Tokyo University of Agriculture
and Technology

Doctoral Thesis

Regularized Time-Varying Graph Learning

Author:
Koki Yamada

Supervisor:
Dr. Yuichi Tanaka

A thesis submitted in fulfillment of the requirements
for the degree of Doctor of Philosophy

in the

Department of Electrical Engineering and Computer Science

Acknowledgements

I am extremely thankful to my supervisor Associate Prof. Yuichi Tanaka, for providing guidance and his consistent support. Besides my advisor, I would like to thank the rest of my thesis committee: Prof. Akinobu Shimizu, Prof. Hiroshi Ishida, Prof. Toshiyuki Kondo, and Prof. Toshihisa Tanaka, for their insightful comments and encouragement. I would also like to thank Prof. Antonio Ortega and Associate Prof. Shunsuke Ono, for the collaboration of this work and useful discussions. We gratefully acknowledge the work of past and present members of our laboratory. And my biggest thanks to my family for all the support you have shown me through this research.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Computer science has been rapidly developed, which enriches people's lives. With a single smartphone, we now enjoy a variety of services, such as phone calls, e-mail, web search, electronic payments, social networking, etc. Giant tech companies such as Google, Apple, Facebook, Amazon (GAFA) have been providing and producing innovative services. The foundation for such services is supported by the development of computer science.

In the last decade, one of the most successful technologies in computer science has been *Deep learning* [1–4]. Deep learning is a generic term that refers to machine learning using a neural network stacking multiple hidden layers (deep neural network), which contain parameters called weights. Deep neural networks have achieved amazing results in difficult tasks, e.g., object recognition, machine translation, speech recognition, self-driving car, point cloud processing, and so on. These amazing results have attracted many researchers, and deep learning is still an active research area. Deep learning seems to be a panacea, but it has some limitations. In many cases, deep neural networks cannot work well with *small data*, which is difficult to collect a large amount of data. This is because deep neural networks require a lot of data to learn many weights in multiple layers. For small data analysis, mathematical modeling and *signal processing* techniques that utilize domain knowledge are useful tools.

Signal processing is a fundamental tool to analyze, modify and synthesize signals such as sound, images, and various engineering measurements. Signal processing covers a broad range of applications, such as image and audio processing, wireless communications, control system, etc. A hot recent topic of signal processing is *graph signal processing* [5, 6]. While classical signal processing is designed to analyze signals that are arranged regularly, graph signal processing aims at analyzing signals distributed on network structures, e.g., sensor, traffic, brain, and social networks. Graph signal processing utilizes graphs to analyze such irregularly distributed signals. Graphs, consisting of sets of

nodes and edges, are a fundamental tool to describe the relationship among entities. Graph edges and the corresponding edge weights can be used to capture the similarity between nodes (where a higher positive weight indicates greater similarity). Introducing a graph representation enables us to efficiently analyze signals on networks in many practical applications such as epidemics [7, 8], transportation networks [9], image processing [10–15], machine learning [16–18], and social networks [19].

Graph signal processing provides effective tools for graph signals, such as filtering, sampling, filter bank, and frequency analysis [20–29]. To use these effective tools, it is required to know graphs underlying data. However, graphs are not provided in many cases *a priori*. *Graph learning* is a method that aims at identifying graphs from observed data [30–36]. Each observation is a vector, and each entry corresponds to the observation at one node. The goal is to obtain the weights of all the edges connecting those nodes. Most graph learning methods identify a single static graph from all observations [37–45]. These static graph learning methods assume that the node relationships obtained from the observations do not change during the measurement process. However, in many applications where the observations are obtained over a period of time, a time-varying graph will provide a better model. Examples of such applications include estimation of time-varying brain functional connectivity from EEG or fMRI data [46], identification of temporal transit of biological networks such as protein, RNA, and DNA [47], and inference of relationships among companies from historical stock price data [48], dynamic point cloud processing, and analysis of physical measurement data such as temperature.

This dissertation addresses the problem of learning time-varying graphs from multivariate time-series data. This problem is called *time-varying graph learning* (TVGL). The main contributions of this dissertation are described in Section 1.1, and the related studies are described in Section 1.2. Section 1.3 shows the outline of this dissertation.

1.1 Contributions

The desired properties of time-varying graph learning are summarized as follows:

1. Time-varying graph learning methods should estimate time-varying graphs considering their temporal relationship. A straightforward approach to estimate a time-varying graph would consist of aggregating temporal observations into non-overlapping windows and then using an existing static

graph learning method to estimate a graph for each time window. However, such an approach estimates a graph *independently* for each temporal interval, thus ignoring temporal relations that may exist in time-varying graphs.

2. Time-varying graph learning requires estimating graphs from time windows containing only a small fraction of observations due to the trade-off between the choice of window length and temporal resolution. For example, if we choose a short window to adapt to fast temporal changes in the graph, then we may not have enough data to learn a graph within each window.

This dissertation proposes two time-varying graph learning methods: one approach with constraints of temporal graph variation and the other based on the temporal multiresolution structure of time-varying graphs. Their contributions are listed as follows:

1. **Time-varying graph learning with constraints on graph temporal variation (Chapter 4 and [49, 50])**
   This method is a generic framework for learning time-varying graphs from spatiotemporal measurements. Given an appropriate prior on the temporal behavior of signals, this method can estimate time-varying graphs from a small number of available measurements. Specifically, this method focuses on two time-varying graph model: **(P1)** time-varying graph with *temporal homogeneity* and **(P2)** time-varying graphs with *switching behavior*. To achieve this, we introduce two regularization terms in convex optimization problems that constrain the sparseness of temporal variations of the time-varying networks. Moreover, a computationally-scalable algorithm is introduced to efficiently solve the optimization problem.

2. **Temporal multiresolution graph learning (Chapter 5 and [51,52])**
   This method is an approach for time-varying graph learning by leveraging a multiresolution property. This method assumes that time-varying graphs can be decomposed by a linear combination of graphs localized at different temporal resolutions. Utilizing the multiresolution property improves a trade-off between temporal resolution and the number of samples available for learning and enables to detection of graphs localized at different temporal resolutions. This problem is formulated as a convex optimization problem for temporal multiresolution graph learning. In experiments using synthetic and real data, the proposed method demonstrates

the promising objective performances for synthetic data and obtains reasonable temporal multiresolution graphs from real data.

In addition to the above two TVGL methods, this dissertation proposes a graph learning information criterion (GLIC), which is a model selection method for graph learning. The contribution of this method is summarized as follows:

3. **Graph learning information criterion (Chapter 6)**
   Although graph learning is required in many applications, e.g., classification, prediction, and clustering, there is no established method to determine hyperparameters that control the strength of the regularization reflecting prior knowledge. To resolve the problem, GLIC considers a model selection criterion for the graph learning problem based on the Laplacian constrained Gaussian Markov random field. GLIC is the value based on model evidence, which is used for model selection in Bayesian statistics. It can be estimated by averaging the negative log-likelihood over the posterior distribution of a graph learning model. To compute this criterion, an efficient sampler of the posterior distribution is presented.

1.2 Related Work

Many methods for graph learning have been proposed thus far. Most of them are summarized in the two overview papers [30, 31]. Without being exhaustive, we review static graph learning (SGL) and time-varying graph learning methods related to our approach.

The basic strategy of SGL is the design of optimization problems based on some desired criteria for learned graphs. For example, [37, 38, 43] assume that signals are smooth on a graph. This characteristic is often represented as the Laplacian quadratic form. Instead of smoothness [39, 45, 53] assume that signals are generated from a Laplacian constrained Gaussian Markov random field (LGMRF), and maximize its regularized likelihood. Some studies such as like [30,31,39] suggest a relation between the signal smoothness and the LGMRF likelihood; the smoothness-based approach in [37] solves a relaxed problem of the LGMRF likelihood criterion.

Among the techniques for learning time-varying graphs, the Kalofolias et al. method [54], where constraints are introduced so that the edge weights change smoothly over time, is close to our method described in 4. This approach uses a smoothness criterion and Tikhonov regularization of temporal variation in graphs to learn a time-varying graph. However, it does not learn

time-varying graphs with temporal homogeneity exactly because Tikhonov regularization promotes smooth variation of edge weights over time, i.e., it allows changes of both edges and edge weights over short-term time horizons. While our approach has similar cost functions as those employed in [54], we use different regularization terms that favors learning time-varying graphs with temporal homogeneity. The optimization problem in our approach cannot be solved straightforwardly in the same manner as [54] because the regularization terms used are not differentiable. Therefore, we reformulate the optimization problem to make it solvable with a primal-dual splitting algorithm that leads to efficient learning of a time-varying graph.

In a different line of GSP research, some graph learning methods assume that observations are generated by applying some filters, e.g., graph variation and heat diffusion operators, to a latent signal [40, 41, 55]. Their extensions to TVGL, proposed in [56,57], focus on estimating graphs and corresponding filters simultaneously under the assumption of stationarities of graph signals or signal generation models based on graph filters. Such a simultaneous estimation is out of the scope of this study. The method proposed in this study, in contrast, is based on a different signal generation model.

Note that all previous works mentioned above are single temporal resolution TVGL approaches. In contrast to the existing approaches, the proposed method described in Chapter 5 estimates graphs having multiple temporal resolutions to capture various temporal relationships. To the best of our knowledge, this is the first attempt in which TVGL has been used to extract a TMR behavior.

Some studies focus on learning multiple graphs (not necessarily time-varying) from observations. While they yield multiple graphs, the learned graphs may not represent time-varying relationships [58, 59]. From a machine learning perspective, TVGL relates to time-varying inverse covariance estimation [48, 60–62]. The main difference between TVGL and inverse covariance estimation is whether the optimization problem contains the constraint on the graph Laplacian. For example, a well-known inverse covariance estimation, graphical Lasso [63], yields a covariance matrix that corresponds to a graph with negative edges and self-loops. This would be inappropriate if we need to learn time-varying graphs with nonnegative edge weights without self-loops, which is a typical assumption of GSP.

1.3 Outline

The remainder of this paper is organized as follows. Chapter 2 presents preliminaries concerning graph signal processing and convex optimization. Chapter 3 describes static graph learning methods from a signal generation perspective. Chapter 4 presents the regularization for temporal graph variation and the optimization problem to learn time-varying graphs and proposes an algorithm to find a solution. Chapter 5 describes a temporal multiresolution graph learning. Chapter 6 presents graph learning information criterion, which is a model selection method for graph learning. Finally, we show the conclusion of this dissertation in Chapter 7.

Chapter 2

Notaion and Preliminaries

This chapter shows the notation and preliminaries that are useful for understanding graph learning.

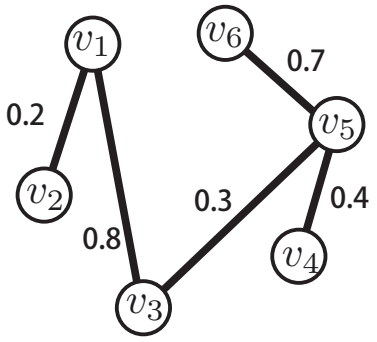
2.1 Notation

The notation used in this paper is summarized in Table 2.1. Throughout this paper, vectors and matrices are written in bold lowercase and bold uppercase letters respectively. The calligraphic capital letters, namely, $\mathcal{V}$ and $\mathcal{W}_m$, denote sets. $\mathcal{O}(\cdot)$ and $\Omega(\cdot)$ are the big-O and big-Omega notations used in complexity theory.

$\mathrm{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ is a multivariate Gaussian distribution with the mean $\boldsymbol{\mu}$ and the covariance $\boldsymbol{\Sigma}$. The inverse Gaussian distribution is denoted by $\mathrm{IGau}(\bar{\mu}, \bar{\lambda})$, where $\bar{\mu}$ and $\bar{\lambda}$ are the mean and the shape parameter, respectively. $\mathrm{Gam}(\bar{\alpha}, \bar{\beta})$ represents the gamma distribution with a shape parameter $\bar{\alpha}$ and an inverse scale parameter $\bar{\beta}$. The uniform distribution in the interval $[x, y]$ is denoted by $\mathrm{U}(x, y)$.

Table 2.1: List of notation

$a_i, (\mathbf{a})_i, a[i]$	ith entry of a vector
$A_{ij}, (\mathbf{A})_{ij}$	(i, j) entry of a matrix
$(\mathbf{A})_i$	ith column of $\mathbf{A}$
$\mathbf{A}^\dagger$	Moore-Penrose pseudoinverse of $\mathbf{A}$
$\mathrm{gdet}(\mathbf{A})$	Generalized determinant of $\mathbf{A}$
$\mathbb{R}_+$	Set of the nonnegative real numbers
$\mathbb{R}_{++}$	Set of the positive real numbers
$\circ$	Hadamard product
$\|\mathbf{a}\|_2^2, \|\mathbf{A}\|_F^2$	sum of squared values of all elements
$\|\mathbf{a}\|_1, \|\mathbf{A}\|_1$	sum of absolute values of all elements
$\mathrm{Tr}(\mathbf{A})$	Trace of a matrix
$\mathrm{diag}(\mathbf{A})$	vector formed by diagonal elements



Undirected graph

$$\begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \end{array} \begin{array}{c} \begin{array}{cccccc} v_1 & v_2 & v_3 & v_4 & v_5 & v_6 \end{array} \\ \begin{pmatrix} 0 & 0.2 & 0.8 & 0 & 0 & 0 \\ 0.2 & 0 & 0 & 0 & 0 & 0 \\ 0.8 & 0 & 0 & 0 & 0.3 & 0 \\ 0 & 0 & 0 & 0 & 0.4 & 0 \\ 0 & 0 & 0.3 & 0.4 & 0 & 0.7 \\ 0 & 0 & 0 & 0 & 0.7 & 0 \end{pmatrix} \end{array}$$

Weighted adjacency matrix

Vertex
Edge

Numbers associated to edges represent **edge weitht**, which reflect the strength of the relationship

FIGURE 2.1: Graphs

2.2 Graph Signal Processing

2.2.1 Graphs

Graphs are mathematical structures to represent the pairwise relationship between objects such as networks (see 2.1). A graph consists of nodes (which are also called vertices), edges, and edge weights. An undirected weighted graph is represented as $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{W})$, where $\mathcal{V}$ is a set of nodes, $\mathcal{E}$ is a set of edges, and $\mathbf{W}$ is a weighted adjacency matrix. The number of nodes is given by $N = |\mathcal{V}|$. Each element of the weighted adjacency matrix is defined by

$$(\mathbf{W})_{mn} = \begin{cases} w_{mn} & \text{if nodes } m \text{ and } n \text{ are connected,} \\ 0 & \text{otherwise,} \end{cases} \tag{2.1}$$

where $w_{mn} \geq 0$ is the edge weight between nodes m and n. The degree matrix $\mathbf{D}$ is a diagonal matrix whose diagonal element is $d_{mm} = \sum_n w_{mn}$.

Definition 2.1. (Various graph Laplacians) Graph Laplacian (also called combinatorial graph Laplacian) is defined as

$$\mathbf{L} := \mathbf{D} - \mathbf{W}, \tag{2.2}$$

and symmetirc normalized Laplacian is defined as

$$\mathbf{L}^{\text{sym}} := \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2} = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}. \tag{2.3}$$

2.2.2 Graph Fourier Transform

One of the fundamental tools of classical signal processing is Fourier transform, which enables us to analyze signals and design filters in frequency domain.

Definition 2.2. (Fourier transform) The Fourier transform of a function x is defined as

$$X(\omega) = \int_{-\infty}^{\infty} x(t) e^{-j\omega t} dt \tag{2.4}$$

where $\omega = 2\pi\xi \in \mathbf{R}$ and ξ is frequency. $X(\omega)$ is *spectrum* of x if $X(\omega)$ is finite for all ω. The inverse Fourier transform of X is defined as

$$x(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} X(\omega) e^{j\omega t} d\omega \tag{2.5}$$

where $t \in \mathbf{R}$.

The function $e^{-j\omega t}$ is the eigenfunction of Laplace operator:

$$-\frac{\partial^2}{\partial t^2} e^{j\omega t} = \omega^2 e^{j\omega t}. \tag{2.6}$$

Graph Fourier transform is the counterpart of Fourier transform in graph signal processing.

Definition 2.3. (Graph Fourier transform) Let $\mathbf{L} \in \mathbf{R}^{n \times N}$ be a graph Laplacian, and its eigenvalue decomposition is given by $\mathbf{L} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^{\mathsf{T}}$, where $\mathbf{U} = [\mathbf{u}_0, \mathbf{u}_1, \ldots, \mathbf{u}_{N-1}]$ is a matrix whose i-th column is the eigenvector $\mathbf{u}_i$ and $\mathbf{\Lambda} = \mathrm{diag}(\lambda_0, \lambda_1, \ldots \lambda_{N-1})$ is a diagonal eigenvalue matrix. The eigenvalues λ_i are arranged in ascending order: $0 = \lambda_0 \leq \lambda_1, \ldots \leq \lambda_{N-1}$ without loss of generality. The graph Fourier transform (GFT) is defined as

$$\hat{f}[i] = \langle \mathbf{u}_i, \mathbf{f} \rangle = \sum_{n=0}^{N-1} u_i[n] f[n], \tag{2.7}$$

and the inverse GFT is defined as

$$f[n] = \sum_{i=0}^{N-1} \hat{f}[i] u_i[n]. \tag{2.8}$$

Eigenvalues of the graph Laplacian correspond to frequencies in the classical Fourier transform, and thus, are often called *graph frequencies*. The eigenvector $\mathbf{u}_i$ is called GFT basis.

The GFT can be interpreted as the projection onto the subspace spanned by the eigenvectors of a graph Laplacian, which is the counterpart of Laplace operator in (2.6).

