. To find the best GGL from a set of residual signals $\{\mathbf{r}_1, \dots, \mathbf{r}_k\}$ in a maximum likelihood sense, we solve the GGL estimation problem (i.e., Problem 1 in Chapter 2) without the ℓ_1-regularization, which is explicitly stated as follows:

$$\begin{aligned} \underset{\mathbf{L} \succeq 0}{\text{minimize}} \quad & \operatorname{Tr}(\mathbf{LS}) - \operatorname{logdet}(\mathbf{L}) \\ \text{subject to} \quad & (\mathbf{L})_{ij} \le 0 \ \text{ if } (\mathbf{A})_{ij} = 1 \\ & (\mathbf{L})_{ij} = 0 \ \text{ if } (\mathbf{A})_{ij} = 0 \end{aligned} \tag{5.31}$$

where $\mathbf{S} = \frac{1}{k}\sum_{i=1}^{k} \mathbf{r}_i \mathbf{r}_i^{\mathsf{T}}$ is the sample covariance of residual signals, and $\mathbf{A}$ is the connectivity matrix representing the structure of the graph (i.e., set of graph edges). In order to optimally solve (5.31), we employ Algorithm 1 discussed in Chapter 2.

5.2.2 Graph-based Transform Design

To design separable and nonseparable GBTs, we first solve instances of (5.31), which is denoted as $\mathsf{GGL}(\mathbf{S}, \mathbf{A})$. Then, the optimized GGL matrices are used to derive GBTs.

Graph-based Separable Transforms (GBST). For the GBST design, we solve two instances of (5.31) to optimize two separate line graphs used to derive $\mathbf{U}_{\text{row}}$ and $\mathbf{U}_{\text{col}}$ in (5.1). Since we wish to design a separable transform, each line graph can be optimized independently[a]. Thus, our basic goal is finding the best line graph pair based on sample covariance matrices $\mathbf{S}_{\text{row}}$ and $\mathbf{S}_{\text{col}}$ created from rows and columns of residual block signals. For $N \times N$ residual blocks, the proposed GBST construction has following steps:

1. Create the connectivity matrix $\mathbf{A}_{\text{line}}$ representing a line graph structure with $n = N$ vertices as in Figure 5.2.

2. Train two $N \times N$ sample covariances, $\mathbf{S}_{\text{row}}$ and $\mathbf{S}_{\text{col}}$, from N rows and N columns of residual blocks in the dataset.

3. Solve instances of the problem in (5.31), $\texttt{GGL}(\mathbf{S}_{\text{row}}, \mathbf{A}_{\text{line}})$ and $\texttt{GGL}(\mathbf{S}_{\text{col}}, \mathbf{A}_{\text{line}})$, by using Algorithm 1 in Chapter 2 to learn generalized graph Laplacian matrices $\mathbf{L}_{\text{row}}$ and $\mathbf{L}_{\text{col}}$, respectively.

4. Perform eigendecomposition on $\mathbf{L}_{\text{row}}$ and $\mathbf{L}_{\text{col}}$ to obtain GBTs, $\mathbf{U}_{\text{row}}$ and $\mathbf{U}_{\text{col}}$, which define the GBST as in (5.1).

Graph-based Nonseparable Transforms (GBNT). Similarly, for $N \times N$ residual block signals, we propose following steps to design a GBNT:

1. Create the connectivity matrix $\mathbf{A}$ based on a desired graph structure. For example, $\mathbf{A}$ can represent a grid graph with $n = N^2$ vertices as in Figure 5.3.

2. Train $N^2 \times N^2$ sample covariance $\mathbf{S}$ using residual block signals in the dataset (after vectorizing the block signals).

3. Solve the problem $\texttt{GGL}(\mathbf{S}, \mathbf{A})$ by using Algorithm 1 in Chapter 2 to estimate the generalized graph Laplacian matrix $\mathbf{L}$.

4. Perform eigendecomposition on $\mathbf{L}$ to obtain the $N^2 \times N^2$ GBNT, $\mathbf{U}$ defined in (5.2).

5.2.3 Optimality of Graph-based Transforms

It has been shown that KLT is optimal for transform coding of jointly Gaussian sources in terms of mean-square error (MSE) criterion under high bitrate assumptions [115, 116, 117]. Since the GMRF models discussed in Section 5.1 lead to jointly Gaussian distributions, the corresponding KLTs are optimal in theory. However, in practice, a KLT is obtained by eigendecomposition of the associated sample covariance, which has to be estimated from a training dataset where (i) data samples may not closely follow the model assumptions (e.g., GMRFs), and (ii) the number of data samples may

[a]Alternatively, joint optimization of the transforms associated with rows and columns has been recently proposed in [113, 114].

not be sufficient to accurately recover the parameters. As a result, the sample covariance may lead a poor estimation of the actual model parameters. From the statistical learning theory perspective [118, 119], the advantage of our proposed GL-GBT over KLT is that KLT requires learning $O(n^2)$ model parameters while GL-GBT only needs $O(n)$. Therefore, our graph learning approach provides better *generalization* in learning the signal model by taking into account variance-bias tradeoff. This advantage can also be justified based on the following error bounds characterized in [120, 73]. Assuming that k residual blocks are used for calculating the sample covariance $\mathbf{S}$, under general set of assumptions, the error bound for estimating $\mathbf{\Sigma}$ with $\mathbf{S}$ derived in [120] is

$$||\mathbf{\Sigma}-\mathbf{S}||_F = O\left(\sqrt{\frac{n^2\log(n)}{k}}\right), \tag{5.32}$$

while estimating the precision matrix $\mathbf{\Omega}$ by using the proposed graph learning approach leads to the following bound shown in [73],

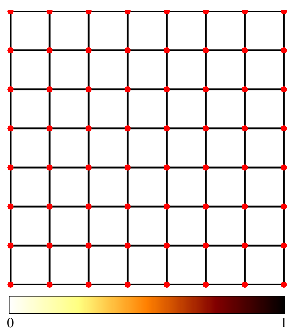
$$||\mathbf{\Omega}-\mathbf{L}||_F = O\left(\sqrt{\frac{n\log(n)}{k}}\right), \tag{5.33}$$

where $\mathbf{L}$ denotes the estimated GGL. Thus, in terms of the worst-case errors (based on Frobenius norm), the proposed method provides a better model estimation as compared to the estimation based on the sample covariance. Section 5.5 empirically justifies the advantage of GL-GBT against KLT.

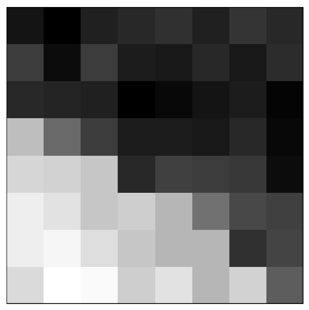
5.3 Edge-Adaptive Graph-based Transforms

The optimality of GL-GBTs relies on the assumption that the residual signal characteristics are the same across different blocks. However, in practice, video blocks often exhibit complex image edge structures that can degrade the coding performance when the transforms are designed from average statistics without any classification based on image edges. In order to achieve better compression for video signals with image edges, we propose edge-adaptive GBTs (EA-GBTs), which are designed on a per-block basis by constructing a graph-based model whose parameters are determined based on the salient image edges in each residual block. In this work, EA-GBTs are defined by using the following two parameters:

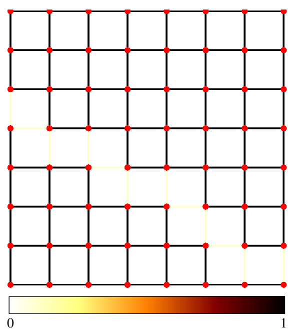
- $\mathcal{E}_{\text{edge}}$ is a subset of edges indicating the locations (of image edges) where the difference between pixel values are large,
- $s_{\text{edge}} \geq 1$ is a parameter representing the level of difference between pixel values (i.e., sharpness of the image edge).



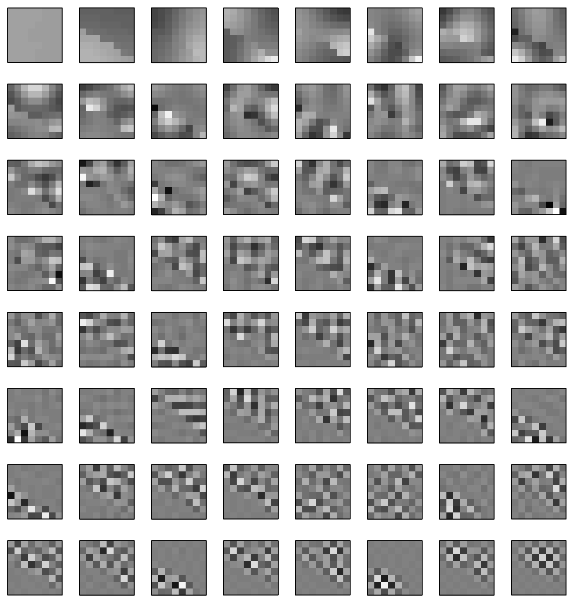
(a) Initial graph



(b) Residual block signal



(c) Constructed graph



(d) Constructed EA-GBT basis patches

Figure 5.5: An illustration of graph construction for a given 8×8 residual block signal where $w_c = 1$ and $w_e = 0.1$, and the corresponding GBT. The basis patches adapt to the characteristics of the residual block. The order of basis patches is in row-major ordering.

5.3.1 EA-GBT Construction

To design EA-GBTs on a per-block basis, we construct a graph for a given $N \times N$ block as follows:

1. We create a nearest neighbor (4-connected) graph $\mathcal{G}$ with edge weights all equal to a fixed constant w_c.
2. On a given residual block, we apply *Prewitt* operator to calculate gradient in vertical and horizontal direction, and then detect image edges based on thresholding on the gradient values.