2.2.3 Smoothness of Graph Signal

Let $\mathbf{f} \in \mathbb{R}^N$ be a signal on the graph, then the Laplacian quadratic form is given by

$$\mathbf{f}^\mathsf{T}\mathbf{L}\mathbf{f} = \frac{1}{2}\sum_{m,n} w_{mn}(f_m - f_n)^2, \tag{2.9}$$

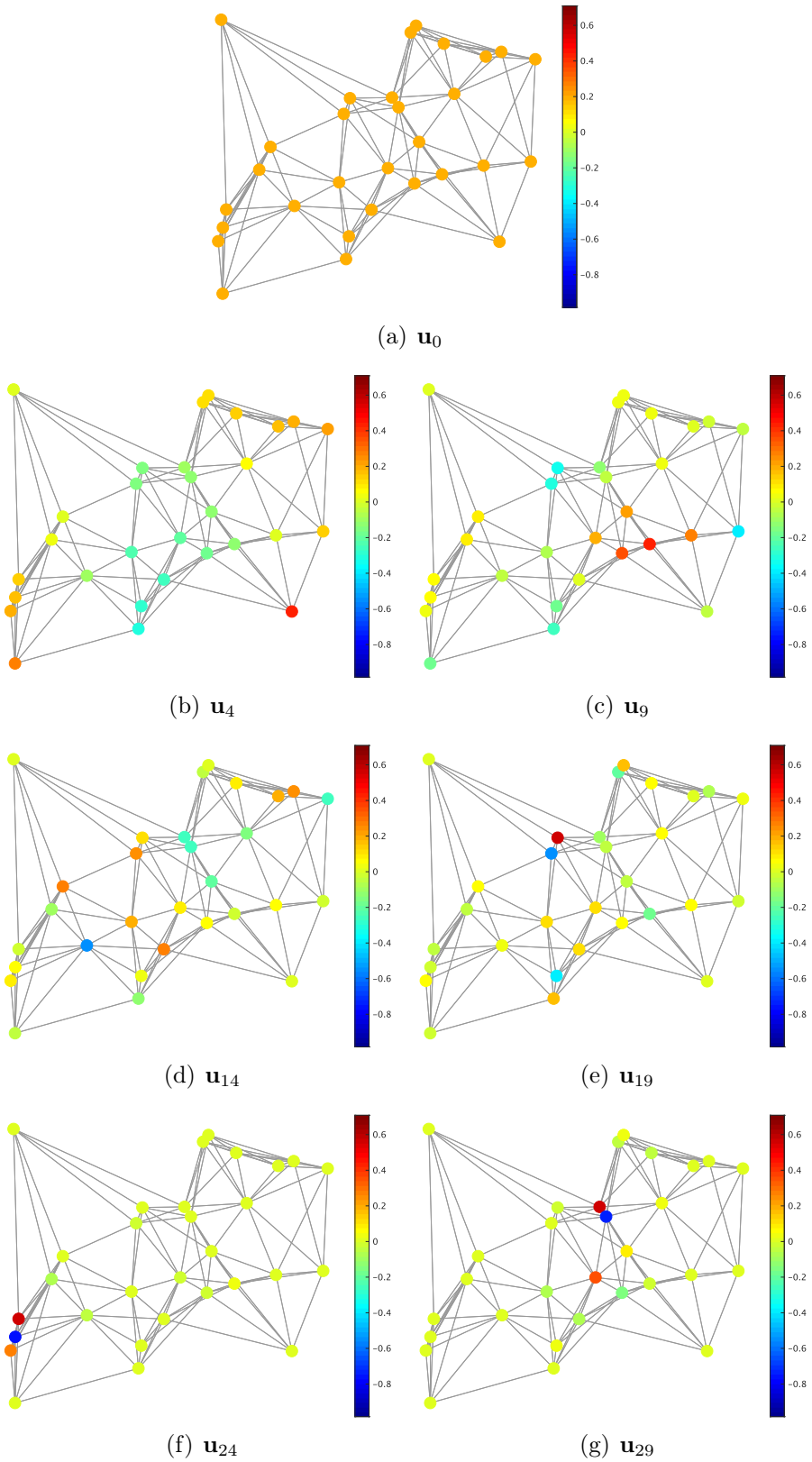
where f_m and f_n denote the signal values at nodes m and n, respectively. This implies that $\mathbf{f}^\mathsf{T}\mathbf{L}\mathbf{f}$ becomes small when the signals on the nodes connected to each other are close, and thus (2.9) provides the metric that measure the variation of the signal on the graph [5].

We demonstrate the relationship between the variation of graph signal and GFT. Consider a successive minimization problem of Rayleigh quotient of a graph Laplacian:

$$\underset{\mathbf{f}^\mathsf{T}\mathbf{u}_{k'},k'=0,\cdots k-1}{\arg\min} \frac{\mathbf{f}^\mathsf{T}\mathbf{L}\mathbf{f}}{\mathbf{f}^\mathsf{T}\mathbf{f}}. \tag{2.10}$$

The solution of this problem is given by the GFT basis $\mathbf{u}_k$ of $\mathbf{L}$ with $\lambda_k = \mathbf{u}^\mathsf{T}\mathbf{L}\mathbf{u}$ if $\mathbf{u}_k$ is normalized. Thus, the variation of the GFT basis corresponding to a small eigenvalue is small, and that of GFT basis with a large eigenvalue is large. Thus, the GFT basis with a small eigenvalue exhibits small variation, and that with a large eigenvalue shows large variation. Fig. 2.2 shows the GFT bases of a sensor graph ($N = 30$). As can be seen in this figure, the basis with low graph frequency is smooth, and the one with high frequency shows large variation.

Many applications of GSP often assume the smoothness of graph signals, e.g., image processing, graph signal sampling, and graph learning. A smooth graph signal is one whose energy tends to lie mainly in the low graph frequency region. Fig. 2.3 shows smooth and noisy graph signals on a polygon mesh and their spectra. Colors on the polygon mesh represent the signal values. Focusing on Fig 2.3(b), it finds that the energy of the spectrum of Fig 2.3(a) is concentrated in low frequency and is hardly lying in high frequency. On the other hand, the spectrum of the noisy signal in Fig 2.3(d) has some energy in the high frequency.



(a) $\mathbf{u}_0$

(b) $\mathbf{u}_4$

(c) $\mathbf{u}_9$

(d) $\mathbf{u}_{14}$

(e) $\mathbf{u}_{19}$

(f) $\mathbf{u}_{24}$

(g) $\mathbf{u}_{29}$

FIGURE 2.2: The GFT bases of a sensor graph.

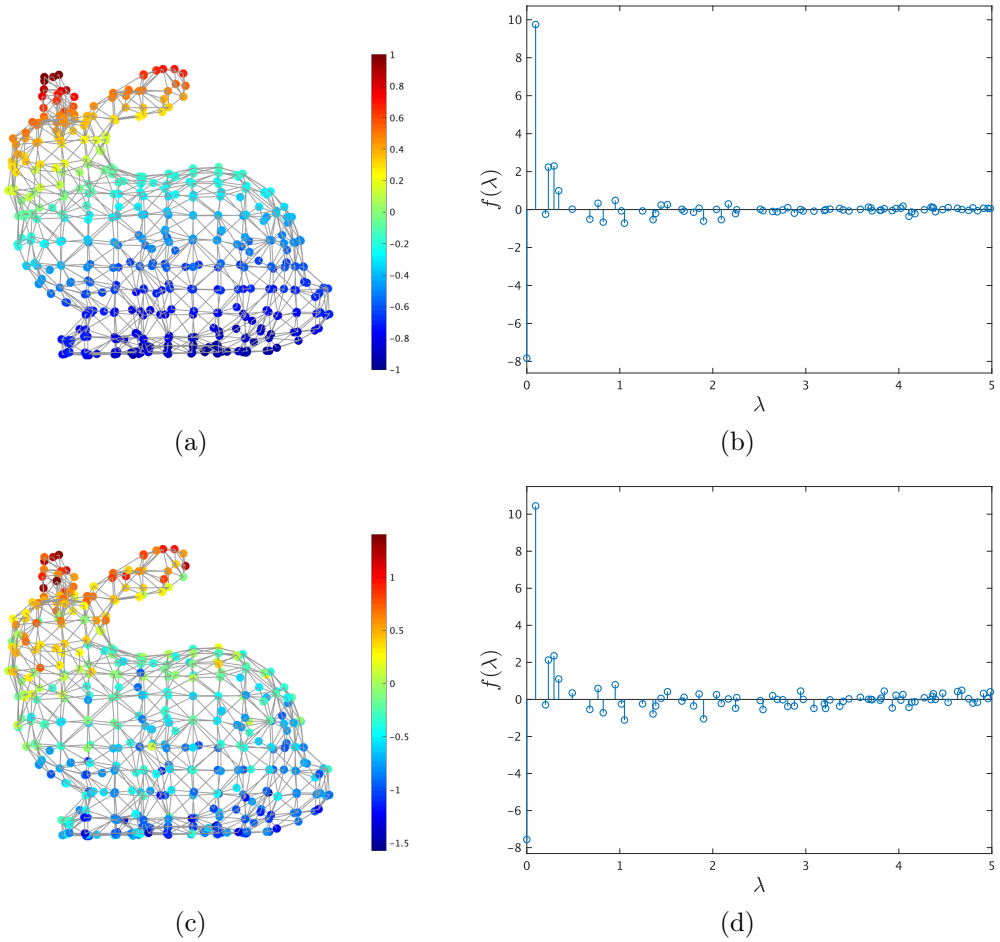


FIGURE 2.3: Graph signals and their spectra. The vertex color (a) Smooth graph signal, (b) the spectrum of (a), (c) the graph signal of (a) with white Gaussian noise $\sigma^2 = 0.3$, (d) the spectrum of (c).

2.3 Convex Optimization using Proximal Algorithm

Many problems in signal processing, machine learning, image processing, and data mining are formulated as convex optimizations, which include functions that are not necessarily differentiable. An approach to solve such problems is a proximal algorithm, which minimizes the objective function by successive evaluations of the proximal operators. Time-varying graph learning methods in introduced in this thesis are formulated as the convex optimization problem, and this can be computed by a primal-dual splitting algorithm, a kind of proximal algorithms. This section shows the basic definitions and an overview of proximal algorithms.

2.3.1 Basic Definition for Convex Optimization

In this chapter, we consider functions f defined on a real Hiilbert space $\mathcal{X}$ with values in $\mathbb{R} \cup \{\infty\}$. Some classical definitions of convex optimization are described as follows.

Definition 2.4. (Proper function.) A function f is said to be *proper* if its domain

$$\mathrm{dom}(g) = \{x \in \mathcal{X}; g(x) < +\infty\} \tag{2.11}$$

is nonempty.

Definition 2.5. (Convex function.) A function f is said to be *convex* if

$$f(ax + (1-a)y) \leq af(x) + (1-a)f(y) \tag{2.12}$$

for all $x, y \in \mathcal{X}$ and $a \in [0, 1]$.

Definition 2.6. (Lower semicontinuous function.) A function f is said to be *lower semicontinuous* at $x_0 \in \mathcal{X}$ if and only if

$$\liminf_{x \to x_0} f(x) \geq f(x_0). \tag{2.13}$$

For an operator $T : \mathcal{X} \to \mathcal{X}$, the set of fixed points is denoted by:

$$\mathrm{Fix}\, T = \{x \in \mathcal{X} \mid Tx = x\} \tag{2.14}$$

Definition 2.7. (Lipschitz continuous and contractive mapping) Let $T : \mathcal{X} \to \mathcal{X}$ defined on a metric space $(\mathcal{X}, d)$. Then T is *Lipschitz continuous* with constant $\kappa \in \mathbb{R}_+$ if

$$d(Tx, Ty) \leq \kappa d(x, y) \quad (\forall x \in \mathcal{X})\,(\forall y \in \mathcal{X}). \tag{2.15}$$

The smallest κ satisfying (2.15) is said to be *Lipschitz constant*. T is said to be contractvie mapping if $\kappa \leq 1$

Banach–Picard fixed point theorem is one of the important theorems in convex analysis.

Theorem 2.1. *(Banach–Picard fixed point theorem) Let $(\mathcal{X}, d)$ be a complete metric space and let $T : \mathcal{X} \to \mathcal{X}$ be a contractive mapping. Given $x_0 \in \mathcal{X}$, T have the unique fixed point $z \in \mathcal{X}$ and the following hold:*

$$\lim_{n \to \infty} T^n x_0 = z. \tag{2.16}$$

$$d(T^n(x_0), z) \leq \frac{\kappa^n}{1-\kappa} d(x_0, T(x_0)) \quad (n = 0, 1, 2, \cdots). \tag{2.17}$$

This theorem states that the fixed point z of an contractive mapping T can be approximated by the iteration $x_{n+1} = T(x_n)$ because $\lim_{n\to\infty} \|x_n - z\| = 0$. For an nonexpansive operator, there exists an algorithm that can approximately compute its fixed point. This is guaranteed by Krasnosel'skiĭ–Mann algorithm theorem.

Definition 2.8. (Nonexpansive and quasinonexpansive operator) Let $T : \mathcal{X} \to \mathcal{X}$. Then T is a nonexpansive operator if

(i) *nonexpansive* if

$$\|T(x) - T(y)\| \leq \|x - y\| \quad (\forall x, y \in \mathcal{X}). \tag{2.18}$$

(ii) *quasinonexpansive* if

$$\|T(x) - y\| \leq \|x - y\| \quad (\forall x \in \mathcal{X}, \forall y \in \operatorname{Fix} T). \tag{2.19}$$

Suppose that T is a nonexpansive operator such that

$$\operatorname{Fix} T := \{x \in \mathcal{H} \mid T(x) = x\} \neq \emptyset. \tag{2.20}$$

Then T is quasinonexpansive and $\operatorname{Fix}(T)$ is a closed set.

Definition 2.9. (Averaged nonexpansive operator) Let $T : \mathcal{X} \to \mathcal{X}$ be nonexpansive, and let $\alpha \in [0, 1)$. Then T is said to be α-averaged nonexpansive operator or averaged nonexpansive, if there exists a nonexpansive operator $R : \mathcal{X} \to \mathcal{X}$ such that

$$T = (1 - \alpha) \operatorname{Id} + \alpha R. \tag{2.21}$$

where Id is an identity function.

Theorem 2.2. *(Krasnosel'skiĭ–Mann algorithm) Let $T : \mathcal{X} \to \mathcal{X}$ be nonexpansive such that $\operatorname{Fix}(T) \neq \emptyset$, and let $\alpha_n \in (0, 1]/(n = 0, 1, 2, \cdots)$ be a sequence such that $\sum_{n=0}^{\infty} \alpha_n(1 - \alpha_n) = \infty$. A sequence*

$$x_{n+1} := (1 - \alpha_n)x_n + \alpha_n T(x_n) \quad (n = 0, 1, 2, \cdots) \tag{2.22}$$

converges weakly a fixed point $z \in \operatorname{Fix} T$.

2.3.2 Proximal Operator

A proximal operator plays an important role in proximal algorithms. The set of proper, convex, lower semicontinuous functions is denoted by $\Gamma_0(\mathcal{X})$, and the proximal operator is defined as follows.

Definition 2.10. (Proximal operator) The proximal operator $\text{prox}_{\gamma f} : \mathcal{X} \to \mathcal{X}$ of $f \in \Gamma_0(\mathcal{X})$ with a parameter $\gamma > 0$ is defined as

$$\text{prox}_{\gamma f}(\mathbf{x}) = \arg\min_{\mathbf{y}} f(\mathbf{y}) + \frac{1}{2\gamma}\|\mathbf{y} - \mathbf{x}\|_2^2. \tag{2.23}$$

If a proximal operator $\text{prox}_{\gamma f}$ can be computed efficiently, the function f is called proximable. Proximal operators have the following properties [64]:

(i) If a function f is separable across variables, i.e., $f(\mathbf{x}) = f_1(\mathbf{x}_1) + f_2(\mathbf{x}_2)$ with $\mathbf{x} = [\mathbf{x}_1^\mathsf{T}\ \mathbf{x}_2^\mathsf{T}]^\mathsf{T}$, then

$$\text{prox}_{\gamma f}(\mathbf{x}) = [(\text{prox}_{\gamma f_1}(\mathbf{x}_1))^\mathsf{T}\ (\text{prox}_{\gamma f_2}(\mathbf{x}_2))^\mathsf{T}]^\mathsf{T}, \tag{2.24}$$

where $\mathbf{x} = [\mathbf{x}_1^\mathsf{T}\ \mathbf{x}_2^\mathsf{T}]^\mathsf{T}$. Thus, the computation in the proximal operator of separable functions reduces to the computation of the proximal operator for each separable part.

(ii) If a function f is fully separable, i.e., $f(\mathbf{x}) = \sum_i f_i(x_i)$, then

$$(\text{prox}_{\gamma f}(\mathbf{x}))_i = \text{prox}_{\gamma f_i}(x_i). \tag{2.25}$$

Therefore, in this case, the computation of proximal operator can be reduced to the elementwise computation.

(iii) If a function f is *orthogonally invariant*, i.e.,

$$f(\mathbf{VXU}) = f(\mathbf{X}), \tag{2.26}$$

for all $\mathbf{X} \in \mathbb{R}^{m\times n}$, $\mathbf{U} \in \mathbb{R}^{n\times n}$, and $\mathbf{V} \in \mathbb{R}^{m\times m}$, where $\mathbf{U}$ and $\mathbf{V}$ are orthogonal matrices, then the following hold:

$$\begin{aligned} \text{prox}_{\gamma f}(\mathbf{X}) &= \mathbf{V}\,\text{diag}\left(\text{prox}_{\gamma F}(\sigma(\mathbf{X}))\right)\mathbf{U}, \\ f &= F \circ \sigma, \end{aligned} \tag{2.27}$$

where $\sigma(\cdot)$ is the singular value mapping that returns its singular values in nonincreasing order, and f is an absolutely symmetric matrix, meaning

that $f(\mathbf{Px}) = f(\mathbf{x})$ for all $\mathbf{x} \in \mathbb{R}$ and any signed permutation matrix $\mathbf{P}$. (2.27) implies that the proximal operator $\text{prox}_{\gamma f}(\mathbf{X})$ can be computed by the evaluation of the proximal operator of the corresponding absolutely symmetric function F at $\sigma(\mathbf{X})$. This properties is used to compute the proximal operator of nuclear norm (also known as trace norm).

2.3.3 Proximal Gradient Method

We consider the following problem:

$$\min_{\mathbf{x}\in\mathbb{R}^N} f(\mathbf{x}) + g(\mathbf{x}) \tag{2.28}$$

where $f \in \Gamma_0(\mathcal{X})$ is differentiable convex function such that ∇f is κ-Lipschitz continuous with $\kappa > 0$, and $g \in \Gamma_0(\mathcal{X})$ is a proximable. Note that g does not need to be differentiable. Suppose that there exists a solution in this problem. Proximal gradient method is an iterative algorithm to solve the problem in (2.28), shown in Algorithm 1.