3. Based on the detected image edges, we determine the set of edges $\mathcal{E}_{\text{edge}}$ in graph $\mathcal{G}$ and then modify the corresponding edge weights by setting them to $w_e = w_c/s_{\text{edge}}$.

4. The resulting graph is used to construct the associated GBT is constructed as in Definition 5.

Figure 5.5 illustrates an example of graph construction for the 8×8 block signal (Figure 5.5b) and the resulting EA-GBT (Figure 5.5d), which captures the characteristic of the residual block. Note that EA-GBT construction can also be viewed as a classification procedure where each residual block (such as in Figure 5.5b) is assigned to a class of signals modeled based on a graph (such as in Figure 5.5c) determined through image edge detection.

The above procedure results in nonseparable EA-GBTs, yet it can be trivially modified for separable EA-GBTs by designing line graphs based on given image edge locations. In terms of the 1-D edge-based residual model in (5.13), the proposed EA-GBT corresponds to the case where ρ_i is fixed to a positive constant w_c for $i = 1, \dots, n$, the parameter s_{edge} is $s_{\text{edge}} = (1 + \alpha_t)$, and the edge set $\mathcal{E}_{\text{edge}}$ represents a single image edge located between vertices v_i and v_{i+1}. By letting $w_c = \sigma_e^2 = 1$, the resulting precision matrix in (5.11) becomes

$$\boldsymbol{\Omega}_{\text{edge}} = \begin{bmatrix} 2 & -1 & 0 & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & 0 \\ -1 & 2 & -1 & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & -1 & 2 & -1 & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & -1 & 1+w_e & -w_e & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & -w_e & 1+w_e & -1 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & -1 & 2 & -1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & -1 & 2 & -1 \\ 0 & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & 0 & -1 & 1 \end{bmatrix} \tag{5.34}$$

where $w_e = w_c/s_{\text{edge}} = 1/(1+\alpha_t)$ is the weight of the edge connecting vertices (pixels) v_{i-1} and v_i. In order to evaluate the effect of the parameter s_{edge} on pixel values, we generate $k = 50000$ samples $\mathbf{r}_1, \dots, \mathbf{r}_k$ based on an attractive GMRF whose precision matrix is (5.34) with $w_e = 1/s_{\text{edge}}$ (i.e., a 1-D edge-based model with an image edge between v_{i-1} and v_i), and the generated samples are scaled to have 8-bit intensity values within the range $[0, 255]$. Then, the average intensity difference

Table 5.2: Average absolute valued intensity differences (m_{diff}) between pixels connected by the edge with weight $w_e = w_c/s_{\text{edge}}$ for various sharpness parameters (s_{edge}).

$s_{\text{edge}} = w_c/w_e$	1	2	5	10	20	40	100	200
m_{diff}	3.99	5.64	8.93	12.65	17.89	25.30	40.13	56.26

between the pixels attached to vertices v_{i-1} and v_i is estimated as

$$m_{\text{diff}} = \frac{1}{k}\sum_{j=1}^{k} |(\mathbf{r}_j)_{i-1} - (\mathbf{r}_j)_i|. \tag{5.35}$$

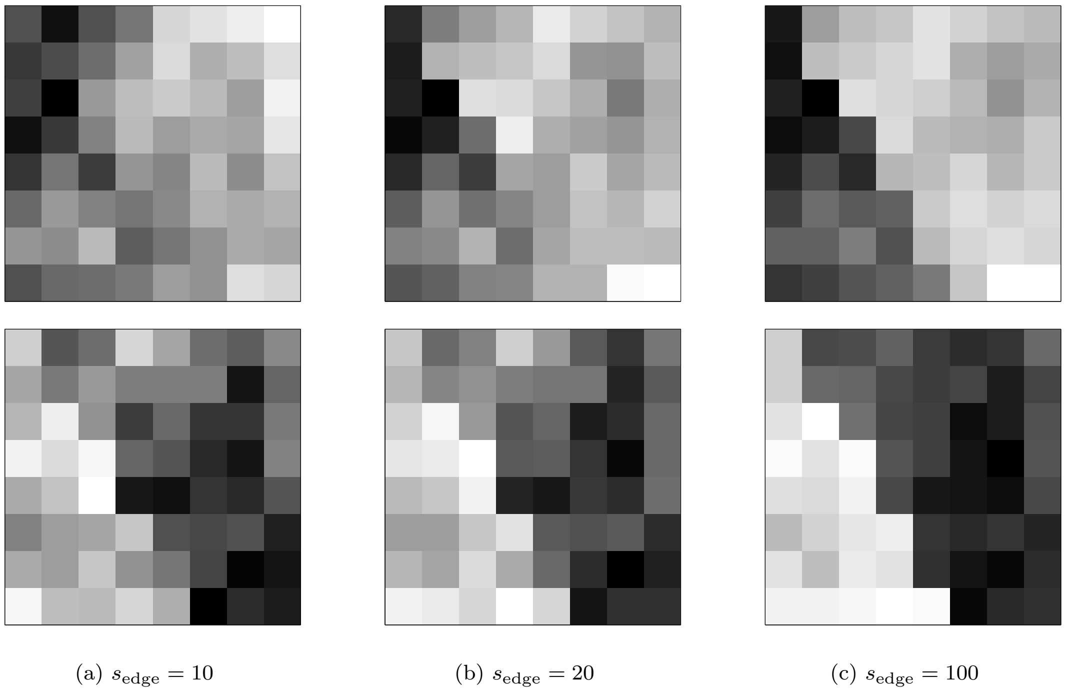
Table 5.2 shows the estimated m_{diff} values for different s_{edge} parameters, where m_{diff} is approximately equal to 4 for $s_{\text{edge}} = 1$ and increases with s_{edge}. For example, m_{diff} approximately triples when $s_{\text{edge}} = 10$. In other words, the degree/sharpness of discontinuity between the pixels at vertices v_{i-1} and v_i becomes three times as large. Moreover, for 2-D GMRF models, we can make a similar analysis based on (5.25), (5.26) and (5.30). Assuming that the 2-D GMRF model has image edges around a pixel x_i, then the prediction from neighboring pixels (as in (5.23)) would be naturally difficult where the prediction coefficients (g_{ij}) have smaller values and the error variance (σ_i^2) is larger. Since $(\mathbf{\Omega})_{ij} = -g_{ij}/\sigma_i^2$ by (5.26), this leads to smaller edge weights ($w_{ij} = -(\mathbf{\Omega})_{ij}$) and also a smaller degree (i.e., $(\mathbf{\Omega})_{ii}$). Figure 5.6 illustrates the effect of s_{edge} by showing random block samples generated from 2-D GMRF models having an image edge with different s_{edge} parameters, where the image edge (i.e., transition) becomes more visually apparent as s_{edge} increases.

Although EA-GBTs can provide efficient coding for transform coefficients, the overall coding performance may not be sufficient due to signaling overhead of graph information to the decoder side, especially if multiple graph weight parameters are used to model image edges (e.g., sharpness). By taking the rate-distortion tradeoff into account, we propose to use a single parameter (s_{edge}) to model image edges, and for efficient representation of detected edges (i.e., $\mathcal{E}_{\text{edge}}$), we employ the state-of-the-art binary edge-map codec called arithmetic edge encoder (AEC) [121].

5.3.2 Theoretical Justification for EA-GBTs

We present a theoretical justification for advantage of EA-GBTs over KLTs. In our analysis, we assume that the signal follows our 1-D edge-based residual signal model in (5.13), where the location of an image edge l is uniformly distributed as

$$\mathsf{P}(l = j) = \begin{cases} \frac{1}{N-1} & \text{for } j = 1, \ldots, N-1 \\ 0 & \text{otherwise} \end{cases} \tag{5.36}$$



(a) $s_{\text{edge}} = 10$ (b) $s_{\text{edge}} = 20$ (c) $s_{\text{edge}} = 100$

Figure 5.6: Two random block samples obtained from three 2-D GMRF models having an image edge at a fixed location with three different sharpness parameters ($s_{\text{edge}} = \{10, 20, 100\}$). A larger s_{edge} leads to a sharper transition across the image edge.

where N is the number of pixels (vertices) on the line graph. This construction leads to a Gaussian mixture distribution obtained by $N-1$ edge-based signal models, that is

$$\mathsf{p}(\mathbf{x}) = \sum_{j=1}^{N-1} \mathsf{P}(l=j)\, \mathsf{N}(\mathbf{0}, \mathbf{\Sigma}_j) \tag{5.37}$$

where $\mathbf{\Sigma}_j$ denotes the covariance of the model in (5.13) with an edge between pixels v_j and v_{j+1} (i.e., the transition variable t is located at v_{j+1}). In general, $\mathbf{x}$ does not have a jointly Gaussian distribution, so the KLT obtained from the covariance of $\mathbf{x}$ (which implicitly performs a second-order approximation of the distribution) can be suboptimal in MSE sense [122]. On the other hand, the proposed EA-GBT removes the uncertainty coming from the random variable l by detecting the location of the image edge in pixel (vertex) domain, and then constructing a GBT based on the detected image edge. Yet, EA-GBT requires allocating additional bits to represent the image edge (side) information, while KLT only allocates bits for coding transform coefficients.