Algorithm 1 Proximal gradient method

Input: $\mathbf{x}^{(0)}, \gamma$
  while A stopping criterion is not satisfied **do**
    $\mathbf{x}^{(n+1)} = \text{prox}_{\gamma g}(\mathbf{x}^{(n)} - \gamma\nabla f(\mathbf{x}^{(n)}))$
    $n \leftarrow n + 1$
  end while
Output: $\mathbf{x}^{(n)}$

Theorem 2.3. *(Averaged nonexpansiveness of proximal gradient method) The sequential update* $\text{prox}_{\gamma g}(\mathbf{x}^{(n)} - \gamma\nabla f(\mathbf{x}^{(n)}))$ *in Algorithm 1 is an averaged nonexpansive operator for any* $\gamma \in (0, 2/\kappa)$.

Theorem 2.2 and 2.3 lead to the following theorem.

Theorem 2.4. *(Convergence of proximal gradient method) A sequence* $(\mathbf{x}^{(n)})_{n\geq 1}$ *generated from Algorithm 1 converges an optimal value of* (2.28) *for any* $\mathbf{x}^{(0)}$ *and any* $\gamma \in (0, 2/\kappa)$.

This theorem implies that the step size γ that guarantees convergence is determined by the Lipschitz constant κ of ∇f.

The proximal gradient method is often used to solve a problem with ℓ_1 norm, which is the important optimization in sparse modeling. Sparse modeling aims to obtain a sparse solution, meaning that most of the elements in the solution are

zero. We demonstrate a linear regression with ℓ_1 regularization as an example of the application of the proximal gradient method. Consider the following problem called as least absolute shrinkage and selection operator (LASSO):

$$\min_{\mathbf{x}\in\mathbb{R}^n} \frac{1}{2}\|\Phi\mathbf{x}-\mathbf{y}\|_2^2 + \alpha\|\mathbf{x}\|_1, \tag{2.29}$$

where $\Phi \in \mathbb{R}^{M\times N}$, $\mathbf{y} \in \mathbb{R}^M$, and $\alpha \in \mathbb{R}$ are given. The first term $\frac{1}{2}\|\Phi\mathbf{x}-\mathbf{y}\|_2^2$ is called as data fidelity term, which measure the difference between $\Phi\mathbf{x}$ and $\mathbf{y}$, the second term is a sparse term, which promotes the sparseness of $\mathbf{x}$, and α is the hyperparamter that controls the strength of the regularization.

Since the first term is a differentiable proper convex function, and the second term is convex but not differentiable, we assign (2.29) to the applicable form of the proximal gradient method in (2.28) as follows:

$$\begin{aligned} f(x) &= \frac{1}{2}\|\Phi\mathbf{x}-\mathbf{y}\|_2^2, \\ g(x) &= \alpha\|\mathbf{x}\|_1. \end{aligned} \tag{2.30}$$

The Lipschitz constant κ of ∇f is given by

$$\kappa = \|\Phi^\mathsf{T}\Phi\|_2 = \lambda_{\max}(\Phi^\mathsf{T}\Phi) \tag{2.31}$$

where $\|\cdot\|_2$ is matrix norm (also called spectral norm), which is given by the largest singular value of a matrix. The proximal operator of ℓ_1 norm is given by soft thresholding

$$\mathrm{prox}_{\gamma\|\cdot\|_1} = S_\gamma(\mathbf{x}) := \begin{cases} x_i-\gamma, & x_i \geq \gamma \\ 0, & -\gamma < x_i < \gamma \\ x_i+\gamma, & x_i \leq -\gamma. \end{cases} \tag{2.32}$$

Applying the proximal gradient method, (2.29) can be solved by the following iteration:

$$\mathbf{x}^{(n+1)} = S_\gamma(\mathbf{x}^{(n)} - \gamma\Phi^\mathsf{T}(\Phi\mathbf{x}^{(n)}-\mathbf{y})) \quad (n = 1, 2, \cdots). \tag{2.33}$$

for any $\mathbf{x}^{(0)}$ and $0 < \gamma \leq \frac{1}{\|\Phi^\mathsf{T}\Phi\|_2}$. This iterative algorithm is called as iterative shrinkage thresholiding algorithm (ISTA).

2.3.4 Primal-Dual Splitting Algorithm

Primal-dual approaches have been growing importance in recent developments in optimization. These approaches utilize the duality of the objective function, and they obtains an optimal solution by concurrently solving a primal problem (the original optimization problem) as well as a dual problem. The advantages of primal-dual approaches are summarized as follows:

Flexibility

Primal-dual approaches can handle both differentiable and nondifferentiable terms, the former using gradient operators and the latter using proximal operators.

Full splitting

Primal-dual approaches can use separately each of the operators contained in the problem (e.g., gradients, proximal operators, and linear operators). This enables to avoid the computation of the inversion operator, which is expensive computational cost, in the optimization process.

Parallelizable algorithm

Full splitting of primal-dual approaches leads to parallelizable algorithms. These parallelizable algorithms are often used for high-dimensional problems and distributed optimization problem.

We introduce the conjucate function, which is a fundamental notion in primal-dual approaches.

Definition 2.11. (Conjugate function) The conjugate (or Legendre transform, or Fenchel conjugate, or Legendre–Fenchel transform) of f is defined as

$$f^* : \mathcal{X} \to (-\infty, +\infty] : u \mapsto \sup_{x \in \mathcal{X}} (\langle x, u \rangle - f(x)). \tag{2.34}$$

$$\min_{x \in \mathcal{X}} f(x) + g(x) + \sum_{m=1}^{M} h_m(L_m x), \tag{2.35}$$

where $f, g \in \Gamma_0(\mathcal{X})$, $h_m \in \Gamma_0(\mathcal{U}_m)$, and $L_m : \mathcal{X} \to \mathcal{U}_m$ are bounded linear operators. f is differentiable on $\mathcal{X}$ and ∇f has a Lipschitz contant $\kappa > 0$. Suppose that the set of minimizers is nonempty, and the following is satisfied:

$$\begin{aligned}(0, 0, \cdots, 0) \in \mathrm{sri}\{(L_m x = u_m)_{1 \le m < M} \mid x \in \mathrm{dom}(g) \\ \text{and } u_m \in \mathrm{dom}(h_m), \forall m = 1, \cdots M\}\end{aligned} \tag{2.36}$$

The algorithm of the primal-dual splitting are summarized in Algorithm 2.

Algorithm 2 Primal-dual splitting algorithm

Input: $x^{(0)} \in \mathcal{X}, u_1^{(0)} \in \mathcal{U}_1, \cdots u_M^{(0)} \in \mathcal{U}_M, \gamma_1, \gamma_2$
  while A stopping criterion is not satisfied **do**
    $x^{(n+1)} := \text{prox}_{\gamma_1 g}\left(x^{(n)} - \gamma_1 \nabla f(x^{(n)}) - \gamma_1 \sum_{m=1}^{M} L_m^* u_m^{(n)}\right)$
    for $m = 1$ to M **do**
      $u_m^{(n+1)} := \text{prox}_{\gamma_2 h_m^*}\left(u_m^{(n)} + \gamma_2 L_m (2x^{(n+1)} - x^{(n)})\right)$
    end for
  end while
Output: $x^{(n)}$

Theorem 2.5. *(Convergence of primal-dual splitting algorithm) The sequence* $(x^{(n)})_{n\in\mathbb{N}}$ *generated by Algorithm 2 converges to an solution* $\hat{x} \in \mathcal{X}$ *of the problem (2.35) if*

$$\gamma_1 \left(\frac{\kappa}{2} + \gamma_2 \left\| \sum_{m=1}^{M} L_m^* L_m \right\|\right) < 1. \tag{2.37}$$

Chapter 3

Graph Learning from a Signal Generation Perspective

3.1 Graph Learning based on Factor Analysis

3.1.1 Static Graph Factor Analysis

We describe the static graph factor analysis (SGFA) [38], which introduces a model for the generation of graph signals. The observation model of SGFA is defined as follows [38]:

$$\mathbf{x} = \mathbf{U}\mathbf{h} + \boldsymbol{\epsilon}, \tag{3.1}$$

where $\mathbf{x} \in \mathbb{R}^N$ is an observed signal, $\mathbf{h} \in \mathbb{R}^N$ is a latent variable represented in the graph frequency domain, $\mathbf{U}$ is the GFT matrix and $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \sigma_\epsilon^2 \mathbf{I})$ is an additive white Gaussian noise. The observation model in (3.1) means that the signals possessing graph structures can be represented by the inverse GFT of latent variables in the spectral domain of the underlying graph. SGFA assumes that the latent variable $\mathbf{h}$ follows the multivariate Gaussian distribution

$$p(\mathbf{h}) = \mathcal{N}(0, \boldsymbol{\Lambda}^\dagger). \tag{3.2}$$

This corresponds to the assumption that signal energy tends to be concentrated in the low frequencies and thus encourages graph signal smoothness. Equations (3.1) and (3.2) lead to the conditional probability of $\mathbf{x}$ given $\mathbf{h}$:

$$p(\mathbf{x}|\mathbf{h}) = \mathcal{N}(\mathbf{U}\mathbf{h}, \sigma_\epsilon^2 \mathbf{I}). \tag{3.3}$$

From (3.2) and (3.3), the marginal distribution of $\mathbf{x}$ is given by:

$$p(\mathbf{x}) = \mathcal{N}(0, \mathbf{L}^\dagger + \sigma_\epsilon^2 \mathbf{I}). \tag{3.4}$$

Note that the fact $\mathbf{L}^\dagger = \mathbf{U}\mathbf{\Lambda}^\dagger\mathbf{U}^\top$ is used in the derivation of (3.4). The marginal distribution in (3.4) indicates that a graph signal $\mathbf{x}$ can be generated by the degenerate multivariate Gaussian distribution whose precision matrix is the graph Laplacian of the underlying graph.

Signals generated from this distribution in (3.4) satisfy graph stationary [65] because their covariance and graph Laplacian can be jointly diagonalizable with the eigenvectors of graph Laplacian. Consider the maximum a posteriori estimation of $\mathbf{h}$:

$$\begin{aligned} h_{\mathrm{MAP}}(\mathbf{x}) &:= \arg\max_{\mathbf{h}} p(\mathbf{h}|\mathbf{x}) \\ &= \arg\max_{\mathbf{h}} p(\mathbf{x}|\mathbf{h})p(\mathbf{h}) \\ &= \arg\min_{\mathbf{h}} \left(-\log e^{-(\mathbf{x}-\mathbf{U}\mathbf{h})^\top(\mathbf{x}-\mathbf{U}\mathbf{h})} - \alpha \log e^{-\mathbf{h}^\top\mathbf{\Lambda}\mathbf{h}} \right) \\ &= \arg\min_{\mathbf{h}} \|\mathbf{x}-\mathbf{U}\mathbf{h}\|_2^2 + \alpha\mathbf{h}^\top\mathbf{\Lambda}\mathbf{h} \end{aligned} \tag{3.5}$$

where α is a parameter, which is proportional to the noise variance σ_ϵ^2. Substituting $\mathbf{y} = \mathbf{U}\mathbf{h}$ for (3.5) leads to the following problem:

$$\min_{\mathbf{L},\mathbf{y}} \ \|\mathbf{x}-\mathbf{y}\|_2^2 + \alpha\mathbf{y}^\top\mathbf{L}\mathbf{y}, \tag{3.6}$$

where $\mathbf{y}$ is regarded as a noiseless signal. Focusing on the second term in (3.6), this optimization problem finds the graph Laplacian that minimizes the smoothness measure as in (2.9).

3.1.2 Graph Learning for Smooth Signal Representation

Based on (3.6), graph learning problem is formulated as [38]:

$$\begin{aligned} \min_{\mathbf{L},\mathbf{Y}} \|\mathbf{X}-\mathbf{Y}\|_F^2 + \alpha\,\mathrm{tr}(\mathbf{Y}^\top\mathbf{L}\mathbf{Y}) + \beta\|\mathbf{L}\|_F^2. \\ \text{s.t. } \mathrm{tr}(\mathbf{L}) = N \\ L_{ij} = L_{ji} \le 0, i \ne j \\ \mathbf{L}\mathbf{1} = \mathbf{0} \end{aligned} \tag{3.7}$$

where $\mathbf{X} \in \mathbb{R}^{N\times K}$ contains K observed data $\{\mathbf{x}_k\}_{k=1}^K$ as columns, and α and β are regularization hyperparameters. The third term of (3.6) added as a regularization term in the objective function controls the distribution of the the off-diagonal entries of $\mathbf{L}$. The first constraint $\mathrm{tr}(\mathbf{L}) = N$ acts to avoid trivial solutions, and the second and third terms are constraints to learn a valid graph Laplacian.

The problem of (3.6) can be solved by an alternative optimization that consists of the steps to update $\mathbf{L}$ and $\mathbf{Y}$. Alternative optimization fixes one variable and solves the subproblem for the other variable at each step and iterates the steps to obtain a local minimum solution. From (3.6) fixed $\mathbf{Y}$, the step to update $\mathbf{L}$ is given by:

$$\begin{aligned} &\min_{\mathbf{L}} \ \alpha \operatorname{tr}(\mathbf{Y}^\mathsf{T}\mathbf{L}\mathbf{Y}) + \beta\|\mathbf{L}\|_F^2. \\ &\text{s.t. } \operatorname{tr}(\mathbf{L}) = N, \ L_{ij} = L_{ji} \leq 0 \ (\ i \neq j), \ \mathbf{L}\mathbf{1} = \mathbf{0} \end{aligned} \tag{3.8}$$

This problem finds a graph Laplacian where the noiseless signals $\mathbf{X}$ are smooth, and can be regarded as the graph learning step. This problem can be solved by operator splitting methods (e.g., alternating direction method of multipliers (ADMM) and primal-dual splitting method). The second step to update $\mathbf{Y}$ is given by the following problem:

$$\min_{\mathbf{L}} \|\mathbf{X} - \mathbf{Y}\|_F^2 + \alpha \operatorname{tr}(\mathbf{Y}^\mathsf{T}\mathbf{L}\mathbf{Y}) \tag{3.9}$$

This problem has the following closed form solution:

$$\mathbf{Y} = (\mathbf{I} + \alpha\mathbf{L})^{-1}\mathbf{X}. \tag{3.10}$$

This corresponds to applying a low pass graph filter to $\mathbf{X}$ and can be interpreted as denoising of $\mathbf{X}$. Thus, the solution of (3.6) is obtained by solving alternately (3.8) and (3.9).

3.1.3 Kalofolias's method

The alternative optimization such as the above problem cannot guarantee the convergence to the optimal solution in general. To avoid this, by assuming $\mathbf{y} = \mathbf{x}$, (3.6) is reformulated as a problem to find the weighted adjacency matrix that minimizes the smoothness measure $\mathbf{x}^\mathsf{T}\mathbf{L}\mathbf{x}$ and some constraint terms [37,43,54]:

$$\min_{\mathbf{W} \geq 0} \|\mathbf{W} \circ \mathbf{Z}\|_1 - \alpha \mathbf{1}^\mathsf{T} \log(\mathbf{W}\mathbf{1}) + \beta\|\mathbf{W}\|_F^2, \tag{3.11}$$

where $\mathbf{Z}$ is a pairwise distance matrix given by

$$(\mathbf{Z})_{ij} = \sum_{k=1}^{K} \|(\mathbf{x}_k)_i - (\mathbf{x}_k)_j\|^2. \tag{3.12}$$

The first term corresponds to the smoothness metric in (2.9):

$$2\mathrm{Tr}(\mathbf{X}^\top \mathbf{L}_t \mathbf{X}) = \mathrm{Tr}(\mathbf{W}\mathbf{Z}) = \|\mathbf{W} \circ \mathbf{Z}\|_1. \tag{3.13}$$

The second term is a log-barrier function for the degree of the graph, which forces the degree on each node to be positive without preventing edge weights from becoming zero. This problem can be solved using a primal-dual splitting algorithm.

3.2 Graph Learning based on Laplacian-Constrained Gaussian Markov Random Fields

Graph learning is a problem of learning graph Laplacian(s) from observations $\mathbf{X} \in \mathbb{R}^{N\times K} = [\mathbf{x}_1, \dots \mathbf{x}_K]$, where K is the number of observations. Assume that the following signal observation model based on Laplacian constrained Gaussian Markov random field (LGMRF) [38, 39, 45]:

$$p(\mathbf{x} \mid \mathbf{\Theta}) = \frac{1}{(2\pi)^{K/2} \left(\mathrm{gdet}(\mathbf{\Theta}^\dagger)\right)^{1/2}} \exp\left(-\frac{1}{2}\mathbf{x}^T \mathbf{\Theta} \mathbf{x}\right), \tag{3.14}$$

where $\mathbf{\Theta} \in \mathcal{L}$ is the precision matrix satisfying graph Laplacian constraints. The negative log-likelihood function $L(\mathbf{\Theta})$ of (3.14) is given by

$$\begin{aligned} L(\mathbf{\Theta}) &= -\log(\prod_{k=1}^{K} p\left(\mathbf{x}_k \mid \mathbf{\Theta}\right)) \\ &= \frac{1}{2}\sum_{k=1}^{K} \mathrm{Tr}\left(\mathbf{x}_k^\top \mathbf{\Theta} \mathbf{x}_k\right) - \frac{K}{2}\log \mathrm{gdet}(\mathbf{\Theta}). \end{aligned} \tag{3.15}$$

Furthermore, suppose that the prior distribution $p(\mathbf{\Theta})$ is the following Laplace distribution.

$$p(\mathbf{\Theta} \mid \mathbf{\Lambda}) = \prod_{i<j} p(\theta_{ij}) = \prod_{i<j} \frac{\lambda}{2} \exp(-\lambda|\theta_{ij}|), \tag{3.16}$$

where λ is a scale parameter of the Laplace distribution. The maximum a posteriori (MAP) estimation of $\mathbf{\Theta}$ with (3.15) and (3.16) leads to the following the optimization problem [39]:

$$\underset{\mathbf{\Theta}\in\mathcal{L}}{\text{minimize}} \ \frac{1}{K}\mathrm{Tr}(\mathbf{\Theta}\mathbf{S}) - \log \mathrm{gdet}(\mathbf{\Theta}) + \alpha\|\mathbf{\Theta}\|_{1,\text{off}} \tag{3.17}$$

where $\mathbf{S} = \mathbf{X}\mathbf{X}^\mathsf{T}$, $\alpha = \lambda/K$, $\|\mathbf{\Theta}\|_{1,\text{off}}$ represents the absolute sum of off-diagonal elements in $\mathbf{\Theta}$, and $\mathcal{L}$ is the set of valid graph Laplacians given by:

$$\mathcal{L} = \left\{ \mathbf{L} \in \mathbb{R}^{N \times N} : L_{ij} = L_{ji} \leq 0 \ (i \neq j), L_{ii} = \sum_{i \neq j} L_{ij} \right\}. \tag{3.18}$$

The optimization problem in (3.17) can be solved using the block coordinate descent algorithm [39].

Chapter 4

Time-varying Graph Learning with Constraints on Graph Temporal Variation

This chapter presents a *time-varying graph learning* method based on time-varying graph factor analysis (TGFA), which is an extension of its static counterpart, static graph factor analysis (SGFA) [38]. We propose the TGFA-based method to estimate time-varying graphs from a collection of spatiotemporal measurements. The SGFA formulates a signal generation model based on a graph signal processing (GSP) perspective, where it is assumed that the observed signals have some specific spectral properties with respect to graph Fourier transform (GFT) of the graph to be learned. For example, if a multivariate Gaussian model is chosen, it leads to the observed signals generated from a Gaussian distribution whose inverse covariance matrix (i.e., precision matrix) is given by the graph Laplacian of the underlying graph [30,38]. Unlike SGFA, TGFA considers the graph evolution as illustrated in Fig. 4.1. The graph evolution can be represented by a sequence of graph Laplacians and their corresponding temporal variations.

This study focuses on two time-varying graph models, with the following two properties:

(P1) Temporal homogeneity: Most edges and their weights in the time-varying graph should remain unchanged over a short-term time horizon. In other words, at any given time, only a small number of edges in time-varying graphs change. Time-varying graphs in many applications satisfy this property. For example, consider a sensor network where nodes and edges represent sensor locations and correlations among sensor measurements, respectively. If the sensors record the temperature in a building, various factors such as air conditioning, sunlight, and the number of people in the room, locally affect the correlations among the sensor measurements. However, these factors vary

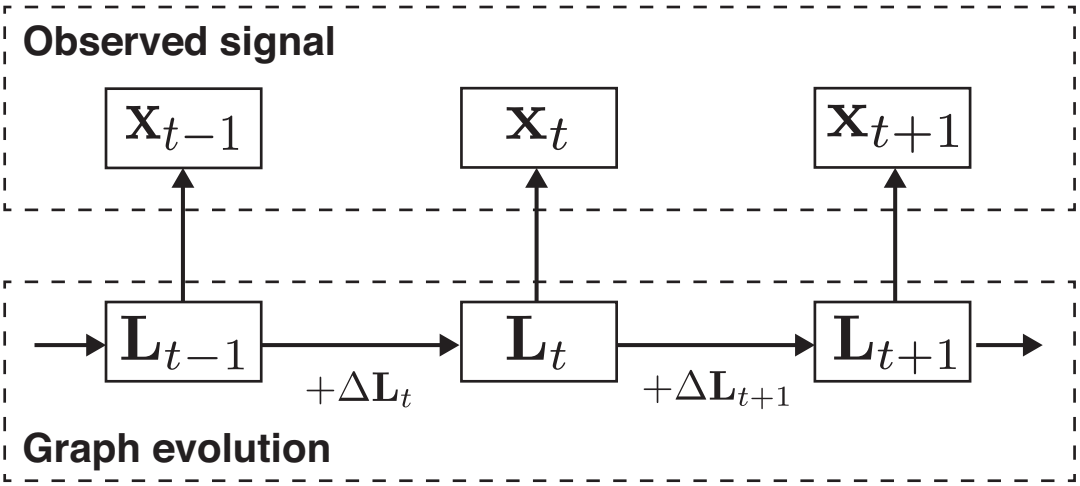


FIGURE 4.1: An overview of time-varying graph factor analysis. $\mathbf{L}_t$ and $\Delta\mathbf{L}_t$ represent the graph Laplacian at the tth time slot and the graph temporal variation. This study focuses on learning a time-varying graph, which is the sequence of the graph Laplacian, from the observed signal $\mathbf{x}$.

smoothly over time. As a result, this sensor network will be a time-varying graph such that most edges remain constant, while the weights change only slightly over time, i.e., it follows **(P1)**. In addition to this example, time-varying graphs in fMRI and various biological networks seem to have this property [47, 66].

(P2) Switching behavior: Edges and weights remain almost unchanged over time; however, they may suddenly change within a few time slots. This type of time-varying graph appears in situations where some factors cause sudden changes in graph topologies. Prominent examples include brain networks, where epileptic seizures make their topology change suddenly [67].

We design an algorithm to estimate the two types of time-varying graphs, namely, graphs with temporal homogeneity and graphs with switching behavior. For this purpose, we formulate the graph learning problem as a convex optimization with regularization of temporal variation derived from TGFA. To solve the convex optimization problem, we utilize a primal-dual splitting algorithm [68], which enables us to estimate time-varying graphs more successfully than static graph learning methods.

4.1 Time-varying Graph Factor Analysis

Suppose that we have a multivariate time series data divided by non-overlapping time windows $\{\mathbf{X}_1, \ldots, \mathbf{X}_T\}$, where $\mathbf{X}_t = [\mathbf{x}_1^{(t)}, \ldots, \mathbf{x}_K^{(t)}] \in \mathbb{R}^{N \times K}$ contains K observations at a time window t, and the graph corresponding to observations in the same time window is invariant. The choice of K depends on a sampling

frequency in measurements. In this paper, for simplicity we use nonoverlapping time windows to formulate the observation model for simplicity. Note that the overlapping sliding windows can also be used instead, with slight modifications. Our goal is to learn a sequence of graph Laplacians $\mathbf{L}_1, \ldots, \mathbf{L}_T$ from the data sequence. We first introduce a time-varying graph factor analysis (TGFA) to formulate the time-varying graph learning. By incorporating a graph evolution process into SGFA, we define TGFA as:

$$\mathbf{x}^{(t)} \sim \mathcal{N}(0, \mathbf{L}_t^{\dagger} + \sigma_{\epsilon}^2 \mathbf{I}), \tag{4.1}$$

$$\mathbf{L}_t = \begin{cases} \mathbf{0} & t = 0 \\ \mathbf{L}_{t-1} + \Delta\mathbf{L}_t & t \geq 1. \end{cases} \tag{4.2}$$

where $\mathbf{x}^{(t)}$, $\mathbf{L}_t$, and $\Delta\mathbf{L}_t$ are a signal, the graph Laplacian of the underlying graph and a temporal graph variation at a given time t, respectively. In Section 3.1, we discuss (4.1), which indicates that time-varying graph learning requires the time-varying observations to be smooth on the learned graphs at the corresponding time slots.

Besides the smoothness criterion, this approach allows us to incorporate prior knowledge about temporal graph variation and will lead to more robust graph learning. Thus, we generalize the time-varying graph learning problem as that of solving the following optimization:

$$\min_{\mathbf{L}_t \in \mathcal{L}} \sum_{t=1}^{T} \mathrm{Tr}(\mathbf{X}_t^{\mathsf{T}} \mathbf{L}_t \mathbf{X}_t) + f(\mathbf{L}_t) + \eta \sum_{t=2}^{T} R(\Delta\mathbf{L}_t \circ \mathbf{H}), \tag{4.3}$$

where $\mathcal{L}$ is the valid set of graph Laplacians and is given by

$$\begin{aligned} \mathcal{L} = \{\ \mathbf{L} \in \mathbb{R}^{N \times N}\ | \mathbf{L} = \mathbf{L}^{\mathsf{T}}, L_{ij} \leq 0 \quad (i \neq j), \\ L_{ii} = -\sum_{j \neq i} L_{ij} \}, \end{aligned} \tag{4.4}$$

$\mathbf{H} = \mathbf{I} - \mathbf{1}\mathbf{1}^{\mathsf{T}}$, $R(\Delta\mathbf{L}_t \circ \mathbf{H})$ is a regularization term that characterizes the temporal change in graph edges, that is, the off-diagonal elements of the temporal graph variation $\Delta\mathbf{L}_t$. The first term in (4.3) corresponds to the smoothness term in the static case (3.6), and quantifies the smoothness of the signals on time-varying graphs. The function in the second term $f(\mathbf{L}_t)$ is a regularization to avoid obtaining trivial solutions. This function, of which examples will be given later, depends on the assumptions made about the graph model. The parameter η controls the regularization strength.

We reformulate the problem (4.3) with a constraint (4.4) by using weighted adjacency matrices. As shown in the following, this leads to a tractable formulation because using weighted adjacency matrices allows us to simplify the condition for the valid set. A problem equivalent to that of (4.3) is then given by:

$$\min_{\mathbf{W}_t \in \mathcal{W}_m} \sum_{t=1}^{T} \frac{1}{2} \|\mathbf{W}_t \circ \mathbf{Z}_t\|_1 + f(\mathbf{W}_t) + \eta \sum_{t=2}^{T} R(\mathbf{W}_t - \mathbf{W}_{t-1}), \tag{4.5}$$

where the valid set of weighted adjacency matrices is defined by

$$\mathcal{W}_m = \left\{ \mathbf{W} \in \mathbb{R}_+^{N \times N} \mid \mathbf{W} = \mathbf{W}^\mathsf{T}, W_{ii} = 0 \right\}. \tag{4.6}$$

The first term of (4.5) is equivalent to $\mathrm{Tr}(\mathbf{X}_t^\mathsf{T} \mathbf{L}_t \mathbf{X}_t)$, given that [37]:

$$\mathrm{Tr}(\mathbf{X}_t^\mathsf{T} \mathbf{L}_t \mathbf{X}_t) = \frac{1}{2} \mathrm{Tr}(\mathbf{W}_t \mathbf{Z}_t) = \frac{1}{2} \|\mathbf{W}_t \circ \mathbf{Z}_t\|_1, \tag{4.7}$$

where $\mathbf{Z}_t$ is the pairwise distance matrix defined by

$$(\mathbf{Z}_t)_{ij} = \sum_{k=1}^{K} \|(\mathbf{x}_k^{(t)})_i - (\mathbf{x}_k^{(t)})_j\|^2. \tag{4.8}$$

In this paper, we solve the optimization problem in (4.5) to learn a time-varying graph. Throughout this paper, we set the regularization for the weighted adjacency matrix to be [54]:

$$f(\mathbf{W}_t) = -\alpha \mathbf{1}^\top \log(\mathbf{W}_t \mathbf{1}) + \beta ||\mathbf{W}_t||_F^2, \tag{4.9}$$

where α and β are the parameters. The first term in (4.9) forces the degree on each node to be positive without preventing edge weights from becoming zero. The second term controls the sparsity of the resulting graphs. Lower value leads to sparser graphs, and higher value leads to denser graphs. Next, we describe how to select the regularization for the graph temporal variation $R(\cdot)$ and solve the optimization problem.

4.2 Learning Time-Varying Graph with Temporal Variation Regularization

The choice of the regularization term for graph temporal variation should be based on our prior knowledge, i.e., the assumption about the temporal graph

evolution. In [54], a temporal variation regularizer $R(\cdot) = \|\cdot\|_F^2$ is selected to learn time-varying graphs such that the edge weights change smoothly over time. In this paper, we consider two types of graph evolution terms different from [54], which appear in many applications. Because our regularization terms are not differentiable, the optimization problem in our approach cannot be solved directly using the same methods as in [54]. Therefore, we also present an algorithm for solving our optimization problem.

4.2.1 Regularization of Graph Temporal Variation

Formulation with Fused Lasso

In the first model, we assume that most edges and their weights are likely to remain unchanged over a short-term time horizon **(P1)**. Thus, we select a regularizer $R(\cdot) = \|\cdot\|_1$ in (4.5) and formulate the optimization problem as:

$$\min_{\mathbf{W}_t \in \mathcal{W}_m} \sum_{t=1}^{T} \frac{1}{2}\|\mathbf{W}_t \circ \mathbf{Z}_t\|_1 + f(\mathbf{W}_t) + \eta \sum_{t=2}^{T} \|\mathbf{W}_t - \mathbf{W}_{t-1}\|_1, \tag{4.10}$$

A penalty on ℓ_1-norm of the difference between neighboring time windows leads to a *fused Lasso* problem [69], which encourages sparse differences, leading to local temporal constancy of weighted adjacency matrices, and thus, solution graphs that tend to satisfy the **(P1)** property. This penalty can also be interpreted as the total variation between neighboring graphs.

In order to make the optimization problem tractable, we rewrite (4.10) and (4.9) in vector form, with $\mathbf{W}_t$ and $\mathbf{Z}_t$ replaced by $\mathbf{w}_t \in \mathbb{R}_+^{N(N-1)/2}$ and $\mathbf{z}_t \in \mathbb{R}_+^{N(N-1)/2}$ respectively. Only the upper-triangular parts of $\mathbf{W}_t$ and $\mathbf{Z}_t$ are considered given that the graph is undirected.

Let us define $\mathbf{w} = [\mathbf{w}_1^\mathsf{T}\ \mathbf{w}_2^\mathsf{T} \dots \mathbf{w}_T^\mathsf{T}]^\mathsf{T}$, $\mathbf{z} = [\mathbf{z}_1^\mathsf{T}\ \mathbf{z}_2^\mathsf{T} \dots \mathbf{z}_T^\mathsf{T}]^\mathsf{T}$, and $\mathbf{d} = [\mathbf{d}_1^\mathsf{T}\ \mathbf{d}_2^\mathsf{T} \dots \mathbf{d}_T^\mathsf{T}]^\mathsf{T}$ where $\mathbf{d}_t = \mathbf{W}_t\mathbf{1}$. We also introduce the linear operators $\mathbf{S}$ and $\mathbf{\Phi}$ that satisfy $\mathbf{S}\mathbf{w} = \mathbf{d}$ and $\mathbf{\Phi}\mathbf{w} = \mathbf{w} - \hat{\mathbf{w}}$ respectively, where $\hat{\mathbf{w}} = [\mathbf{w}_1^\mathsf{T}\ \mathbf{w}_1^\mathsf{T}\ \mathbf{w}_2^\mathsf{T} \dots \mathbf{w}_{T-1}^\mathsf{T}]^\mathsf{T}$.

Then, we can rewrite (4.10) and (4.9) as

$$\min_{\mathbf{w} \in \mathcal{W}_v} \mathbf{z}^\mathsf{T}\mathbf{w} - \alpha\mathbf{1}^\mathsf{T}\log(\mathbf{S}\mathbf{w}) + \beta\|\mathbf{w}\|_2^2 + \eta\|\mathbf{\Phi}\mathbf{w}\|_1, \tag{4.11}$$

where

$$\mathcal{W}_v = \left\{\mathbf{w} \in \mathbb{R}^{TN(N-1)/2} \mid w_i \geq 0\ (i = 1, 2, \dots)\right\} \tag{4.12}$$

represents the nonnegativity constraint. By expressing (4.6) in vector form, the symmetric and diagonal constraints of (4.6) can be simplified.

Formulation with Group Lasso

The second model aims at learning graphs having the switching behavior **(P2)**. This is achieved by choosing $R(\cdot) = \|\cdot\|_2$ as the regularizer in (4.5), leading to a *group Lasso* [70] penalty as follows:

$$\min_{\mathbf{W}_t \in \mathcal{W}_m} \sum_{t=1}^{T} \frac{1}{2}\|\mathbf{W}_t \circ \mathbf{Z}_t\|_1 + f(\mathbf{W}_t) + \eta \sum_{t=2}^{T} \|\mathbf{W}_t - \mathbf{W}_{t-1}\|_F, \tag{4.13}$$

where only the last term differs from (4.10). Group Lasso penalizes the sum of the ℓ_2-norm, leading to a $\ell_{2,1}$-norm that encourages sparsity in each group. In this formulation, the temporal variation of the weighted adjacency matrices at a certain time slot is regarded as one group. Thus the group Lasso yields graphs whose edge weights can vary significantly at a few time slots while they remain almost constant at all the other time slots, thus satisfying the **(P2)** property.

In order to make the optimization problem tractable, we rewrite (4.13) in the same manner as in the previous model as:

$$\min_{\mathbf{w} \in \mathcal{W}_v} \mathbf{z}^\mathsf{T}\mathbf{w} - \alpha \mathbf{1}^\mathsf{T} \log(\mathbf{S}\mathbf{w}) + \beta\|\mathbf{w}\|_2^2 + \eta \sum_j \|[\mathbf{\Phi}\mathbf{w}]_j\|_2, \tag{4.14}$$

where $[\mathbf{\Phi}\mathbf{w}]_j = \mathbf{w}_j^\mathsf{T} - \mathbf{w}_{j-1}^\mathsf{T}$, $j = 2, \dots T$.

4.2.2 Optimization

We solve (4.10) and (4.13) with a primal-dual splitting (PDS) method [68], which solves a problem in the form,

$$\min_{\mathbf{w}} f_1(\mathbf{w}) + f_2(\mathbf{w}) + f_3(\mathbf{M}\mathbf{w}), \tag{4.15}$$

where f_1 is a differentiable convex function with gradient ∇f_1 and Lipschitz constant ξ; f_2 and f_3 are proper lower semi-continuous convex functions which are proximable; and $\mathbf{M}$ is a linear operator.

Many algorithms to solve (4.15) have been proposed [68, 71, 72]. In this proposed method, we use a forward-backward-forward (FBF) approach [73], which makes it possible to compute the proximal operators of functions f_2 and f_3 in parallel.