To demonstrate the rate-distortion tradeoff between KLT and EA-GBT based coding schemes, we set the model parameters of (5.13) so that the corresponding precision matrix has the form in (5.34) with different edge locations. Based on the classical rate-distortion theory results with

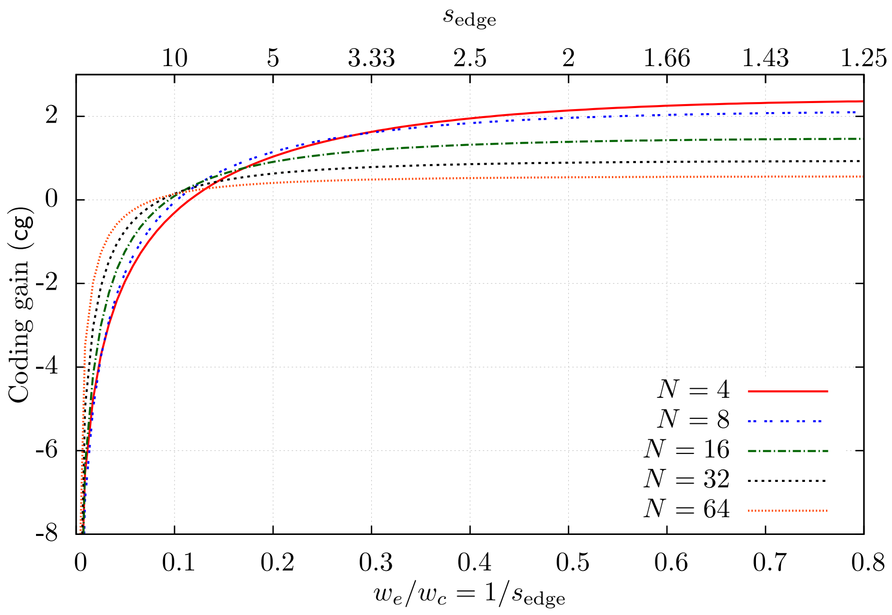


Figure 5.7: Coding gain (cg) versus s_{edge} for block sizes with $N = 4, 8, 16, 32, 64$. EA-GBT provides better coding gain (i.e., cg is negative) when s_{edge} is larger than 10 across different block sizes.

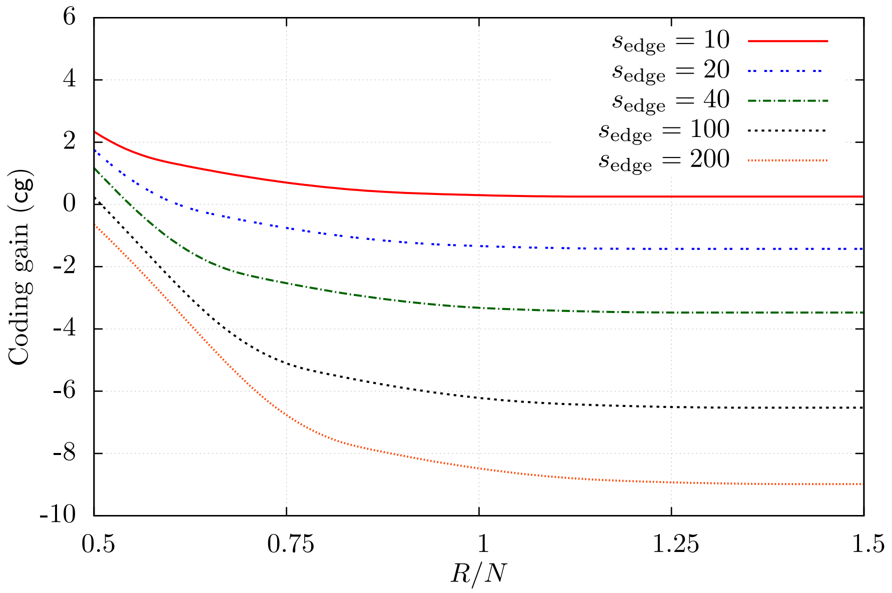


Figure 5.8: Coding gain (cg) versus bits per pixel (R/N) for different edge sharpness parameters $s_{\text{edge}} = 10, 20, 40, 100, 200$. EA-GBT provides better coding gain (i.e., cg is negative) if s_{edge} is larger than 10 for different block sizes.

high-bitrate assumptions the distortion can be written as a function of bitrate [115, 116, 117],

$$D(\bar{R}) = \frac{N}{12} 2^{2\bar{H}_d} 2^{-2\bar{R}} \tag{5.38}$$

with

$$\bar{R} = \frac{R}{N} \quad \text{and} \quad \bar{H}_d = \frac{1}{N}\sum_{i=1}^{N} H_d((\mathbf{c})_i) \tag{5.39}$$

where R denotes the total bitrate allocated to code transform coefficients in $\mathbf{c} = \mathbf{U}^\mathsf{T}\mathbf{x}$, and $H_d((\mathbf{c})_i)$ is the differential entropy of transform coefficient $(\mathbf{c})_i$. For EA-GBT, the R is allocated to code both transform coefficients ($R^{\text{coeff}}_{\text{EA-GBT}}$) and side information ($R^{\text{edge}}$), so we have

$$R = R^{\text{coeff}}_{\text{EA-GBT}} + R^{\text{edge}} = R^{\text{coeff}}_{\text{EA-GBT}} + \log_2(N-1) \tag{5.40}$$

while for KLT, the bitrate is allocated only to code transform coefficients ($R^{\text{coeff}}_{\text{KLT}}$), so that $R = R^{\text{coeff}}_{\text{KLT}}$. Figure 5.7 depicts the coding gain of EA-GBT over KLT for different sharpness parameters (s_{edge}) in terms of the following metric, called coding gain,

$$\mathsf{cg}(D_{\text{EA-GBT}}, D_{\text{KLT}}) = 10\log_{10}\left(\frac{D_{\text{EA-GBT}}}{D_{\text{KLT}}}\right) \tag{5.41}$$

where $D_{\text{EA-GBT}}$ and D_{KLT} denote distortion levels measured at high-bitrate regime for EA-GBT and KLT, respectively. EA-GBT provides better compression for negative cg values in Figure 5.7 which appear when the sharpness of edges s_{edge} is large (e.g., $s_{\text{edge}} > 10$). Moreover, the coding loss is negligible even if the image edge is not sharp (e.g., $s_{\text{edge}} < 2$) for large block sizes (e.g., $N = 64$).

Note that the distortion function in (5.38) is derived based on high-bitrate assumptions. To characterize rate-distortion tradeoff for different bitrates, we employ the reverse water-filling technique [123, 115] by varying the parameter θ to obtain rate and distortion measures as follows

$$R(D) = \sum_{i=1}^{N} \frac{1}{2}\log_2\left(\frac{\lambda_i}{D_i}\right) \tag{5.42}$$

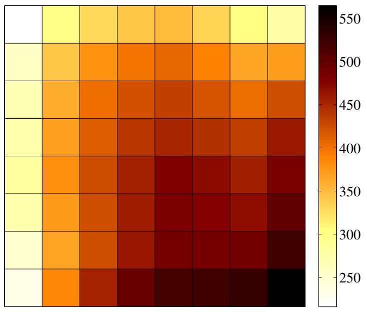
where λ_i is the i-th eigenvalue of the signal covariance, and

$$D_i = \begin{cases} \lambda_i & \text{if } \lambda_i \geq \theta \\ \theta & \text{if } \theta < \lambda_i \end{cases} \tag{5.43}$$

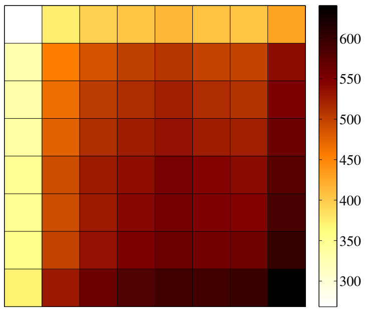
so that $D = \sum_{i=1}^{N} D_i$.

Figure 5.8 illustrates the coding gain formulated in (5.41) achieved at different bitrates, where each curve correspond to a different s_{edge} parameter. Similar to Figure 5.7, EA-GBT leads to a better compression if the sharpness of edges, s_{edge}, is large (e.g., $s_{\text{edge}} > 10$ for $R/N > 0.6$)[a]. At low-bitrates (e.g., $R/N = 0.5$), EA-GBT can perform worse than KLT for $s_{\text{edge}} = 20, 40$, yet EA-GBT outperforms as bitrate increases.

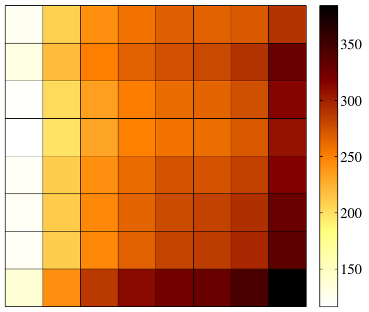
5.4 Residual Block Signal Characteristics and Graph-based Models



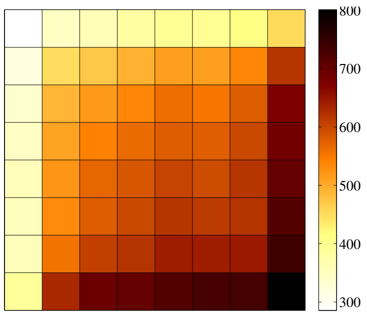
(a) Intra (Planar)



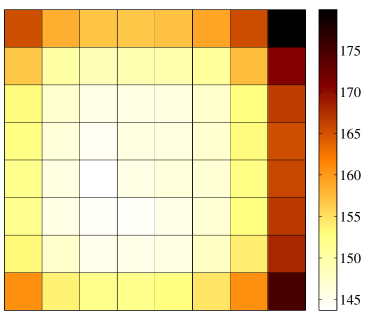
(b) Intra (DC)



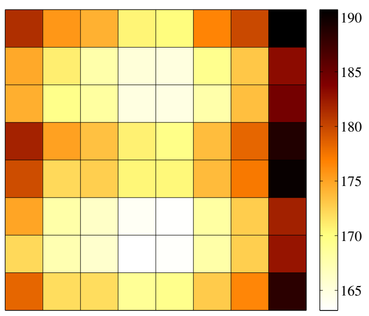
(c) Intra (Horizontal)



(d) Intra (Diagonal)



(e) Inter ($2N \times 2N$)



(f) Inter ($N \times 2N$)

Figure 5.9: Sample variances of residual signals corresponding to different prediction modes. Each square corresponds to a sample variance of pixel values. Darker colors represent larger sample variances.

In this section, we discuss statistical characteristics of intra and inter predicted residual blocks, and empirically justify our theoretical analysis and observations in Section 5.1. Our empirical results are based on residual signals obtained by encoding 5 different video sequences (*City*, *Crew*, *Harbour*, *Soccer* and *Parkrun*) using the HEVC reference software (HM-14) [95] at 4 different QP paremeters ($QP = \{22, 27, 32, 37\}$). Although the HEVC standard does not implement optimal MMSE prediction (which is the main assumption in Section 5.1), it includes 35 intra and 8 inter prediction modes,

[a]Table 5.2 shows the effect of different sharpness parameters (s_{edge}) on pixel values.

which provide reasonably good prediction for different classes of block signals.

Naturally, residual blocks have different statistical characteristics depending on the type of prediction and the prediction mode. Figure 5.9 illustrates the sample variances of 8×8 residual blocks for different prediction modes[a] and leads us to the following observations:

- As expected, inter predicted blocks have smaller sample variance (energy) across pixels compared to intra predicted blocks, because inter prediction provides better prediction with larger number of reference pixels as shown in Figure 5.3.
- In intra prediction, sample variances are generally larger at the bottom-right part of residual blocks, since reference pixels are located at the top and left of a block where the prediction is relatively better. This holds specifically for planar, DC and diagonal modes using pixels on both top and left as references for prediction.
- For some angular modes including intra horizontal/vertical mode, only left/top pixels are used as references. In such cases the sample variance gets larger as distance from reference pixels increases. Figure 5.9c illustrates sample variances corresponding to the horizontal mode.
- In inter prediction, sample variances are larger around the boundaries and corners of the residual blocks mainly because of occlusions leading to partial mismatches between reference and predicted blocks.
- In inter prediction, PU partitions lead to larger residual energy around the partition boundaries as shown in Figure 5.9f corresponding to horizontal PU partitioning $(N \times 2N)$.

Figures 5.10–5.15 depict the graphs (i.e., normalized edge and vertex weights) optimized for residuals obtained by 4 intra and 2 inter prediction modes, whose corresponding sample variances are shown in Figure 5.9. In the figures, grid and line graphs are estimated based on the GBNT and GBST construction procedures, which involve solving the GGL estimation problem in (5.31) as discussed in Chapter 2 in detail. Inspection of Figures 5.10–5.15 leads to following observations, which validate our theoretical analysis and justify the interpretation of model parameters in terms of graph weights discussed in Section 5.1:

- Irrespective of the prediction mode/type, vertex (self-loop) weights tend to be larger for the pixels that are connected to reference pixels. Specifically, in intra prediction, graphs have larger vertex weights for vertices (pixels) located at the top and/or left boundaries of the block (Figures 5.10-5.13), while the vertex weights are approximately uniform across vertices in inter prediction (Figures 5.14 and 5.15).
- In intra prediction, the grid and line graphs associated with planar and DC modes are similar in structure (Figures 5.10 and 5.11), where their edge weights decrease as the distance of edges

[a]For 4×4 and 16×16 residual blocks, the structure of sample variances are quite similar to the ones in Figure 5.9.

to the reference pixels increase. Also, vertex weights are larger for the pixels located at top and left boundaries, since planar and DC modes use reference pixels from the both sides (top and left). These observations indicate that the prediction performance gradually decreases for pixels increasingly farther away from the reference pixels.

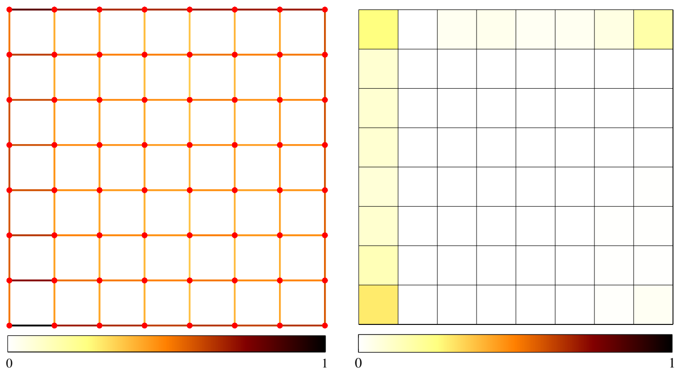
- For intra prediction with horizontal mode (Figure 5.12), the grid graph has larger vertex weights at the left boundary of the block. This is because the prediction only uses reference pixels on the left side of the block. Due to this reason, the line graph associated to rows has a large self-loop at the first pixel, while the other line graph has no dominant vertex weights. However, grid and line graphs for the diagonal mode (Figure 5.13), are more similar to the ones for planar and DC modes, since the diagonal mode also uses the references from both top and left sides.

- For inter prediction with PU mode $2N \times 2N$ (do not perform any partitioning), the graph weights (both vertex and edge weights) are approximately uniform across the different edges and vertices (Figure 5.14). This shows that the prediction performance is similar at different locations (pixels). In contrast, the graphs for the PU mode $N \times 2N$ (performs horizontal partitioning) leads to smaller edge weights around the PU partitioning (Figure 5.15). In the grid graph, we observe smaller weights between the partitioned vertices. Among line graphs, only the line graph designed for columns has a small weight in the middle, as expected.

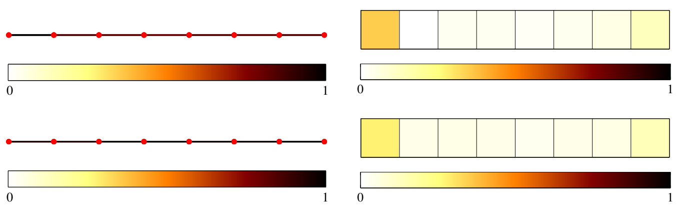
5.5 Experimental Results

5.5.1 Experimental Setup

In our experiments, we generate two residual block datasets, one for training and the other for testing. The residual blocks are collected by using HEVC reference software (HM version 14) [95]. For the training dataset, residual blocks are obtained by encoding 5 video sequences, *City*, *Crew*, *Harbour*, *Soccer* and *Parkrun*, and for the test dataset, we use 5 different video sequences, *BasketballDrill*, *BQMall*, *Mobcal*, *Shields* and *Cactus*. The sequences are encoded using 4 different quantization parameters, $QP = \{22, 27, 32, 37\}$, and transform block sizes are restricted to 4×4, 8×8 and 16×16. In both datasets, residual blocks are classified based on the side information provided by the HEVC encoder [95]. Specifically, intra predicted blocks are classified based on 35 intra prediction modes offered in HEVC. Similarly, inter predicted blocks are classified into 7 different classes using prediction unit (PU) partitions, such that 2 square PU partitions are grouped as one class and other 6 rectangular PU partitions determine other classes. Hence, we have $35+7 = 42$ classes in total. For each class and block size, the optimal GBST, GBNT and KLT are designed using the residual blocks in training dataset, while EA-GBTs are constructed based on the detected image edges. The details of transform the construction are discussed in Sections 5.2 and 5.3.

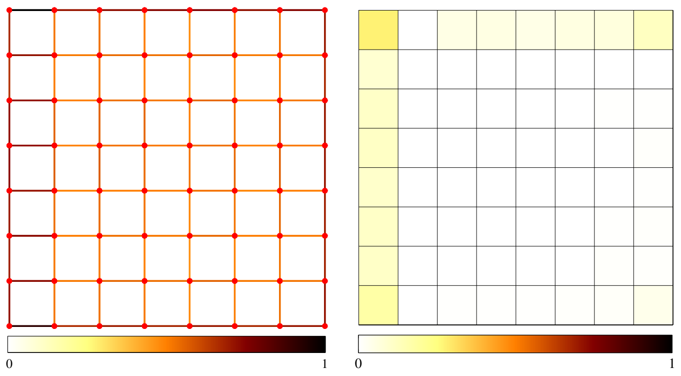


(a) Graph weights of a grid graph

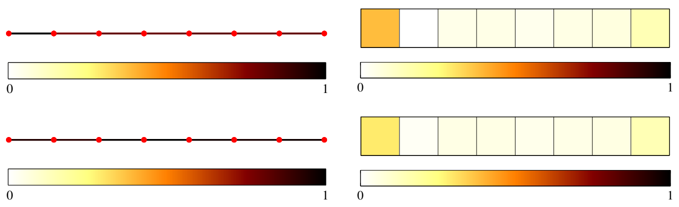


(b) Graph weights of line graphs for rows (top) and columns (bottom)

Figure 5.10: Edge weights (left) and vertex weights (right) learned from residual blocks obtained by *intra prediction* with *planar mode*.