Fused Lasso

By introducing the indicator function of $\mathcal{W}_v$ in (4.12), we rewrite (4.11) as follows:

$$\min_{\mathbf{w}} \ \mathbf{z}^{\mathsf{T}}\mathbf{w} - \alpha\mathbf{1}^{\mathsf{T}}\log(\mathbf{S}\mathbf{w}) + \beta\|\mathbf{w}\|_2^2 + \eta\|\mathbf{\Phi}\mathbf{w}\|_1 + \iota_{\mathcal{W}_v}(\mathbf{w}), \tag{4.16}$$

where the indicator function $\iota_{\mathcal{W}_v}$ is defined by

$$\iota_{\mathcal{W}_v}(\mathbf{w}) = \begin{cases} 0 & w_i \geq 0 \\ \infty & \text{otherwise.} \end{cases} \tag{4.17}$$

The objective function (4.16) further reduces to the applicable form of the PDS as follows:

$$\begin{aligned} f_1(\mathbf{w}) &= \beta\|\mathbf{w}\|_2^2 \ \text{ with } \xi = 2\beta, \\ f_2(\mathbf{w}) &= \mathbf{z}^{\mathsf{T}}\mathbf{w} + \iota_{\mathcal{W}_v}(\mathbf{w}), \\ f_3(\mathbf{v}) &= -\alpha\mathbf{1}^{\mathsf{T}}\log(\mathbf{v}_1) + \eta\|\mathbf{v}_2\|_1, \\ \mathbf{M} &= \begin{bmatrix} \mathbf{S} \\ \mathbf{\Phi} \end{bmatrix}, \end{aligned} \tag{4.18}$$

where the dual variable is $\mathbf{v} := \mathbf{M}\mathbf{w} = [\mathbf{v}_1^{\mathsf{T}} \ \mathbf{v}_2^{\mathsf{T}}]^{\mathsf{T}}$.

The proximal operator of f_2 is given by

$$\left(\text{prox}_{\gamma(\|\cdot\|_1 + \iota_{\mathcal{W}_v})}(\mathbf{x})\right)_i = \begin{cases} 0 & x_i \leq \gamma \\ x_i - \gamma & \text{otherwise.} \end{cases} \tag{4.19}$$

Because f_3 is separable across variables, the proximal operator can be computed separately for each term (see (2.24)). The proximal operator for the log barrier function [64] as the first term of f_3 is given by

$$\left(\text{prox}_{\gamma(-\mathbf{1}^{\mathsf{T}}\log(\cdot))}(\mathbf{x})\right)_i = \frac{x_i + \sqrt{x_i^2 + 4\gamma}}{2}. \tag{4.20}$$

The proximal operator for the ℓ_1-norm, i.e., the second term in f_3, is well known to be the element-wise soft-thresholding operation [64]:

$$\left(\text{prox}_{\gamma\|\cdot\|_1}(\mathbf{x})\right)_i = \begin{cases} 0 & |x_i| \leq \gamma \\ \text{sgn}(x_i)(|x_i| - \gamma) & \text{otherwise.} \end{cases} \tag{4.21}$$

See Algorithm 3 for the complete algorithm.

Group Lasso

Similarly, the second objective function (4.14) can be reduced to the applicable form of the PDS by replacing f_3 with

$$f_3(\mathbf{v}) = -\alpha \mathbf{1}^\mathsf{T} \log(\mathbf{v}_1) + \eta \sum_j \|(\mathbf{v}_2)_j\|_2, \tag{4.22}$$

where $[\mathbf{v}_2]_j = [\mathbf{\Phi w}]_j$. The proximal operator of the group Lasso in f_3 is well known to be the blockwise soft-thresholding operation [64] :

$$\text{prox}_{\gamma\|\cdot\|_2}([\mathbf{x}]_j) = \begin{cases} \mathbf{0} & \|[\mathbf{x}]_j\|_2 \leq \gamma \\ (1 - \gamma/\|[\mathbf{x}]_j\|_2)([\mathbf{x}]_j) & \text{otherwise.} \end{cases} \tag{4.23}$$

Because the proximal operator of the functions f_2 and f_3 can be computed efficiently, the second model can also be solved with the PDS algorithm. See Algorithm 3.

The computational complexity for both of our learning problems is $\mathcal{O}(TN^2)$ per iteration. The hyperparameter γ in the PDS algorithm is determined such that the convergence condition [71] is satisfied. Since the optimization problem in the proposed method is convex and lower-semicontinuous, convergence of the algorithms is guaranteed.

Algorithm 3 Proposed PDS Algorithm for solving (4.11) and (4.14)

Input: $\mathbf{w}^{(0)}, \mathbf{v}_1^{(0)}, \mathbf{v}_2^{(0)}$
Output: $\mathbf{w}^{(i)}$
while $\|\mathbf{w}^{(i+1)} - \mathbf{w}^{(i)}\|/\|\mathbf{w}^{(i)}\| > \epsilon$ **do**

$$\mathbf{y}^{(i)} = \mathbf{w}^{(i)} - \gamma(2\beta\mathbf{w}^{(i)} + \mathbf{S}^\mathsf{T}\mathbf{v}_1^{(i)} + \mathbf{\Phi}^\mathsf{T}\mathbf{v}_2^{(i)})$$
$$\bar{\mathbf{y}}_1^{(i)} = \mathbf{v}_1^{(i)} + \gamma(\mathbf{S}\mathbf{w}^{(i)})$$
$$\bar{\mathbf{y}}_2^{(i)} = \mathbf{v}_2^{(i)} + \gamma(\mathbf{\Phi}\mathbf{w}^{(i)})$$
$$\mathbf{p}^{(i)} = \text{prox}_{\gamma(\|\cdot\|_1 + \iota_{\mathcal{W}_v})}(\mathbf{y}^{(i)})$$
$$\bar{\mathbf{p}}_1^{(i)} = \bar{\mathbf{y}}_1^{(i)} - \gamma\text{prox}_{\frac{1}{\gamma}(-\mathbf{1}^\mathsf{T}\log(\cdot))}\left(\frac{\bar{\mathbf{y}}_1^{(i)}}{\gamma}\right)$$
$$\bar{\mathbf{p}}_2^{(i)} = \begin{cases} \bar{\mathbf{y}}_2^{(i)} - \gamma\text{prox}_{\frac{1}{\gamma}\|\cdot\|_1}\left(\frac{\bar{\mathbf{y}}_2^{(i)}}{\gamma}\right) & \text{for solving (4.11)} \\ \bar{\mathbf{y}}_2^{(i)} - \gamma\text{prox}_{\frac{1}{\gamma}\|\cdot\|_2}\left(\frac{\bar{\mathbf{y}}_2^{(i)}}{\gamma}\right) & \text{for solving (4.14)} \end{cases}$$
$$\mathbf{q}^{(i)} = \mathbf{p}^{(i)} - \gamma(2\beta\mathbf{p}^{(i)} + \mathbf{S}^\mathsf{T}\bar{\mathbf{p}}_1^{(i)} + \mathbf{\Phi}^\mathsf{T}\bar{\mathbf{p}}_2^{(i)})$$
$$\bar{\mathbf{q}}_1^{(i)} = \bar{\mathbf{p}}_1^{(i)} + \gamma(\mathbf{S}\mathbf{p}^{(i)})$$
$$\bar{\mathbf{q}}_2^{(i)} = \bar{\mathbf{p}}_2^{(i)} + \gamma(\mathbf{\Phi}\mathbf{p}^{(i)})$$
$$\mathbf{w}^{(i+1)} = \mathbf{w}^{(i)} - \mathbf{y}^{(i)} + \mathbf{q}^{(i)}$$
$$\mathbf{v}_1^{(i+1)} = \mathbf{v}_1^{(i)} - \bar{\mathbf{y}}_1^{(i)} + \bar{\mathbf{q}}_1^{(i)}$$
$$\mathbf{v}_2^{(i+1)} = \mathbf{v}_2^{(i)} - \bar{\mathbf{y}}_2^{(i)} + \bar{\mathbf{q}}_2^{(i)}$$

end while

4.3 Experimental Results with Synthetic Dataset

TABLE 4.1: List of alternative and proposed methods

	Method	Temporal Regularization	Complexity
Static	SGL-GLasso [39]	-	$\mathcal{O}(T(N^2+\Omega(N^3)))$
	SGL-Smooth [37]	-	$\mathcal{O}(TN^2)$
Time Varying	TGL-Tikhonov [54]	Tikhonov	$\mathcal{O}(TN^2)$
	TGL-TH (proposed)	Fused Lasso	$\mathcal{O}(TN^2)$
	TGL-SB (proposed)	Group Lasso	$\mathcal{O}(TN^2)$

In this section, we present the experimental results with synthetic data and compare the performance of our proposed method with related graph learning methods (see Table 4.1 for a summary of all the methods we compare).

4.3.1 Synthetic Datasets

We create three types of synthetic graph signals generated from time-varying graphs: A time-varying graph based on random waypoint model (abbreviated as TV-RW graph) [74], a temporal homogeneity-based Erdős–Rényi graph (TH-ER graph), and a

The dataset construction consists of two steps:

1. Constructing a time-varying graph.
2. Generating time-varying graph signals from probability distributions based on graph Laplacians of the created time-varying graph.

We then describe the details of three datasets by referring to the desired properties **(P1)** and **(P2)**.

Time-Varying Graph Construction

The TV-RW graph is constructed in two steps. First, we simulate the RW model to obtain the time-varying coordinates over time. The model simulates some sensors moving around a square space of 400 m^2 with random speed and directions. We obtain the time-varying data corresponding to the position of each sensor at any given time. The number of sensors is set to $N = 36$ and the motion speeds is set in the range of 0.05–0.5 m/s. We sample sensor positions with sampling period $\tau = 0.1$s for a total of 300 observations. In the second step, we construct a 3-nearest-neighbor graph at each time from the position

data. These edge weights are given by

$$w(i,j) = \exp\left(-\frac{\text{dist}(i,j)}{2\theta}\right), \tag{4.24}$$

where $\text{dist}(i,j)$ is the Euclidean distance between nodes i and j, and θ is a parameter. The edge weights in this graph generally vary smoothly as in (4.24); however, some edges will (dis)appear in the next time slot due to the nature of the k-neighborhood graph. Note that our method estimates time-varying graphs robustly for various k. This graph partially satisfies **(P1)** because the topology mostly remains unchanged in a short time horizon while the edge weights change smoothly over time.

The TH-ER graph is constructed by varying edges in an ER graph over time. We first construct an initial static ER graph $\mathbf{W}_1$ with $N = 36$ and an edge connection probability $p = 0.05$. The obtained graph satisfies **(P1)**. The tth graph $\mathbf{W}_t$ is obtained by resampling 5% of edges from the previous graph $\mathbf{W}_{t-1}$. In this way, we construct a set of graphs that varies over 300 time slots. The edge weights on this graph are set to be uniformly distributed in the interval $[0, 1]$. Once they are randomly selected, they remain fixed until the edges are resampled. In this graph, only a few edges switch at any given time while most of the edges remain unchanged, i.e., this graph also follows **(P1)**.

The SB-ER graph is a simulated example of the time-varying graph with **(P2)**. It is often observed in temporal brain connectivity dynamics [75, 76]. First, we generate six connectivity states, i.e., six static graphs. Each of the graph is an ER graph with $N = 36$ and an edge connection probability $p = 0.05$.[1] Second, the initial state is selected randomly from the six states.

Its state remains unchanged with a 98% probability and changes to another connectivity state with the 2% probability at each time. This graph follows the **(P2)** property. An SB-ER is constructed in this way and we have 300 observations in time.

Generating Data Samples

We create data samples from the constructed time-varying graphs in the same manner for all the graphs. Let $\mathbf{L}_t$ be the graph Laplacian corresponding to $\mathbf{W}_t$. A data sample $\mathbf{x}_t$ is generated from a Gaussian Markov random field model defined by

$$p(\mathbf{x}_t|\mathbf{L}^{(t)}) = \mathcal{N}(\mathbf{x}_t|0, \mathbf{L}_t^{\dagger} + \sigma^2\mathbf{I}), \tag{4.25}$$

[1] In our preliminary experiments, p hardly affects the graph learning performances.

where σ^2 is the covariance of the Gaussian noise. We set $\sigma = 0.5$ in this experiments. Pairs of variables generated from this distribution have closer values to each other when the corresponding nodes connect with a larger edge weight.

4.3.2 Performance Comparison

Experimental Conditions

We evaluate the performance in terms of relative error and F-measure, each averaged over all the time slots. Relative error is given by

$$\text{Relative error} = \frac{\|\widehat{\mathbf{W}} - \mathbf{W}^*\|_F}{\|\mathbf{W}^*\|_F}, \tag{4.26}$$

where $\widehat{\mathbf{W}}$ is the estimated weighted adjacency matrix, and $\mathbf{W}^*$ is the ground truth. It reflects the accuracy of edge weights on the estimated graph. The F-measure is given by

$$\text{F-measure} = \frac{2\text{tp}}{2\text{tp} + \text{fn} + \text{fp}}, \tag{4.27}$$

where the true positive (tp) is the number of edges that are included both in $\widehat{\mathbf{W}}$ and $\mathbf{W}^*$, the false positive (fn) is the number of edges that are not included in $\widehat{\mathbf{W}}$ but are included in $\mathbf{W}^*$, and the false positive (fp) is the number of edges that are included in $\widehat{\mathbf{W}}$ but are not included in $\mathbf{W}^*$. The F-measure, which is the harmonic average of the precision and recall, represents the accuracy of the estimated graph topology. The F-measure takes values between 0 and 1. The higher the F-measure is, the higher the performance of capturing the graph topology is.

In this experiment, our goal is to compare all the methods based on their best achievable performance. We perform Monte-Carlo simulations for each method to find the parameters minimizing the relative error. Note that, as shown in the next section, fixed parameters still work well for our graph learning with real dataset. For SGL-Smooth, TGL-Tikhonov, TGL-TH, and TGL-SB, α is selected by fine-tuning, and β is selected from the following set:

$$\{0\} \cup \{0.75^r z_{\max} \mid r = 1, 2, \ldots, 14\}, \tag{4.28}$$

where $z_{\max} = \max_{i \neq j}(\mathbf{Z})_{ij}$. The parameter η for TGL-TH and TGL-SB is selected from 0.1 to 2 in steps of 0.1. The parameter for SGL-GLasso is selected

by the method described in [39]. The tolerance value ϵ in Algorithm 3 is set to 1.0×10^{-3}. We evaluate the performance of each method with the different number of data samples $K = \{1, 5, 10, 25, 50, 100\}$ and measure the average of the relative error and the F-measure over all the time frames. Note that SGL-GLasso is not evaluated with $K = 1$ because the sample covariance used in SGL-GLasso cannot be calculated.

Results

Fig. 4.2 shows the performance comparisons according to K. Figs. 4.2(a) and (d) show the average F-measure and the relative error for TV-RW graphs; Figs. 4.2(c) and (e) show those for TH-ER graphs; Figs. 4.2(e) and (f) show those for SB-ER graphs. As shown in Fig. 4.2, the existing static graph learning methods, i.e., SGL-GLasso and SGL-Smooth, present worse performance than the time-varying methods. This is because SGL-Smooth learns a graph from each time slot independently. It does not consider the temporal relation of graphs. Note that SGL-GLasso sometimes presents a comparable performance with time-varying methods when K is large.

Fig. 4.2(a) and (d) present that TGL-TH and TGL-SB outperform TGL-Tikhonov despite the fact that the edges in a TV-RW graph vary smoothly in this case. As can be seen in this figure, TGL-Smooth yields undesirable edges. This could be due to the Tikhonov regularization, i.e., the squared ℓ_2 norm of the temporal variation. In TGL-Tikhonov, when the hyperparameter η is large, it yields time-varying graphs that have small temporal variations (in terms of the ℓ_2 variation) but the resulting graphs can be dense. In contrast, choosing small η may not properly reflect the assumption (P1). Refer to Section 4.6.1 for experiments that relate graph density to the choice of η.

TGL-TH significantly outperforms the other methods for TH-ER graphs as shown in Figs. 4.2(c) and (d). This indicates that the fused Lasso constraint on the difference between graphs in neighboring time slots works well. In contrast, TGL-Tikhonov and TGL-SB present almost the same performance as the static methods. This can be attributed to the fact that the assumptions on the temporal variation of TGL-Tikhonov and TGL-SB are not valid. Thus, since a TH-ER graph has neither smooth nor sudden changes, TGL-Tikhonov and TGL-SB may not be suited to learn TH-ER graph. It is also worth noting that, thanks to the regularization term, TGL-TH can successfully learn accurate time-varying graphs from a small number of measurements.