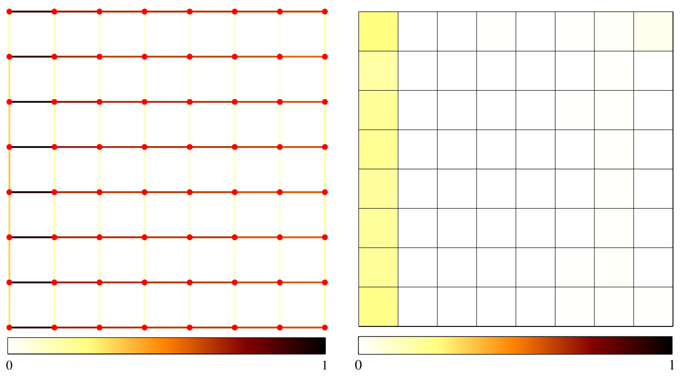


(a) Graph weights of a grid graph

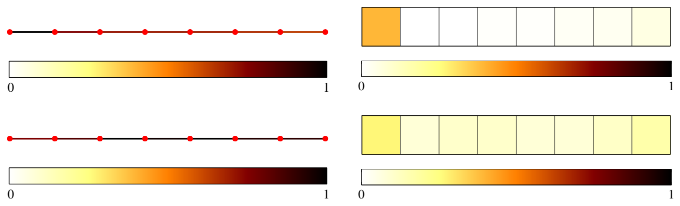


(b) Graph weights of line graphs for rows (top) and columns (bottom)

Figure 5.11: Edge weights (left) and vertex weights (right) learned from residual blocks obtained by *intra prediction* with *DC mode*.

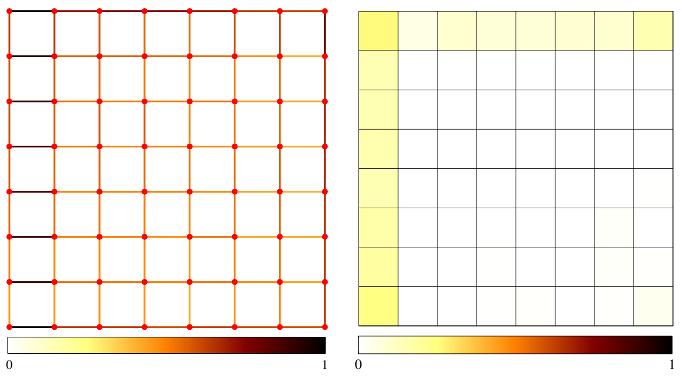


(a) Graph weights of a grid graph

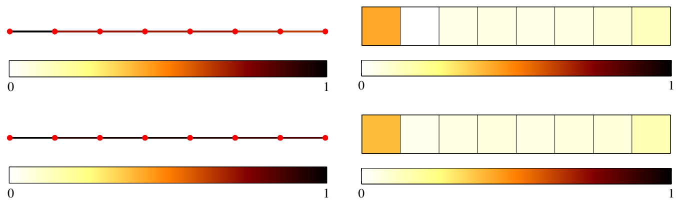


(b) Graph weights of line graphs for rows (top) and columns (bottom)

Figure 5.12: Edge weights (left) and vertex weights (right) learned from residual blocks obtained by *intra prediction* with *horizontal mode*.

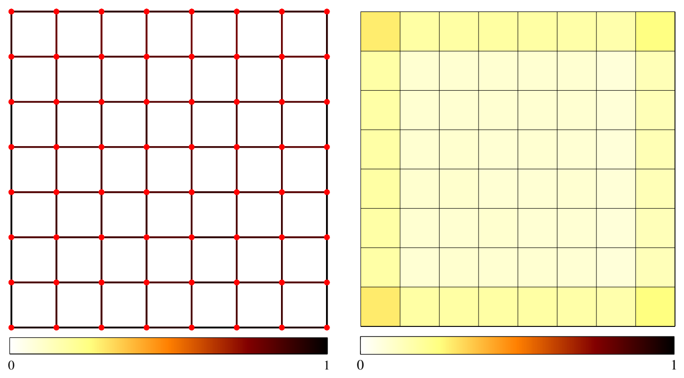


(a) Graph weights of a grid graph

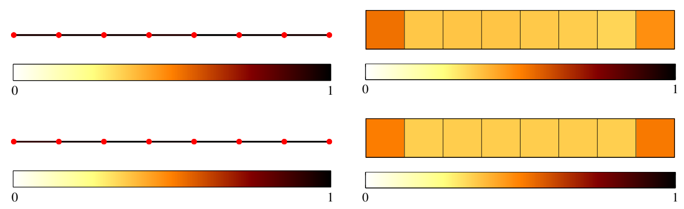


(b) Graph weights of line graphs for rows (top) and columns (bottom)

Figure 5.13: Edge weights (left) and vertex weights (right) learned from residual blocks obtained by *intra prediction* with *diagonal mode*.

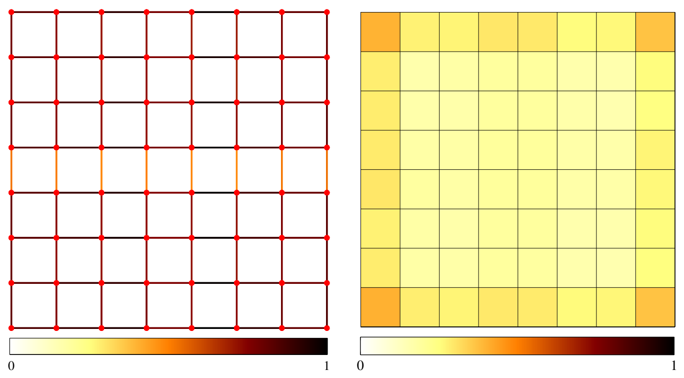


(a) Graph weights of a grid graph

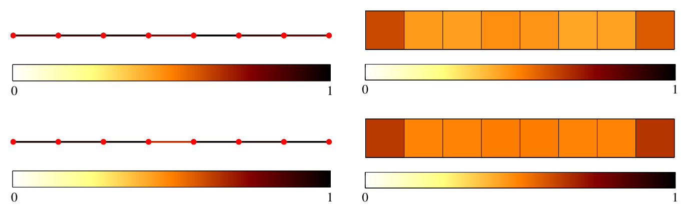


(b) Graph weights of line graphs for rows (top) and columns (bottom)

Figure 5.14: Edge weights (left) and vertex weights (right) learned from residual blocks obtained by *inter prediction* with *PU mode* $2N \times 2N$.



(a) Graph weights of a grid graph



(b) Graph weights of line graphs for rows (top) and columns (bottom)

Figure 5.15: Edge weights (left) and vertex weights (right) learned from residual blocks obtained by *inter prediction* with *PU mode* $N \times 2N$.

To evaluate the performance of transforms, we adopt the mode-dependent transform (MDT) and the rate-distortion optimized transform (RDOT) schemes. The MDT scheme assigns a single transform trained for each mode and each block size. In RDOT scheme, the best transform is selected from a set of transforms $\mathcal{T}$ by minimizing the rate-distortion cost $J(\lambda_{\text{rd}}) = D + \lambda_{\text{rd}} R$ [106] where the multiplier $\lambda_{\text{rd}} = 0.85 \times 2^{(QP-12)/3}$ [95] depends on QP parameter. In our simulations, different transform sets are chosen for each mode (i.e., class) and block size. Specifically, the RDOT scheme selects either the DCT or the transform designed for each mode and block size pair, so that the encoder has two transform choices for each block. Note that, this requires the encoder to send one extra bit to identify the RD optimized transform at the decoder side. For EA-GBTs, the necesary graph (i.e., image edge) information is also sent by using the arithmetic edge encoder (AEC) [121]. After the transformation of a block, the resulting transform coefficients are uniformly quantized, and then entropy coded using arithmetic coding [124]. The compression performance is measured in terms of Bjontegaard-delta rate (BD-rate) [125].

5.5.2 Compression Results

The following four tables compare the BD-rate performances of different transforms:

- Table 5.3 demonstrates the overall coding gains obtained by using KLTs, GBSTs and GBNTs with MDT and RDOT schemes for intra and inter predicted blocks. According to the results, GBNT outperforms KLT irrespective of the prediction type and coding scheme. This validates our observation that the proposed graph learning method leads to a more robust transform than KLT. For inter prediction, GBST performs slightly better than GBNT, since the inter predicted residuals tend to have a separable structure as shown in Figures 5.14 and 5.15. Moreover, RDOT scheme significantly outperforms MDT.

- Table 5.4 compares the RDOT coding performance of KLTs, GBSTs and GBNTs on residual blocks with different prediction modes. In RDOT scheme the transform sets are $\mathcal{T}_{\text{KLT}} = \{\text{DCT}, \text{KLT}\}$, $\mathcal{T}_{\text{GBST}} = \{\text{DCT}, \text{GBST}\}$ and $\mathcal{T}_{\text{GBNT}} = \{\text{DCT}, \text{GBNT}\}$, which consist of DCT and a trained transform for each mode and block size. The results show that GBNT consistently outperforms KLT for all prediction modes. Similar to Table 5.3, GBST provides slightly better compression compared to KLT and GBST. Also for angular modes (e.g., diagonal mode) in intra predicted coding, GBNT significantly outperforms GBST as expected.

- Table 5.5 shows (i) the coding gains obtained by using GL-GBTs (i.e., GBNTs) over KLTs and (ii) the amount of training data used to design transforms in terms of k/n, where k is the number of data samples and n is the number of vertices (i.e., number of pixels in the model). For all cases, k/n is larger than 1000, so that the training dataset has sufficient amount of data as empirically shown in Section 2.5.1 where relative errors (RE) are very close to zero for $k/n = 1000$. Hence, the performance gain obtained by GL-GBT over KLTs is not due to lack of data, yet it is mainly because GL-GBTs provide better generalization compared to KLTs. Also, the BD-rates in Table 5.5 demonstrates that the level of coding gains largely depend on the residual signal characteristics rather than k/n. Note that having a small k/n does not indicate whether the BD-rate would be larger or smaller. For example, the gain is larger for Planar mode even though k/n is smaller for DC mode, where the opposite is the case in comparison of diagonal and horizontal modes.

- Table 5.6 demonstrates the RDOT coding performance of EA-GBTs for different modes. As shown in the table, the contribution of EA-GBT within the transform set $\mathcal{T}_{\text{GL-GBT+EA-GBT}} = \{\text{DCT}, \text{GL-GBT}, \text{EA-GBT}\}$ is limited to 0.3% for intra predicted coding, while it is approximately 0.8% for inter coding. On the other hand, if the transform set is selected as $\mathcal{T}_{\text{EA-GBT}} = \{\text{DCT}, \text{EA-GBT}\}$ the contribution of EA-GBT provides considerable coding gains, which are approximately 0.5% for intra and 1% for inter predicted coding.

Table 5.3: Comparison of KLT, GBST and GBNT with MDT and RDOT schemes in terms of BD-rate (% bitrate reduction) with respect to the DCT. Smaller (negative) BD-rates mean better compression.

Transform	Intra Prediction		Inter Prediction	
	MDT	RDOT	MDT	RDOT
KLT	-1.81	-6.02	-0.09	-3.28
GBST	-1.16	-4.61	$\mathbf{-0.25}$	$\mathbf{-3.89}$
GBNT	$\mathbf{-2.04}$	$\mathbf{-6.70}$	-0.18	-3.68

5.6 Conclusion

In this work, we discuss the class of transforms, called graph-based transforms (GBTs), with their applications to video compression. In particular, separable and nonseparable GBTs are introduced and two different design strategies are proposed. Firstly, the GBT design problem is posed as a graph learning problem, where we estimate graphs from data and the resulting graphs are used to define GBTs (GL-GBTs). Secondly, we propose edge-adaptive GBTs (EA-GBTs) which can be adapted on a per-block basis using side-information (image edges in a given block). We also give theoretical justifications for these two strategies and show that well-known transforms such as DCTs and DSTs are special cases of GBTs, and graphs can be used to design generalized (DCT-like or DST-like) separable transforms. Our experiments demonstrate that GL-GBTs can provide considerable coding gains with respect to standard transform coding schemes using DCT. In comparison with the Karhunen-Loeve transform (KLT), GL-GBTs are more robust and provide better generalization. Although coding gains obtained by including EA-GBTs in addition to GL-GBTs in the RDOT scheme are limited, using EA-GBTs only provides considerable coding gains over DCT.

Table 5.4: Comparison of KLT, GBST and GBNT for coding of different prediction modes in terms of BD-rate with respect to the DCT. Smaller (negative) BD-rates mean better compression.

Transform Set	Intra Prediction					Inter Prediction		
	Planar	DC	Diagonal	Horizontal	All modes	Square	Rectangular	All modes
$\mathcal{T}_{\text{KLT}}$	−5.79	−4.57	−7.68	−6.14	−6.02	−3.47	−2.93	−3.35
$\mathcal{T}_{\text{GBST}}$	−5.45	−4.12	−3.32	−6.45	−4.61	**−3.95**	**−3.25**	**−3.89**
$\mathcal{T}_{\text{GBNT}}$	**−6.27**	**−5.04**	**−8.74**	**−6.53**	**−6.70**	−3.84	−3.18	−3.68

Table 5.5: The coding gains achieved by using GL-GBT over KLT in terms of BD-rate and the number of training data samples used per number of vertices (k/n) to design transforms for different prediction modes. Negative BD-rates mean that GL-GBT outperforms KLT.

	Intra Prediction					Inter Prediction		
	Planar	DC	Diagonal	Horizontal	All modes	Square	Rectangular	All modes
BD-rate	−0.51	−0.49	−1.15	−0.42	−0.72	−0.38	−0.26	−0.34
k/n	9.48×10^4	5.56×10^4	2.87×10^3	2.45×10^4	1.26×10^4	3.54×10^4	5.05×10^3	6.80×10^3

Table 5.6: The contribution of EA-GBTs in terms of BD-rate with respect to DCT.

Transform Set	Intra Prediction					Inter Prediction		
	Planar	DC	Diagonal	Horizontal	All modes	Square	Rectangular	All modes
$\mathcal{T}_{\text{GL-GBT}}$	−6.27	−5.04	−8.74	−6.53	−6.70	−3.95	−3.25	−3.89
$\mathcal{T}_{\text{EA-GBT}}$	−0.51	−0.47	−0.69	−0.66	−0.54	−1.01	−0.73	−0.93
$\mathcal{T}_{\text{GL-GBT+EA-GBT}}$	−6.58	−5.34	−9.07	−6.89	−7.01	−4.80	−3.65	−4.73

Chapter 6

Conclusions and Future Work

In this thesis, we propose novel techniques to build graph-based models from signals/data where the models of interest are defined based on graph Laplacian matrices. Particularly, three categories of graph learning problems are addressed for graph-based modeling. Firstly (in Chapter 2), an optimization framework is proposed to estimate three types of graph Laplacian matrices (i.e., GGLs, DDGLs and CGLs defined in Section 1.1) from data under structural constraints. For each type of graph Laplacian, specialized block-coordinate descent algorithms are developed by incorporating the structural constraints. From the probabilistic perspective, proposed algorithms can be viewed as methods to estimate parameters of attractive GMRFs, whose precision matrices are graph Laplacian matrices (as discussed in Section 2.3). Our comprehensive experimental results demonstrate that our proposed algorithms provide more accurate and computationally efficient graph learning as compared to the state-of-the-art approaches in [30, 31, 32, 33, 34]. The proposed methods can be useful in applications that involve learning a similarity graph to represent affinity relations between the entries of a dataset. Depending on the application, learning specific type of graph Laplacian matrices (GGL, DDGL or CGL matrices) can also be useful (as discussed in Section 1.2). The present work primarily focuses on applications of graph-based models used to design transforms for video coding, and other related applications such as spectral clustering [18, 19, 20], graph-based regularization and denoising [21, 79, 9] are considered as part of the future work. In addition, some of the ideas proposed in [50, 51, 52] for efficient and large-scale sparse inverse covariance estimation can be implemented in our algorithms as extensions. Secondly (in Chapter 3), the graph-based modeling is formulated as a graph system identification problem, where the aim is to jointly identify a combinatorial graph Laplacian (CGL) matrix and a graph-based filter (GBF) from a set of observed signals. In this work, we specifically consider the identification graph systems with the GBFs in Table 3.1, and propose a novel alternating optimization algorithm, which iteratively solves for a CGL and a GBF. At each iteration, a prefiltering operation defined by the estimated GBF is applied on the observed signals, and a CGL is estimated from prefiltered signals by solving Problem 3 in Chapter 2. The experimental results demonstrate that our proposed algorithm outperforms the existing methods on modeling smooth signals and learning diffusion-based models [31, 32, 33, 34] in terms of graph estimation accuracy. The proposed approach can be used to learn diffusion kernels from signals/data, which are special cases of graph systems widely used in many applications [21, 17, 79]. Our future work focuses on the extensions of our algorithm for joint identification of graphs and polynomial filters (i.e., estimation of polynomials of graph Laplacians), which can provide more degrees of freedom in designing filters than the GBFs in Table 3.1. Thirdly (in Chapter 4), the multigraph combining problem is studied and a novel algorithm is proposed to optimize a single graph from a dataset consisting of multiple graph Laplacians. Our simulation results show that the proposed algorithm provides better graph-based models than (i) the commonly used averaging method and (ii) the direct Laplacian estimation approach in terms of both coding gain and graph Laplacian quadratic form (signal smoothness) metrics. The proposed combining method can also be used to build an aggregate/ensemble graph-based model from clusters/groups of data samples. For example, multiple graph Laplacians can be estimated from clusters of data samples by using the methods in Chapters 2 and 3 so that each graph is associated with a cluster, and then the proposed multigraph combining algorithm can be employed to learn an aggregate graph-based model. This type of approach can be useful in developing optimized transforms for coding clusters of signals/data. In the case of video coding, multigraph combining can be used to develop graphs from multiple edge-based models (discussed in Section 5.3) in order to adapt block characteristics with different edge structures or to combine graph-based models obtained from clusters of samples associated with different intra/inter modes (discussed in Section 5.4). In our future work, we will explore applications of multigraph combining to design transforms for video coding.

Finally in (Chapter 5), two distinct methods are developed to construct graph-based transforms (GBTs) for video compression. In one method, instances of the GGL estimation problem (i.e., Problem 1 in Chapter 2) with line and grid graph (connectivity) constraints to learn the optimal graph weights from residual block samples, and the resulting line and grid graphs are used to derive separable and nonseparabe GBTs (GL-GBTs), respectively. The other method constructs a graph (i.e., an edge-based residual model discussed in Section 5.3) for each residual block based on detected image edges, so that the corresponding GBT (EA-GBTs) are adapted on a per-block basis. The proposed methods are theoretically and empirically justified. The experimental results demonstrate that proposed GL-GBTs can provide better compression than KLTs, and EA-GBTs can achieve considerable coding gains over DCT. However, the overall contribution of EA-GBT over GL-GBT remains limited (in terms of BD-rates). Our future work includes extending our edge-based model for image edges with smooth transitions (e.g., ramp edges [91]) in order to improve EA-GBT designs. Also, practical implementations of GBTs on a state-of-the-art encoder are considered as part of the future work.

Bibliography

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

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

[3] 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.

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

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

[6] 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.

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

[8] 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.

[9] 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.

[10] S. K. Narang and A. Ortega, "Perfect reconstruction two-channel wavelet filter banks for graph structured data," *IEEE Transactions on Signal Processing*, vol. 60, no. 6, pp. 2786–2799, June 2012.

[11] H. E. Egilmez and A. Ortega, "Spectral anomaly detection using graph-based filtering for wireless sensor networks," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2014, pp. 1085–1089.