As can be seen in Fig. 4.2, the performance gain for TH-ER graphs is much better than that of TV-RW graphs. This is because the TH-ER graphs have

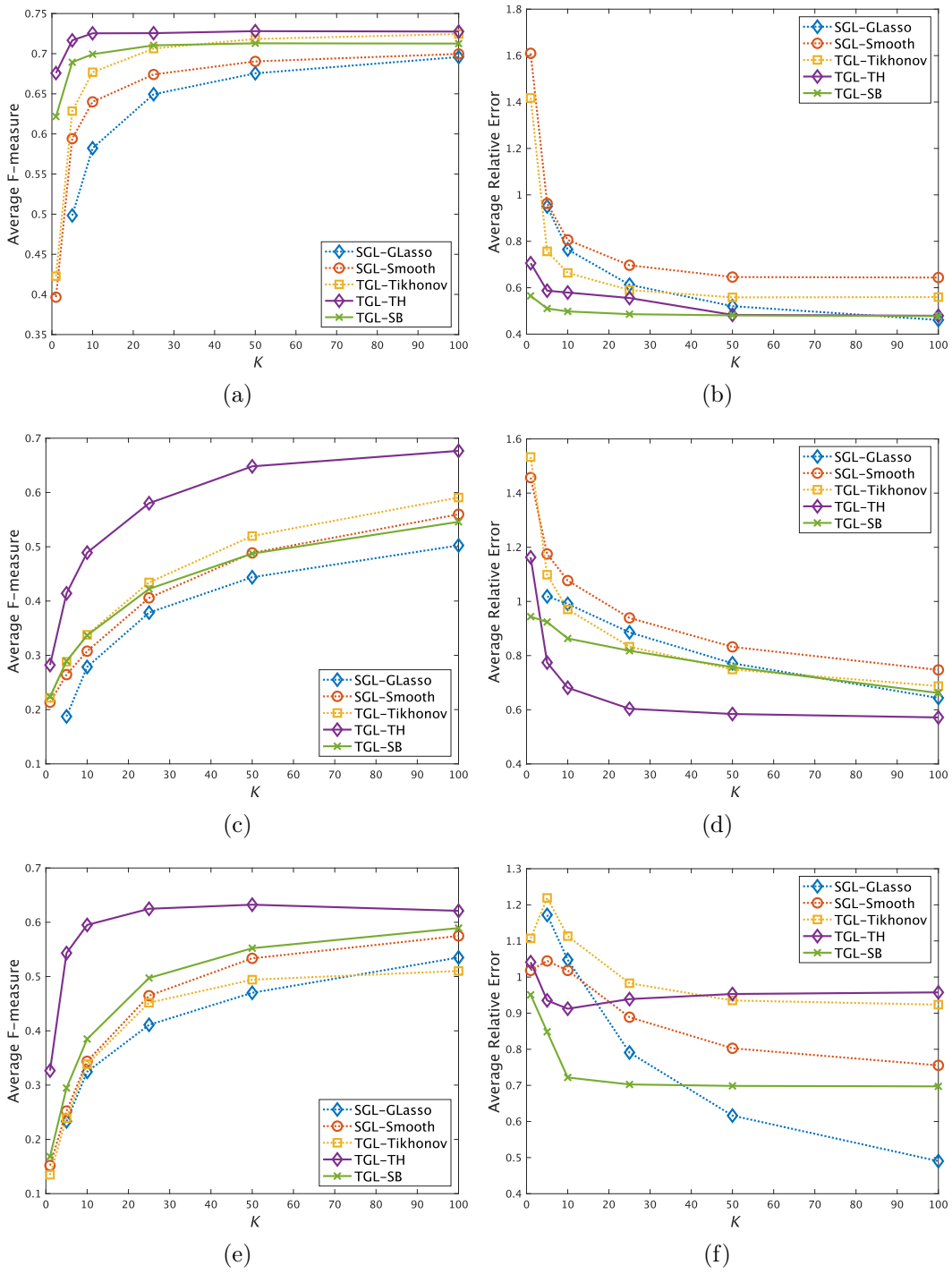
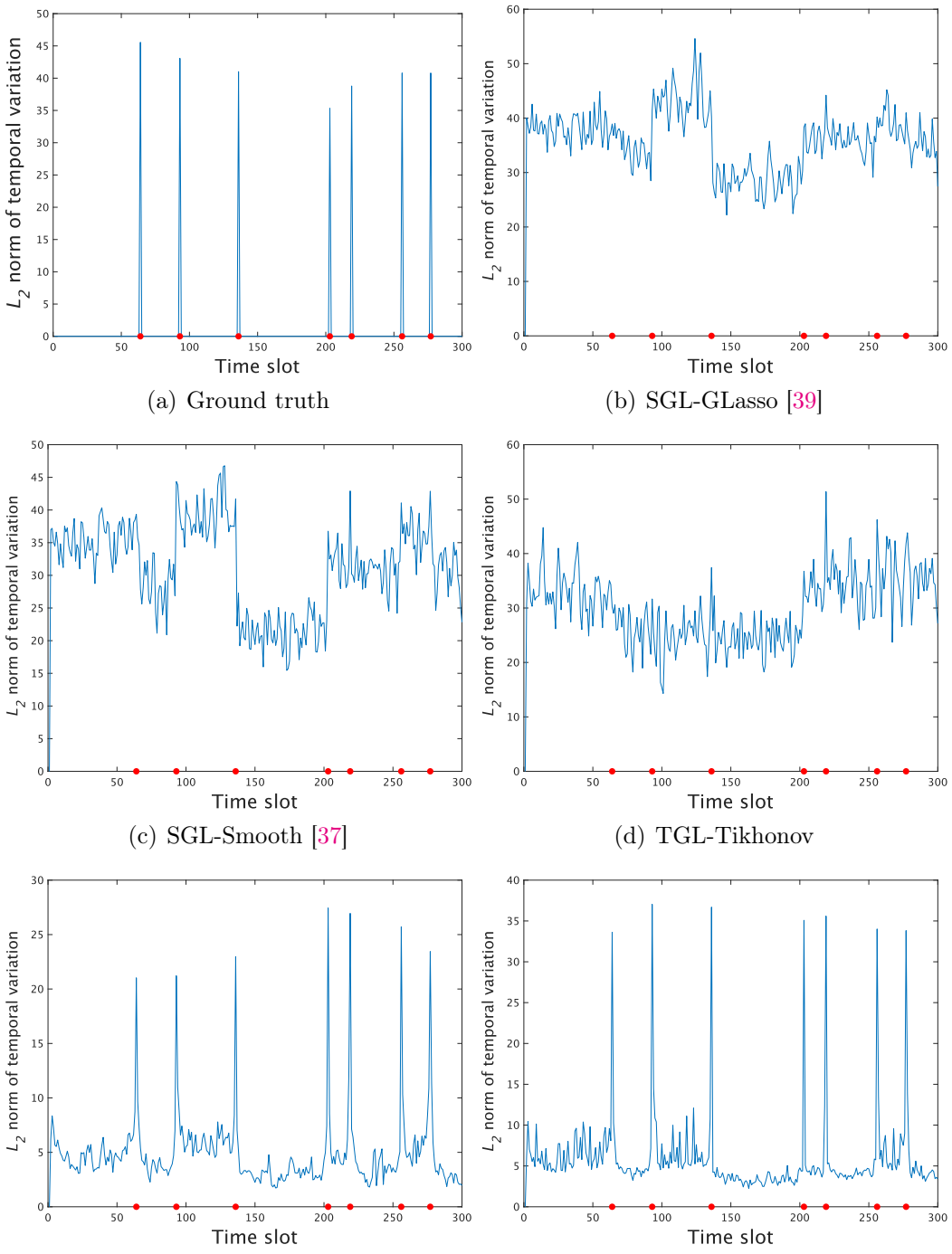


FIGURE 4.2: The performance of learning time-varying graph for different number of data samples. Top row demonstrates the F-measure for the datasets based on TV-RW graph, TH-ER graph, and SB-ER graph, respectively. Bottom row demonstrates the relative error for those datasets.

the **(P1)** properties while the TV-RW graphs lack a part of **(P1)**, as previously mentioned.

Focusing on Fig. 4.2(e) and (f), TGL-SB also outperforms the other methods



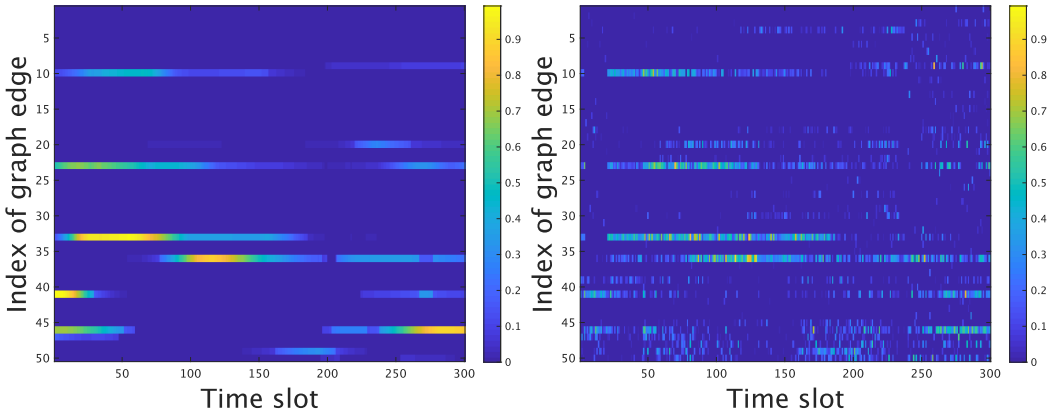
(a) Ground truth (b) SGL-GLasso [39]

(c) SGL-Smooth [37] (d) TGL-Tikhonov

(e) TGL-TH (proposed) (f) TGL-SB (proposed)

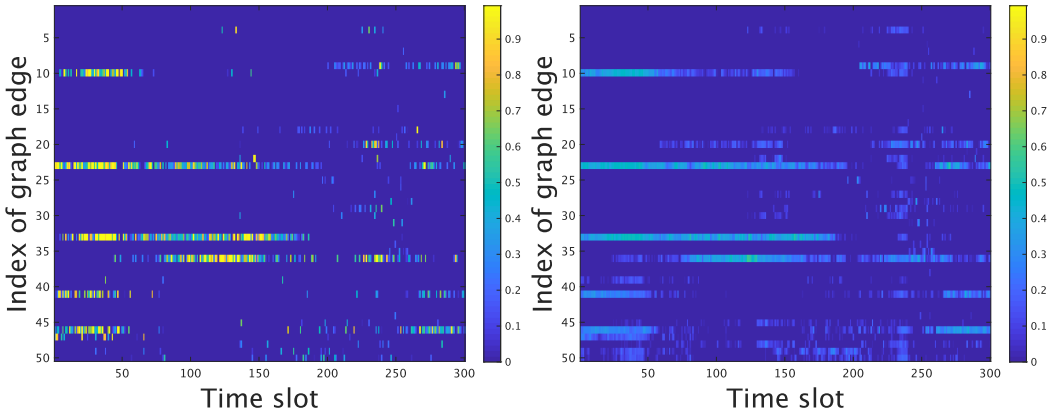
FIGURE 4.3: The visualization of the temporal variations in the time-varying graph learned from the dataset based on the graph in which large fluctuations occur at a few time slots. Red points in these figures represent time slots where the connectivity state changes.

for small K. This indicates that TGL-SB works well even in the situations where the number of available samples is small. In this result, we can also see that the regularization deviated from the assumption of temporal variation cannot



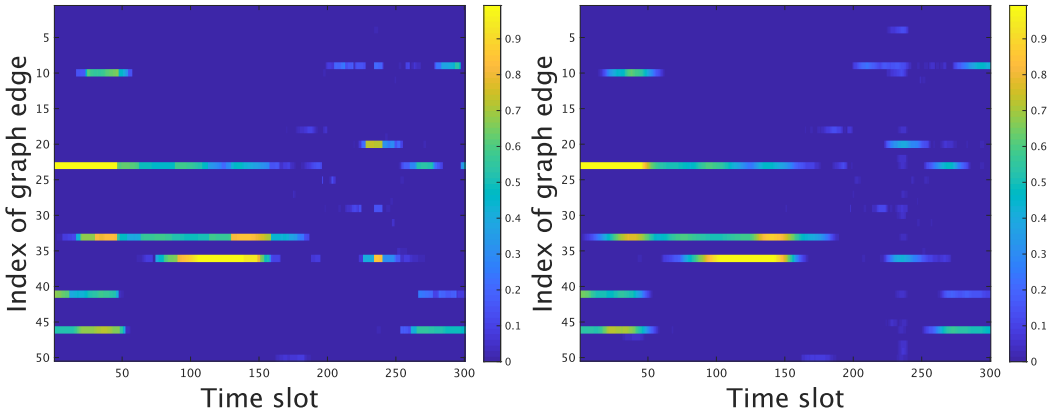
(a) Ground truth

(b) SGL-GLasso [39]



(c) SGL-Smooth [37]

(d) TGL-Tikhonov [54]

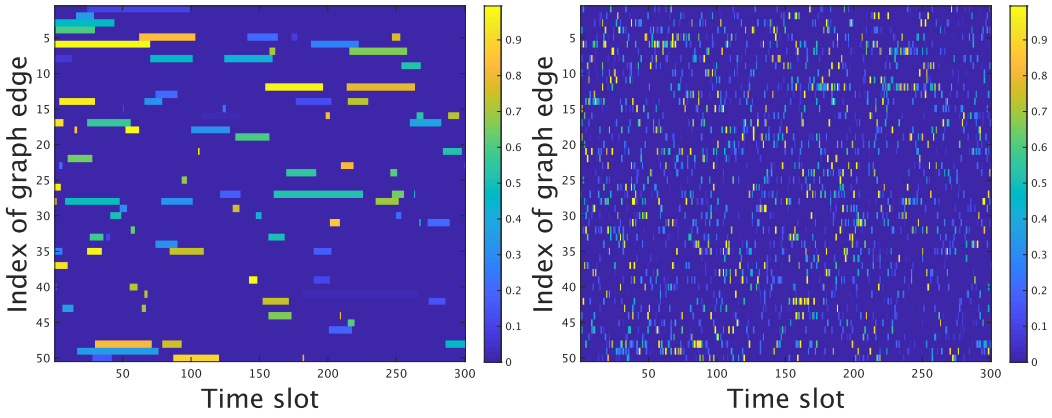


(e) TGL-TH (proposed)

(f) TGL-SB (proposed)

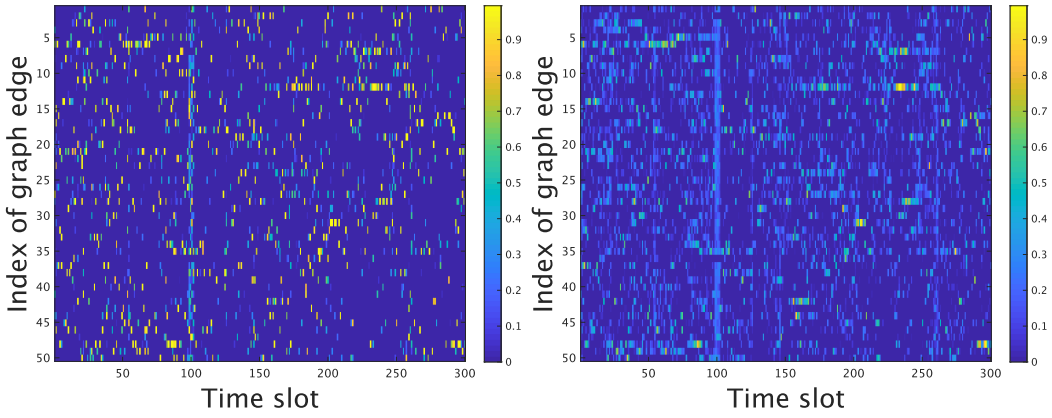
FIGURE 4.4: The visualization of the temporal variations in TV-RW graph. Top row demonstrates the variations for the dataset based on the TV-RW graph. Bottom row is the variations for TH-ER graph datasets. Colors in these figures represent the weights of the edges.

improve the performance as TGL-Tikhonov and TGL-TH shown in Fig. 4.2(e) and (f): The performance of the relative error for TGL-Tikhonov and TGL-TH is inferior to the static methods especially when K is large. Note that TGL-TH



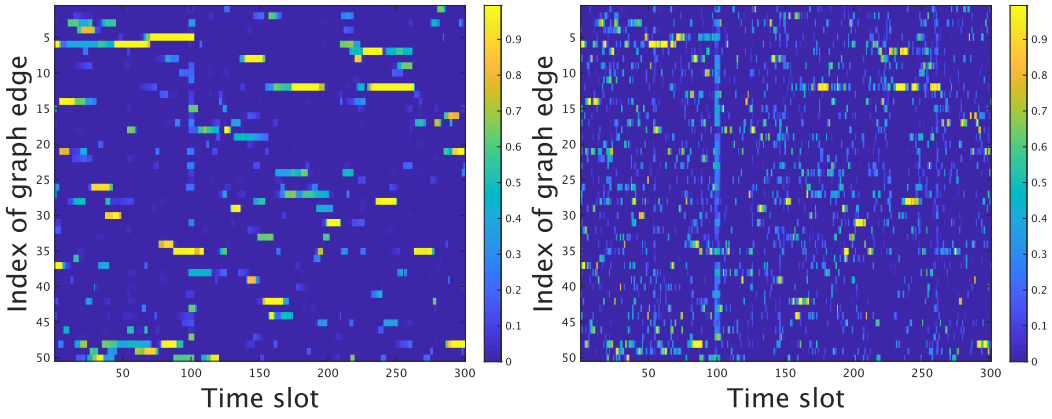
(a) Ground truth

(b) SGL-GLasso [39]



(c) SGL-Smooth [37]

(d) TGL-Tikhonov [54]



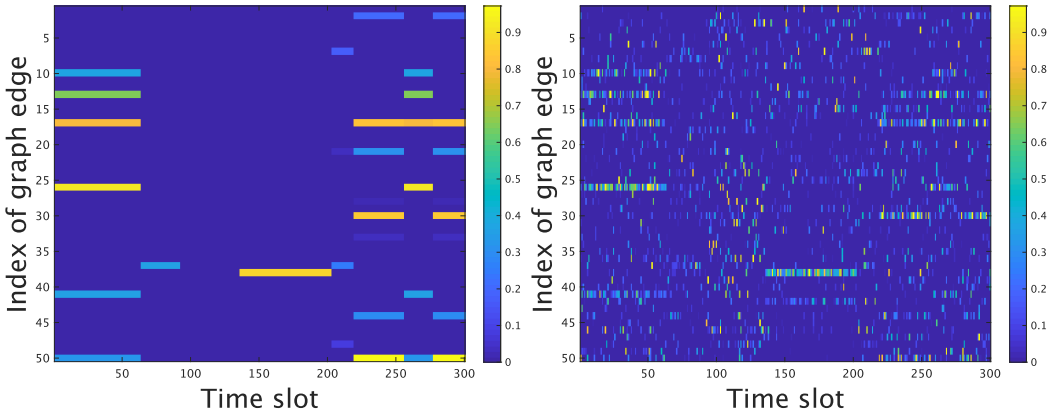
(e) TGL-TH (proposed)

(f) TGL-SB (proposed)

FIGURE 4.5: The visualization of the temporal variations in TV-ER graph.

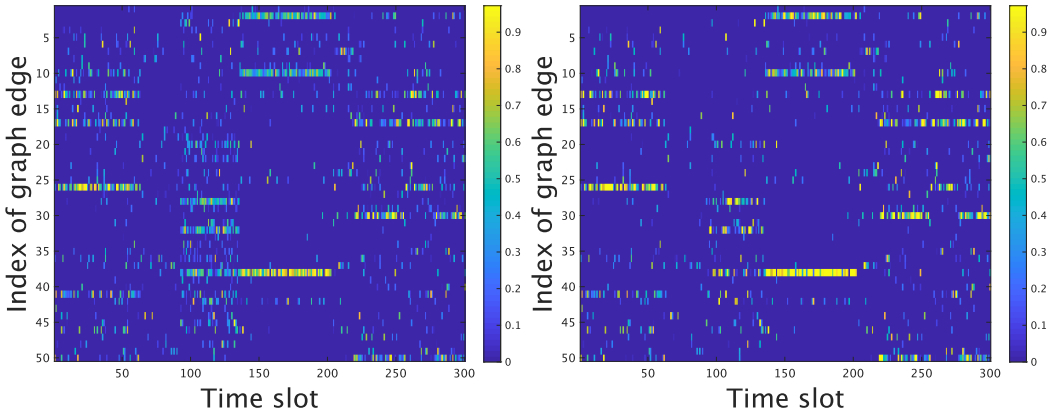
has high performance on F-measure. Since the structure of the SB-ER graph is unchanged in most time slots and partially satisfies **(P1)**, it can be considered that TGL-TH can successfully capture the structure.