[12] A. Anis, A. Gadde, and A. Ortega, "Efficient sampling set selection for bandlimited graph signals using graph spectral proxies," *IEEE Transactions on Signal Processing*, vol. 64, no. 14, pp. 3775–3789, July 2016.

[13] T. Hastie, R. Tibshirani, and J. Friedman, *The elements of statistical learning: data mining, inference and prediction*, 2nd ed. Springer, 2008.

[14] Y. S. Abu-Mostafa, M. Magdon-Ismail, and H.-T. Lin, *Learning From Data.* AMLBook, 2012.

[15] R. I. Kondor and J. D. Lafferty, "Diffusion kernels on graphs and other discrete input spaces," in *Proceedings of the Nineteenth International Conference on Machine Learning*, ser. ICML '02. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2002, pp. 315–322. [Online]. Available: http://dl.acm.org/citation.cfm?id=645531.655996

[16] F. R. K. Chung, *Spectral Graph Theory.* USA: American Mathematical Society, 1997.

[17] J. Lafferty and G. Lebanon, "Diffusion kernels on statistical manifolds," *J. Mach. Learn. Res.*, vol. 6, pp. 129–163, Dec. 2005.

[18] A. Y. Ng, M. I. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm," in *Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic*, ser. NIPS'01. Cambridge, MA, USA: MIT Press, 2001, pp. 849–856.

[19] M. Soltanolkotabi, E. Elhamifar, and E. J. Candès, "Robust subspace clustering," *Ann. Statist.*, vol. 42, no. 2, pp. 669–699, 04 2014.

[20] U. Luxburg, "A tutorial on spectral clustering," *Statistics and Computing*, vol. 17, no. 4, pp. 395–416, Dec. 2007.

[21] A. J. Smola and I. R. Kondor, "Kernels and regularization on graphs." in *Proceedings of the Annual Conference on Computational Learning Theory*, 2003.

[22] D. A. Spielman, "Algorithms, graph theory, and the solution of laplacian linear equations," in *Automata, Languages, and Programming - 39th International Colloquium, ICALP 2012*, 2012, pp. 24–26.

[23] T. Bıyıkoglu, J. Leydold, and P. F. Stadler, "Laplacian eigenvectors of graphs," *Lecture notes in mathematics*, vol. 1915, 2007.

[24] S. Kurras, U. Von Luxburg, and G. Blanchard, "The f-adjusted graph Laplacian: A diagonal modification with a geometric interpretation," in *Proceedings of the 31st International Conference on International Conference on Machine Learning*, 2014, pp. 1530–1538.

[25] F. Dorfler and F. Bullo, "Kron reduction of graphs with applications to electrical networks," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 60, no. 1, pp. 150–163, Jan 2013.

[26] D. A. Spielman and S.-H. Teng, "A local clustering algorithm for massive graphs and its application to nearly linear time graph partitioning," *SIAM Journal on Computing*, vol. 42, no. 1, pp. 1–26, 2013.

[27] J. Batson, D. A. Spielman, N. Srivastava, and S.-H. Teng, "Spectral sparsification of graphs: Theory and algorithms," *Commun. ACM*, vol. 56, no. 8, pp. 87–94, Aug. 2013.

[28] D. A. Spielman and S. Teng, "Nearly linear time algorithms for preconditioning and solving symmetric, diagonally dominant linear systems," *SIAM J. Matrix Analysis Applications*, vol. 35, no. 3, pp. 835–885, 2014.

[29] A. Sandryhaila and J. M. F. Moura, "Discrete signal processing on graphs: Frequency analysis," *IEEE Transactions on Signal Processing*, vol. 62, no. 12, pp. 3042–3054, June 2014.

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

[31] X. Dong, D. Thanou, P. Frossard, and P. Vandergheynst, "Learning Laplacian matrix in smooth graph signal representations," *IEEE Transactions on Signal Processing*, vol. 64, no. 23, pp. 6160–6173, Dec 2016.

[32] 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.

[33] B. M. Lake and J. B. Tenenbaum, "Discovering structure by learning sparse graph," in *Proceedings of the 33rd Annual Cognitive Science Conference*, 2010, pp. 778–783.

[34] 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

[35] 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.

[36] ——, "Graph learning from data under structural and Laplacian constraints," *CoRR*, vol. abs/1611.05181v1, 2016. [Online]. Available: https://arxiv.org/abs/1611.05181v1

[37] ——, "Graph learning from data under laplacian and structural constraints," *IEEE Journal of Selected Topics in Signal Processing*, vol. PP, no. 99, pp. 1–1, 2017.

[38] MATLAB, *version 8.1.0 (R2013a)*. Natick, Massachusetts: The MathWorks Inc., 2013.

[39] H. E. Egilmez, E. Pavez, and A. Ortega, "GLL: Graph Laplacian learning package, version 1.0," https://github.com/STAC-USC/Graph_Learning, 2017.

[40] E. J. Candès, M. B. Wakin, and S. P. Boyd, "Enhancing sparsity by reweighted ℓ-1 minimization," *Journal of Fourier Analysis and Applications*, vol. 14, no. 5, pp. 877–905, 2008.

[41] 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.

[42] 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.

[43] 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.

[44] H. Rue and L. Held, *Gaussian Markov Random Fields: Theory and Applications*, ser. Monographs on Statistics and Applied Probability. London: Chapman & Hall, 2005, vol. 104.

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

[46] A. P. Dempster, "Covariance selection," *Biometrics*, vol. 28, no. 1, pp. 157–175, 1972.

[47] N. Meinshausen and P. Buhlmann, "High-dimensional graphs and variable selection with the lasso," *The Annals of Statistics*, vol. 34, no. 3, pp. 1436–1462, 2006.

[48] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 58, no. 1, pp. 267–288, 1996.

[49] R. Mazumder and T. Hastie, "The graphical lasso: New insights and alternatives," *Electronic Journal of Statistics*, vol. 6, pp. 2125–2149, 2012.

[50] ——, "Exact covariance thresholding into connected components for large-scale graphical lasso," *The Journal of Machine Learning Research*, vol. 13, no. 1, pp. 781–794, 2012.

[51] C.-J. Hsieh, M. A. Sustik, I. S. Dhillon, P. K. Ravikumar, and R. Poldrack, "Big & quic: Sparse inverse covariance estimation for a million variables," in *Advances in Neural Information Processing Systems 26*, 2013, pp. 3165–3173.

[52] M. Grechkin, M. Fazel, D. Witten, and S.-I. Lee, "Pathway graphical lasso," in *AAAI Conference on Artificial Intelligence*, 2015, pp. 2617–2623.

[53] G. Poole and T. Boullion, "A survey on M-matrices," *SIAM review*, vol. 16, no. 4, pp. 419–427, 1974.

[54] M. Slawski and M. Hein, "Estimation of positive definite M-matrices and structure learning for attractive Gaussian Markov random fields," *Linear Algebra and its Applications*, vol. 473, pp. 145 – 179, 2015.

[55] E. Pavez and A. Ortega, "Generalized Laplacian precision matrix estimation for graph signal processing," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, March 2016, pp. 6350–6354.

[56] B. Pasdeloup, M. Rabbat, V. Gripon, 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

[57] 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.

[58] S. J. Wright, "Coordinate descent algorithms," *Math. Program.*, vol. 151, no. 1, pp. 3–34, Jun. 2015.

[59] C. Zhang, D. Florêncio, and P. A. Chou, "Graph signal processing–a probabilistic framework," *Microsoft Research Technical Report*, 2015.

[60] S. Hassan-Moghaddam, N. K. Dhingra, 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.

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

[62] 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.

[63] 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.

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

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

[66] 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.

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

[68] 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.

[69] A. Beck and L. Tetruashvili, "On the convergence of block coordinate descent type methods," *SIAM Journal on Optimization*, vol. 23, no. 4, pp. 2037–2060, 2013.

[70] C. L. Lawson and R. J. Hanson, *Solving Least Squares Problems*, ser. Series in Automatic Computation. Englewood Cliffs, NJ, USA: Prentice-Hall, 1974.

[71] 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.

[72] 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.

[73] 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, pp. 935–980, 2011.

[74] 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.

[75] H. E. Egilmez, E. Pavez, and A. Ortega, "Graph learning from filtered signals: Graph system and diffusion kernel identification," 2017, in preparation.

[76] S. Segarra, G. Mateos, A. G. Marques, and A. Ribeiro, "Blind identification of graph filters," *IEEE Transactions on Signal Processing*, vol. 65, no. 5, pp. 1146–1159, March 2017.

[77] D. Thanou, X. Dong, D. Kressner, and P. Frossard, "Learning heat diffusion graphs," *CoRR*, vol. abs/1611.01456v1, 2016. [Online]. Available: https://arxiv.org/abs/1611.01456v1

[78] J. Mei and J. M. F. Moura, "Signal processing on graphs: Causal modeling of unstructured data," *IEEE Transactions on Signal Processing*, vol. PP, no. 99, pp. 1–1, 2016.

[79] A. D. Szlam, M. Maggioni, and R. R. Coifman, "Regularization on graphs with function-adapted diffusion processes," *J. Mach. Learn. Res.*, vol. 9, pp. 1711–1739, Jun. 2008.

[80] E. Kalnay, M. Kanamitsu, R. Kistler, W. Collins, D. Deaven, L. Gandin, M. Iredell, S. Saha, G. White, J. Woollen, Y. Zhu, A. Leetmaa, R. Reynolds, M. Chelliah, W. Ebisuzaki, W. Higgins, J. Janowiak, K. C. Mo, C. Ropelewski, J. Wang, R. Jenne, and D. Joseph, "The ncep/ncar 40-year reanalysis project," *Bulletin of the American Meteorological Society*, vol. 77, no. 3, pp. 437–471, 1996.