Fig. 4.3 shows the L_2-norm of the temporal difference of learned adjacency matrices in each time slots, i.e., $\|\mathbf{W}_t - \mathbf{W}_{t-1}|\|_2$. Clearly, it can be observed



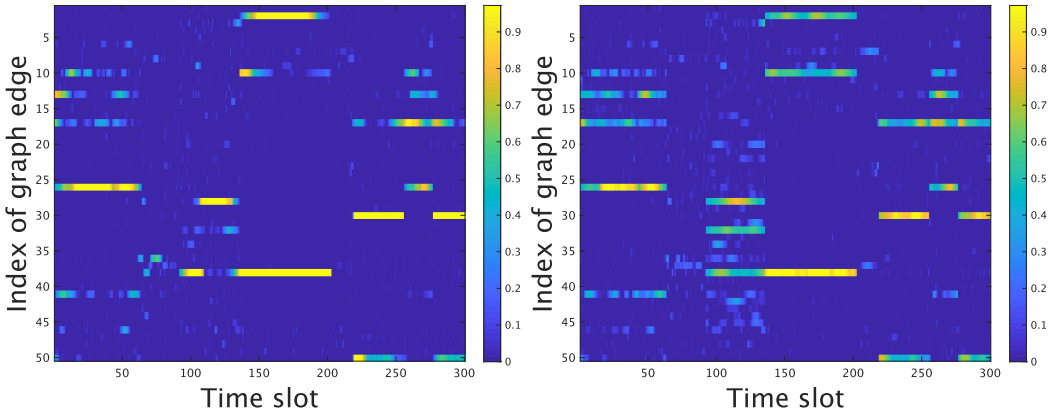
(a) Ground truth

(b) SGL-GLasso [39]



(c) SGL-Smooth [37]

(d) TGL-Tikhonov



(e) TGL-TH (proposed)

(f) TGL-SB (proposed)

FIGURE 4.6: The visualization of the temporal variations in TV-SB graph.

that TGL-TH and TGL-SB can detect sudden changes of the graph in time slots, while SGL-GLasso and SGL-Smooth have several unclear jumps. On the other hand, TGL-Tikhonov can suppress the temporal variation of graphs than SGL-Smooth but obscures the change points in the time-varying graph. As a result, TGL-SB is a graph learning method suitable for change detection, and

the group Lasso of temporal variation is effective in graph learning.

Figs. 4.4, 4.5 and 4.6 visualizes the temporal variation of a part of edge weights on the learned graphs from the datasets of TV-RW, TH-ER, and SB-ER graphs with $K = 10$. The vertical and horizontal axes of these figures represent the edge and time slot indices of the learned graph, and the color represents the intensity of the edge weights. To make an easy view, they visualize the first 50 edge indices. As can be seen in these figures, SGL-GLasso and SGL-Smooth cannot capture the original graph structure, and they are sensitive to noise and output the graph ignoring its temporal relations. TGL-Tikhonov is slightly superior to the static graph learning methods. However, it yields many undesirable edges and the edge weights are relatively small. In contrast, TGL-TH estimates the original structure better than the other methods and outputs the graph considering its temporal relationships.

Focusing on Fig. 4.6, it can be observed that SGL-GLasso and SGL-Smooth cannot capture the change points in the graph. In contrast, TGL-SB can detects large edge changes in the original time-varying graph.

4.3.3 Effect of Temporal Resolution

To verify the robustness of the proposed method against the temporal graph transition, i.e., temporal resolution, we evaluate the performance for the dataset of the TV-RW graph with different sampling periods $\tau = 0.1, 0.5, 1.0, 1.5, 2.0$. Table 4.2 shows the performance comparisons with the dataset of TV-RW graph with $K = 10$ according to τ. TGL-TH outperforms other methods when the sampling period τ is small. On the other hand, TGL-TH and TGL-Tikhonov have comparable performance for the average relative error with $\tau = 1.0$, and TGL-TH gets worse performance than TGL-Tikhonov when τ gets large. This is because many edge weights of graphs may vary over time when the sampling period is long, i.e., it does not satisfy **(P1)**. For the same reason, the performance of TGL-SB also is not improved as the sampling period is longer.

4.3.4 Computation Time

We compare the computation time of the proposed methods with some related works. All methods are implemented in MATLAB R2018b and run on a 2.3-GHz Intel Xeon W processor with 128-GB RAM. The experiments are tested on the TH-ER graph dataset for different number of nodes $N = \{10, 50, 100, 250, 500\}$. The tolerance value ϵ in each methods is set to 1.0×10^{-3}.

TABLE 4.2: The performance of learning time-varying graph for different sampling periods.

Average F-measure					
Sampling period	0.1	0.5	1.0	1.5	2.0
SGL-GLasso	0.613	0.574	0.574	0.580	0.575
SGL-Smooth	0.639	0.562	0.562	0.563	0.534
TGL-Tikhonov	0.681	0.606	0.613	0.611	0.596
TGL-TH	**0.718**	**0.636**	**0.656**	**0.664**	**0.627**
TGL-SB	0.712	0.614	0.583	0.582	0.519
Average Relative Error					
Sampling period	0.1	0.5	1.0	1.5	2.0
SGL-GLasso	0.602	0.690	0.699	0.697	0.696
SGL-Smooth	0.612	0.683	0.692	0.685	0.731
TGL-Tikhonov	0.529	0.606	**0.607**	**0.605**	**0.633**
TGL-TH	**0.431**	**0.574**	0.609	0.656	0.701
TGL-SB	0.439	0.597	0.649	0.639	0.738

Fig. 4.7 shows the computation time of each method, demonstrating that the proposed methods (TGL-TH and TGL-SB) converge faster than SGL-GLasso. Because SGL-GLasso needs to solve nonnegative quadratic programs as the subproblem in each iteration, it requires very significant computation time. The computational complexity of one iteration in SGL-GLasso is $\mathcal{O}(T(N^2 + T_p(N)))$ complexity where $\mathcal{O}(T_p(N))$ is the worst-case complexity of solving the subproblem, and $T_p(N) = \Omega(N^3)$. On the other hand, the complexity iteration in TGL-TH and TGL-SB is $\mathcal{O}(TN^2)$ complexity.

TGL-TH and TGL-SB required longer computation time than SGL-Smooth and TGL-Tikhonov, where they estimate a graph using almost the same algorithm; that is, they have the same computational complexity $\mathcal{O}(TN^2)$. However, TGL-TH and TGL-SB introduce dual variables to solve the optimization problem with temporal variation regularization, i.e., they have more variables to update than SGL-Smooth and TGL-Tikhonov. Although TGL-Tikhonov also has temporal variation regularization, it can be computed without introducing a dual variable because its temporal variation regularization can be differentiable.

4.4 Denoising of Dynamic Point Clouds

Our proposed method is applied to dynamic point cloud data denoising. Dynamic point cloud data contains 3D coordinates of dynamically evolving points.

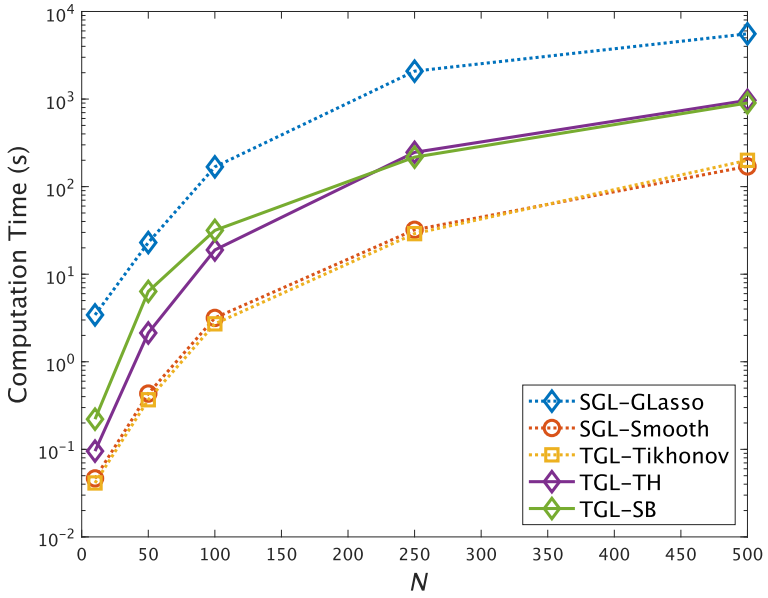


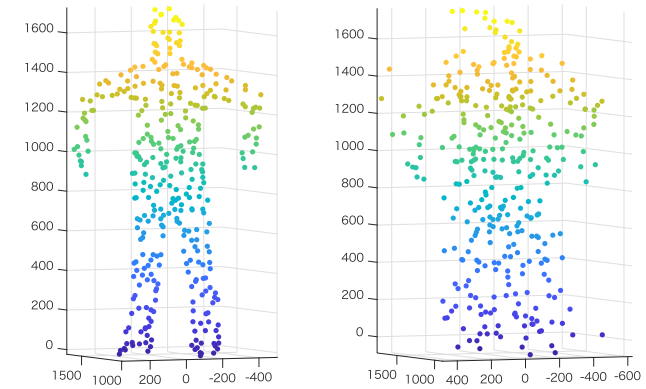
FIGURE 4.7: The comparison of computation time for the different number of N.

TABLE 4.3: Denoising Results: SNR (dB)

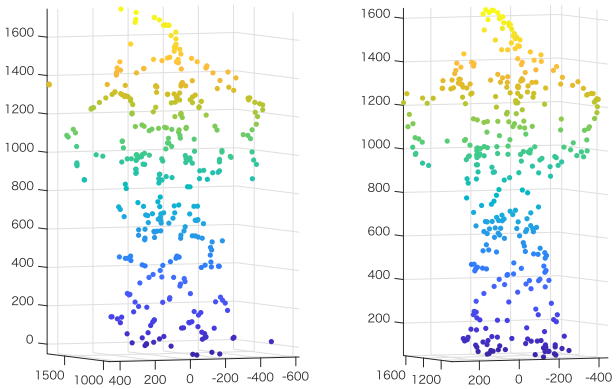
	dog	*handstand*	*skirt*	*wheel*
noisy	12.12	13.97	13.73	15.45
k-NN	12.35	14.29	14.01	15.76
SGL-Smooth	13.13	15.37	15.03	16.74
TGL-Tikhonov	13.34	15.69	15.34	17.06
TGL-TH	**20.05**	**21.32**	**21.58**	**22.97**

When point cloud data are acquired, the measurement error leads to the displacements of the geometric coordinates of the point clouds. Here, we consider graph-based denoising approaches. The performance of noise removal depends on the underlying graph. In this experiment, denoising is performed by using graph heat kernel filtering [77]. The time-varying graph used in the denoising is estimated from noisy point cloud data.

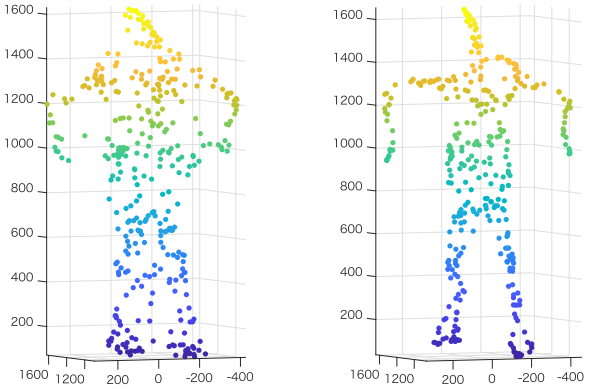
We use the dynamic point cloud dataset in [78], which contains five dynamic point clouds: *dance, dog, handstand, skirt*, and *wheel*. As this dataset is clean data, with position ranges from -260 to 1932, we added Gaussian noise with $\sigma = 90$, which is a significantly higher noise level. The time-varying graph is learned from the dataset downsampled to 357 points and evolving over 240 time slots. In this experiment, we use fixed parameters for each method, which is determined by a grid search with *dance* data, and evaluate the denoising performance with the other four data.



(a) Ground truth (b) Noisy

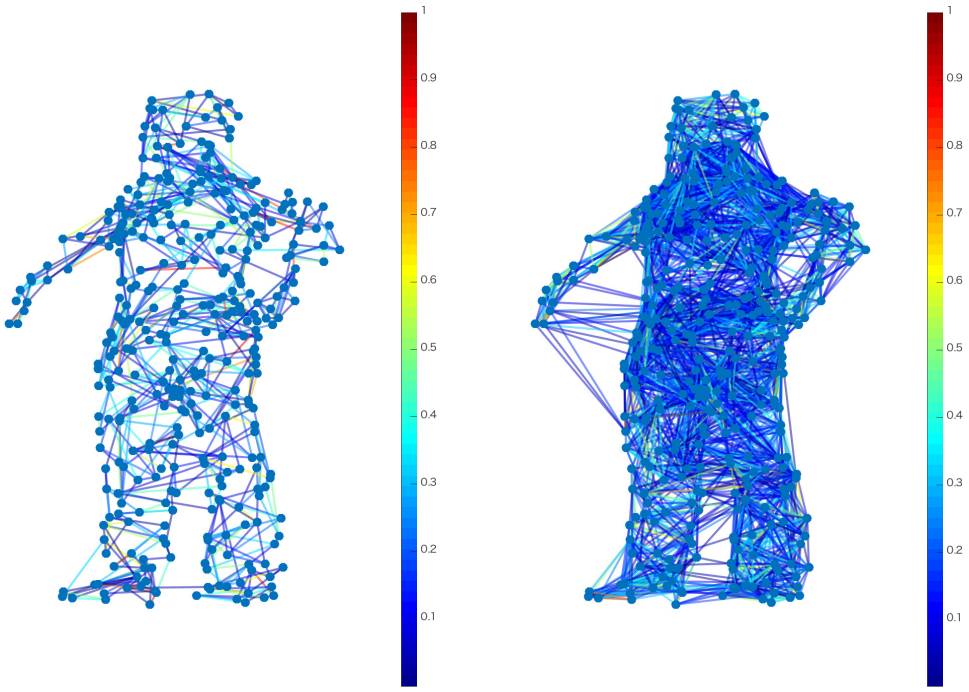


(c) k-nearest neighbor (d) SGL-Smooth [37]



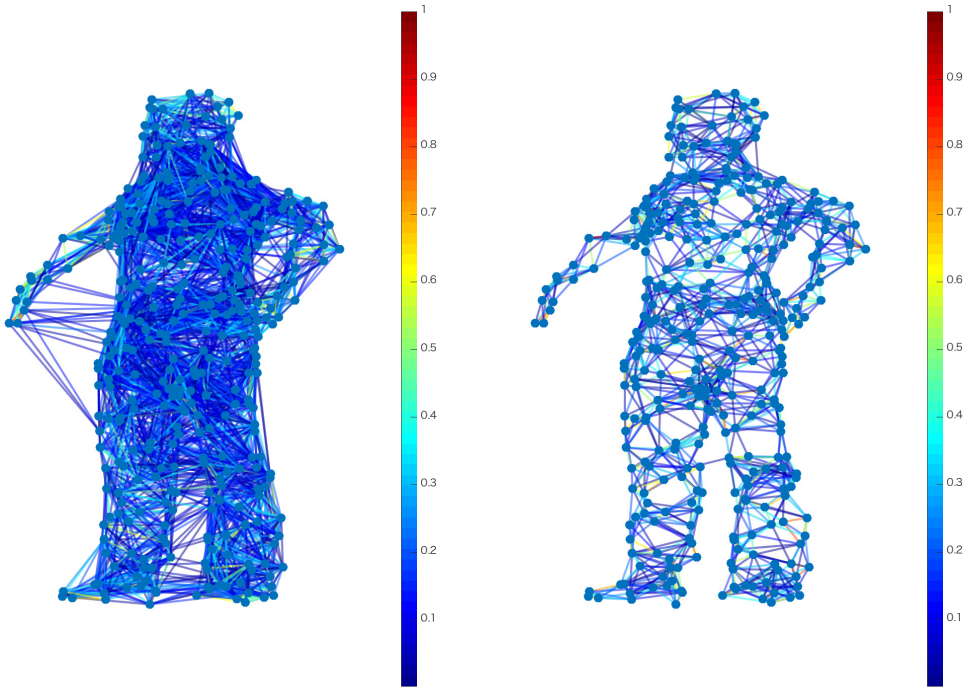
(e) TGL-Tikhonov [54] (f) TGL-TH (proposed)

FIGURE 4.8: The visualization of denoising result of *wheel* at a certain time.



(a) k-nearest neighbor

(b) SGL-Smooth [37]



(c) TGL-Tikhonov [54]

(d) TGL-TH (proposed)

FIGURE 4.9: The visualization of a graph at a certain time in the time-varying graph learned from the noisy point cloud data.

Table 4.3 summarizes the dynamic point cloud denoising qualities. Time-varying graph learning methods, TGL-Tikhonov and TGL-TH, show better

results than SGL-Smooth and k-nearest neighbor. This indicates that the k-nearest neighbor cannot construct the meaningful graph from noisy data. Additionally, TGL-TH outperformed TGL-Tikhonov up to 6 dB.

Fig. 4.8 shows the visualization of denoising results of the *wheel* at a certain time. Similar to the numerical performance, the k-nearest neighbor cannot capture the structure of the human body. SGL-Smooth and TGL-Tikhonov yield slightly better outputs than that by the k-nearest neighbor; however, the arms and legs are still problematic. On the other hand, TGL-TH can successfully capture the structure of the human body than the other methods.

Fig. 4.9 visualizes a graph at a certain time in the time-varying graph estimated from the noisy *wheel* data in the dynamic point cloud dataset. In this figure, the nodes in the graphs are plotted in the correct position for visualization. As shown in Fig. 4.9, k-nearest neighbor yields a sparse graph but nodes are connected without a temporal relationship. SGL-Smooth and TGL-Tikhonov yielded very dense edges along with connecting different parts in the body. In contrast, TGL-TH yields the graph whose nodes are connected accurately and preserves the human body structure.