[81] L. C. Evans, *Partial differential equations*. Providence, R.I.: American Mathematical Society, 2010.

[82] H. E. Egilmez, A. Ortega, O. G. Guleryuz, J. Ehmann, and S. Yea, "An optimization framework for combining multiple graphs," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, March 2016, pp. 4114–4118.

[83] W. Tang, Z. Lu, and I. S. Dhillon, "Clustering with multiple graphs," in *Proc. IEEE International Conference on Data Mining (ICDM)*, 2009, pp. 1016–1021.

[84] A. Argyriou, M. Herbster, and M. Pontil, "Combining graph Laplacians for semi-supervised learning," in *Neural Information Processing Systems, (NIPS)*, 2005, pp. 67–74.

[85] B. N. Flury, "Common principal components in k groups," *Journal of the American Statistical Association*, vol. 79, no. 388, pp. 892–898, 1984.

[86] B. N. Flury and W. Gautschi, "An algorithm for simultaneous orthogonal transformation of several positive definite symmetric matrices to nearly diagonal form," *SIAM J. Sci. Stat. Comput.*, vol. 7, no. 1, pp. 169–184, Jan. 1986.

[87] A. Belouchrani, K. Abed-Meraim, J.-F. Cardoso, and E. Moulines, "A blind source separation technique using second-order statistics," *IEEE Transactions on Signal Processing*, vol. 45, no. 2, pp. 434–444, Feb 1997.

[88] J.-F. Cardoso and A. Souloumiac, "Jacobi angles for simultaneous diagonalization," *SIAM J. Mat. Anal. Appl.*, vol. 17, no. 1, pp. 161–164, Jan. 1996.

[89] G. H. Golub and H. A. van der Vorst, "Eigenvalue computation in the 20th century," *Journal of Computational and Applied Mathematics*, vol. 123, no. 12, pp. 35 – 65, 2000.

[90] K. R. Rao and P. Yip, *Discrete Cosine Transform Algorithms, Advantages, Applications*. San Diego, CA, USA: Academic Press Professional, Inc., 1990.

[91] Y. H. Chao, H. E. Egilmez, A. Ortega, S. Yea, and B. Lee, “Edge adaptive graph-based transforms: Comparison of step/ramp edge models for video compression,” in *2016 IEEE International Conference on Image Processing (ICIP)*, Sept 2016, pp. 1539–1543.

[92] H. E. Egilmez, A. Said, Y.-H. Chao, and A. Ortega, “Graph-based transforms for inter predicted video coding,” in *IEEE International Conference on Image Processing (ICIP)*, Sept 2015, pp. 3992–3996.

[93] H. E. Egilmez, Y. H. Chao, and A. Ortega, “Graph-based transforms for video coding,” 2017, in preparation.

[94] W. B. Pennebaker and J. L. Mitchell, *JPEG Still Image Data Compression Standard*, 1st ed. Norwell, MA, USA: Kluwer Academic Publishers, 1992.

[95] G. J. Sullivan, J.-R. Ohm, W.-J. Han, and T. Wiegand, “Overview of the High Efficiency Video Coding (HEVC) Standard,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 1649–1668, Dec. 2012.

[96] D. Mukherjee, J. Bankoski, R. S. Bultje, A. Grange, J. Han, J. Koleszar, P. Wilkins, and Y. Xu, “The latest open-source video codec VP9 – an overview and preliminary results,” in *Proc. 30th Picture Coding Symp.*, San Jose, CA, Dec. 2013.

[97] J. Han, A. Saxena, V. Melkote, and K. Rose, “Jointly optimized spatial prediction and block transform for video and image coding,” *IEEE Trans. Image Process.*, vol. 21, no. 4, pp. 1874–1884, Apr. 2012.

[98] Y. Ye and M. Karczewicz, “Improved H.264 intra coding based on bi-directional intra prediction, directional transform, and adaptive coefficient scanning,” in *IEEE International Conference on Image Processing (ICIP)*, Oct 2008, pp. 2116–2119.

[99] S. Takamura and A. Shimizu, “On intra coding using mode dependent 2D-KLT,” in *Proc. 30th Picture Coding Symp.*, San Jose, CA, Dec. 2013, pp. 137–140.

[100] A. Arrufat, P. Philippe, and O. Deforges, “Non-separable mode dependent transforms for intra coding in hevc,” in *2014 IEEE Visual Communications and Image Processing Conference*, Dec 2014, pp. 61–64.

[101] F. Zou, O. Au, C. Pang, J. Dai, X. Zhang, and L. Fang, “Rate-distortion optimized transforms based on the lloyd-type algorithm for intra block coding,” *IEEE Journal of Selected Topics in Signal Processing*, vol. 7, no. 6, pp. 1072–1083, Dec 2013.

[102] X. Zhao, J. Chen, M. Karczewicz, L. Zhang, X. Li, and W. J. Chien, “Enhanced multiple transform for video coding,” in *2016 Data Compression Conference (DCC)*, March 2016, pp. 73–82.

[103] X. Zhao, J. Chen, A. Said, V. Seregin, H. E. Egilmez, and M. Karczewicz, "NSST: Non-separable secondary transforms for next generation video coding," in *2016 Picture Coding Symposium (PCS 2016)*, December 2016, pp. 1–5.

[104] A. Said, X. Zhao, M. Karczewicz, H. E. Egilmez, V. Seregin, and J. Chen, "Highly efficient non-separable transforms for next generation video coding," in *2016 Picture Coding Symposium (PCS 2016)*, December 2016, pp. 1–5.

[105] Y. H. Chao, A. Ortega, and S. Yea, "Graph-based lifting transform for intra-predicted video coding," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, March 2016, pp. 1140–1144.

[106] A. Ortega and K. Ramchandran, "Rate-distortion methods for image and video compression," *IEEE Signal Process. Mag.*, vol. 15, no. 6, pp. 23–50, Nov. 1998.

[107] W. Hu, G. Cheung, and A. Ortega, "Intra-prediction and generalized graph fourier transform for image coding," *IEEE Signal Processing Letters*, vol. 22, no. 11, pp. 1913–1917, Nov 2015.

[108] C. Zhang, D. Florencio, and P. Chou, "Graph signal processing - a probabilistic framework," Microsoft Research, Tech. Rep. MSR-TR-2015-31, April 2015. [Online]. Available: http://research.microsoft.com/apps/pubs/default.aspx?id=243326

[109] G. Shen, W.-S. Kim, S. Narang, A. Ortega, J. Lee, and H. Wey, "Edge-adaptive transforms for efficient depth map coding," in *Picture Coding Symposium (PCS), 2010*, Dec 2010, pp. 566–569.

[110] W. Hu, G. Cheung, A. Ortega, and O. C. Au, "Multiresolution graph fourier transform for compression of piecewise smooth images," *IEEE Transactions on Image Processing*, vol. 24, no. 1, pp. 419–433, Jan 2015.

[111] G. Strang, "The discrete cosine transform," *SIAM Rev.*, vol. 41, no. 1, pp. 135–147, Mar. 1999.

[112] M. Püschel and J. M. F. Moura, "Algebraic signal processing theory: 1-D space," *IEEE Transactions on Signal Processing*, vol. 56, no. 8, pp. 3586–3599, 2008.

[113] H. E. Egilmez, O. G. Guleryuz, J. Ehmann, and S. Yea, "Row-column transforms: Low-complexity approximation of optimal non-separable transforms," in *2016 IEEE International Conference on Image Processing (ICIP)*, Sept 2016, pp. 2385–2389.

[114] E. Pavez, A. Ortega, and D. Mukherjee, "Learning separable transforms by inverse covariance estimation," in *2017 IEEE International Conference on Image Processing (ICIP)*, Sept 2017, pp. 1–1.

[115] A. Gersho and R. M. Gray, *Vector Quantization and Signal Compression.* Norwell, MA, USA: Kluwer Academic Publishers, 1991.

[116] V. K. Goyal, "Theoretical foundations of transform coding," *IEEE Signal Processing Magazine*, vol. 18, no. 5, pp. 9–21, Sep 2001.

[117] S. Mallat, *A Wavelet Tour of Signal Processing, Third Edition: The Sparse Way*, 3rd ed. Academic Press, 2008.

[118] V. N. Vapnik, "An overview of statistical learning theory," *IEEE Transactions on Neural Networks*, vol. 10, no. 5, pp. 988–999, Sep 1999.

[119] U. von Luxburg and B. Schölkopf, *Statistical Learning Theory: Models, Concepts, and Results.* Amsterdam, Netherlands: Elsevier North Holland, May 2011, vol. 10, pp. 651–706.

[120] R. Vershynin, "How close is the sample covariance matrix to the actual covariance matrix?" *Journal of Theoretical Probability*, vol. 25, no. 3, pp. 655–686, 2012.

[121] I. Daribo, D. Florencio, and G. Cheung, "Arbitrarily shaped motion prediction for depth video compression using arithmetic edge coding," *IEEE Transactions on Image Processing*, vol. 23, no. 11, pp. 4696–4708, Nov 2014.

[122] M. Effros, H. Feng, and K. Zeger, "Suboptimality of the karhunen-loeve transform for transform coding," *IEEE Transactions on Information Theory*, vol. 50, no. 8, pp. 1605–1619, Aug 2004.

[123] T. M. Cover and J. A. Thomas, *Elements of Information Theory.* New York, NY, USA: Wiley-Interscience, 1991.

[124] A. Said, "Introduction to Arithmetic Coding: theory and practice," HP Labs, Tech. Rep. HPL-2004-76, 2004.

[125] G. Bjontegaard, "Calculation of average PSNR differences between RD-curves," *ITU-T Q.6/SG16 VCEG-M33 Contribution*, vol. 48, Apr 2001.