4.5 Application to Temperature Data

We apply the proposed method to estimate a time-varying graph from temperature data in the island of Hokkaido, the northernmost region in Japan. The goal of this experiment is to learn a time-varying graph to explore the relationship between geographical regions over time. In this experiment, we use the average daily temperature data[2] collected from 172 locations in Hokkaido in 2015, i.e., the overall the dataset has 365 time slots with $N = 172$ and $K = 24$. From this data, we estimate a time-varying graph by using TGL-TH.

Fig. 4.10 shows the graph learned from temperature data[3]. For visualization, we remove edges having very small weight ($< 1.0 \times 10^{-4}$). From the figure, the inferred characteristics of the learned graph are as follows:

- Nodes close to each other are basically connected. This is reasonable because the temperature is similar in the same geographical area. Note that, if the recording points are separated by mountain ranges, nodes may not be connected even if the distances are short.

[2]The Japan Meteorological Agency provided the daily temperature data from their website at https://www.jma.go.jp/jma/index.html

[3]The Geospatial Information Authority of Japan provided the map in Hokkaido with altitude from their website at http://www.gsi.go.jp/

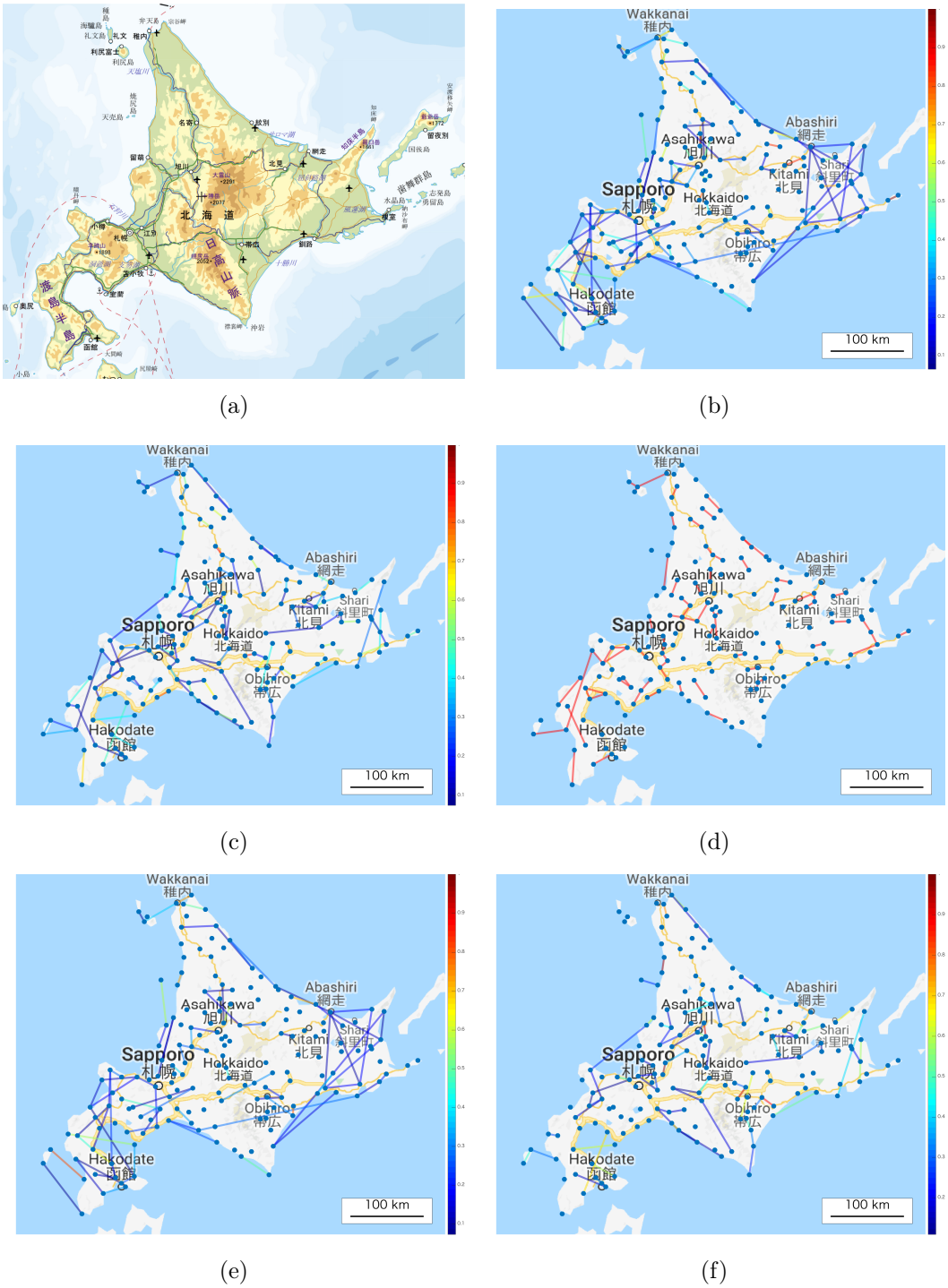
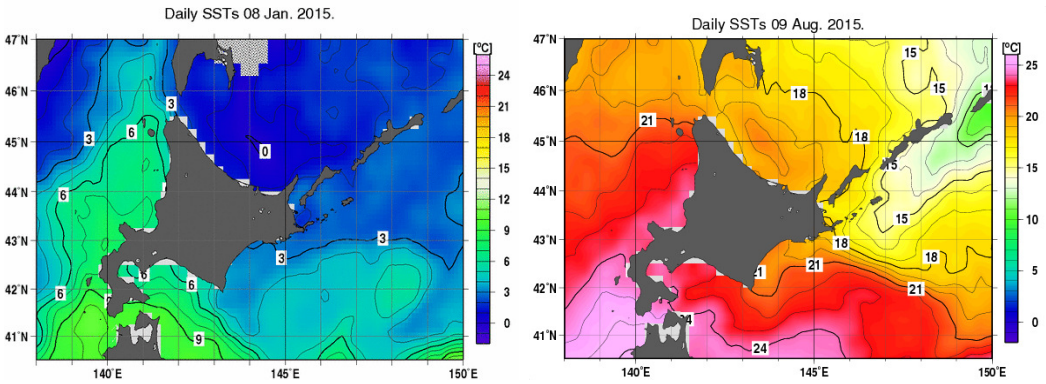


FIGURE 4.10: The visualization of the graph learned from the temperature data. (a) Map of Hokkaido with altitude. (b) The graph learned on January 8, 2015 (winter graph). (c) The graph learned on August 9, 2015 (summer graph). (d) Edges common in winter and summer. (e) The winter graph from which the common edges have been removed (winter-specific graph). (f) The summer graph from which the common edges have been removed (summer-specific graph).



(a) (b)

FIGURE 4.11: Daily sea surface temperature (SST) on (a) 8 January 2015. (b) 9 August 2015.

- Nodes along the coast are often connected to each other, and the inland nodes are also connected to each other. Especially, coastal nodes may connect to each other despite being far away.

- Locally connected structures in the learned graph mostly remain unchanged over time as shown in Fig. 4.10(d).

Figs. 4.10(e) and 4.10(f) show the season-specific graphs. We have found that a coastal node often has a connection to another coastal node even if they are not close to each other. This can be justified by the warming or cooling effect of sea currents that occur seasonally in this area and that affect coastal areas in similar ways. Fig. 4.11 represents daily sea surface temperatures (SST) on January 8 and August 9, 2015 [4]. As can be seen in Figs. 4.10(e), 4.10(f), and 4.11, nodes in similar SST areas are connected to each other in the learned graph. It is important to emphasize that there was no prior information about the SST areas used in the graph learning process.

4.6 Learning Graphs from COVID-19 data

In this section, we apply the proposed method to another set of real-world data: Daily confirmed cases of COVID-19 spread in Italy and California. We use daily confirmed cases in Italy [79] from February 26 to July 28, 2020 in 103 provinces[5] and those in California[6] [80] from March 1 to July 12, 2020, in 58 counties. We

[4]The daily SST was provided by Japan Meteorological Agency, from their website at https://www.jma.go.jp/jma/index.html

[5]We excluded the island of Sardinia because it is far from the Italian Peninsula.

[6]The COVID-19 cases in California were provided from COVID-19 Data Repository by the Center for Systems Science and Engineering (CSSE) at Johns Hopkins University.

split the data into segments, one per week, and time-varying graphs are learned for each of the segments.

4.6.1 Graph Sparseness and Parameter Tuning

First, we study the relation between the sparseness of the graph and the hyperparameter η that controls the temporal variation of the learned time-varying graphs. We measure the sparseness of time-varying graphs as the temporal average of the average degree (abbreviated as TAAD):

$$\text{TAAD} = \frac{2\sum_{t=1}^{T}|\mathcal{E}_t|}{NT} \tag{4.29}$$

where $|\mathcal{E}_t|$ represents the number of edges in the graph at t.

Fig. 4.12 shows TAAD with different η when $\alpha = 1$ and $\beta = 0$ where we only have the log barrier term for avoiding trivial solutions. Note that TAAD of SGL-Smooth is constant because it is a static approach and does not depend on η.

As can be seen in Fig. 4.12, TGL-Tikhonov yields denser graphs as η becomes larger. In contrast, TGL-TH yields sparser graphs than those by the other methods. The sparsest graphs by TGL-TH have the average degree less than four: This cannot be obtained by SGL-Smooth and TGL-Tikhonov.

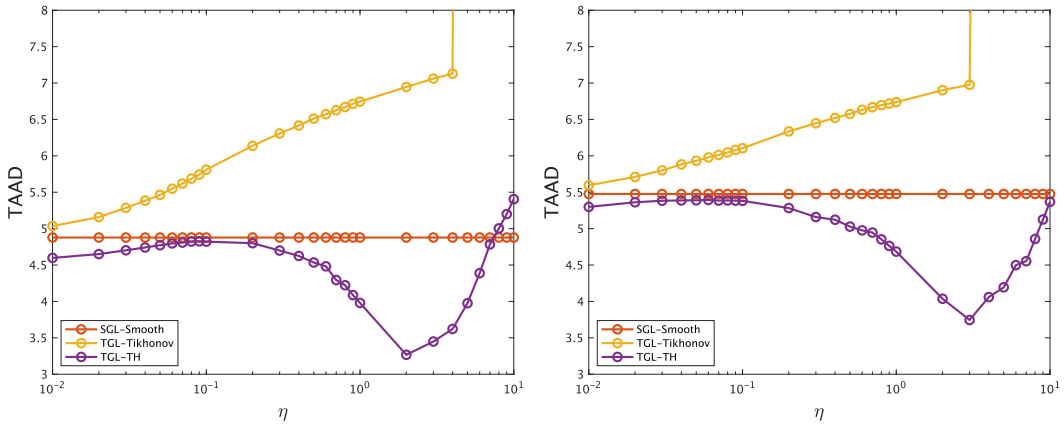
In the following experiments, we set the hyperparameters of the proposed and alternative methods such that TAAD becomes a predetermined value. For all the methods, TAAD $= 6$ is selected. We also present the results of the sparser version of TGL-TH with TAAD $= 4$. This could not be obtained by the other methods in this experiment. Table 4.4 summarizes the hyperparameters for each method.

TABLE 4.4: Hyperparameter tuning for each method

	α	β	η
SGL-Smooth	1	Adjusted	-
TGL-Tikhonov	1	0	Adjusted
TGL-TH	1	Adjusted	1

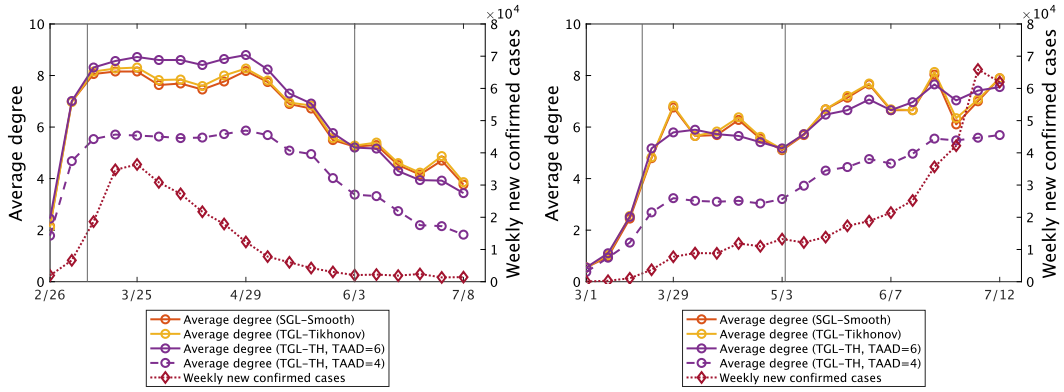
4.6.2 Case Study: Italy

In Italy, the first lockdown was started on February 21 in the northern provinces of Lodi and Padua. On March 9, the quarantine zones were expanded to the entire country, and nonessential movements were restricted. Then, the nationwide



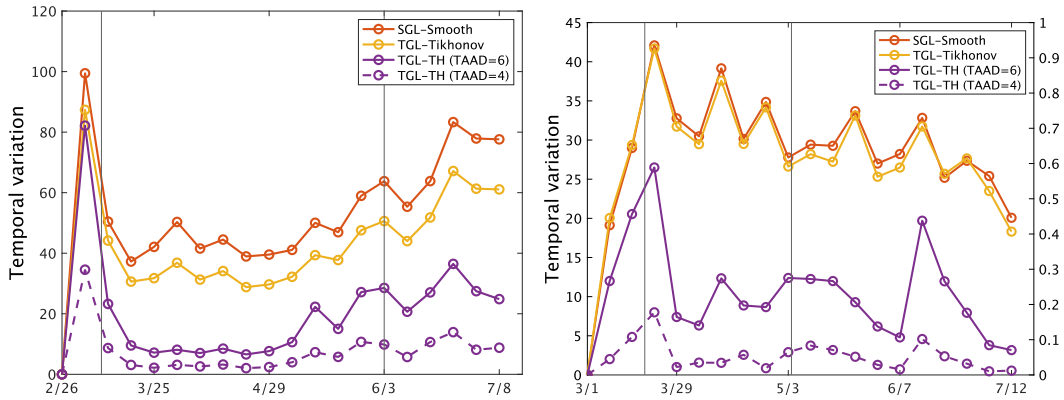
(a) Italy (b) California

FIGURE 4.12: The average degree in the learned graph with different parameter η.



(a) Italy (b) California

FIGURE 4.13: Weekly new confirmed cases of COVID-19 and the average degrees of the learned graphs.



(a) Italy (b) California

FIGURE 4.14: The temporal variation of time-varying graphs learned by each method.

travel restrictions were relaxed on June 3.

Fig. 4.13(a) shows the weekly new confirmed cases of COVID-19 in Italy and the average degree of the learned graphs. The two vertical lines represent the period between the start of the lockdown and the relaxation of it. As can be seen in Fig. 4.13(a), all of the methods show the same tendency during the lockdown: The average degree is almost unchanged first and then slightly decreased. This implies the effect of the travel restriction, i.e., the relationship among nodes (provinces) is decreased.

4.6.3 Case Study: California

In California, a state of emergency was in place on March 4, 2020. A mandatory statewide stay-at-home order was issued on March 19, and people were directed to stay home except to go to an essential job or to shop for essential needs. The stay-at-home order was relaxed on May 4: Some business were allowed to reopen.

Fig. 4.13(b) shows the weekly new confirmed cases of COVID-19 in California and the average degrees of the learned graphs. The two vertical lines represent the beginning of the stay-at-home order and its relaxation, respectively. As can be seen in this figure, during the stay-at-home order, the average degree of the learned graph by the TGL-TH presents a similar behavior to that in Italy. While the average degrees of the graph learned by SGL-Smooth and TGL-Tikhonov seem to be decreased during the stay-at-home order, they are oscillated. After the relaxation of the order, all the graphs grow the number of edges which imply the increased relationship between counties.

4.6.4 Discussion

We then discuss the characteristics of the learned graphs. As shown in Figs. 4.13(a) and 4.13(b), every method yields graphs that exhibit global behavior similar to each other, but the graphs learned by TGL-TH have smoother changes in the average degree than the other methods. Fig. 4.14 shows the temporal variation $\|\mathbf{W}_t - \mathbf{W}_{t-1}\|_1$ as a function of t. This figure also demonstrates that the graphs learned by TGL-TH have the smallest temporal variation, as expected from its objective function even under the same TAAD condition.

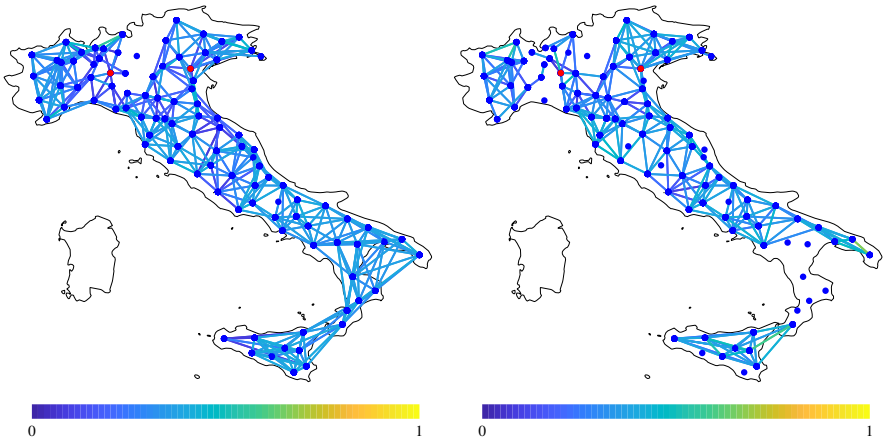
Finally, we compare the visualization of the graphs by TGL-Tikhonov and TGL-TH. Figs. 4.15(a)–4.15(f) show the learned graphs for two periods: 1) graphs between March 4 to March 10 (after the first lockdown and starting the expansion of the quarantine zone) and 2) graphs between May 27 to June 2 (just

before relaxing the travel restrictions). Two red nodes in each graph represent the provinces of Lodi and Padua, two of the first locked-down large cities.

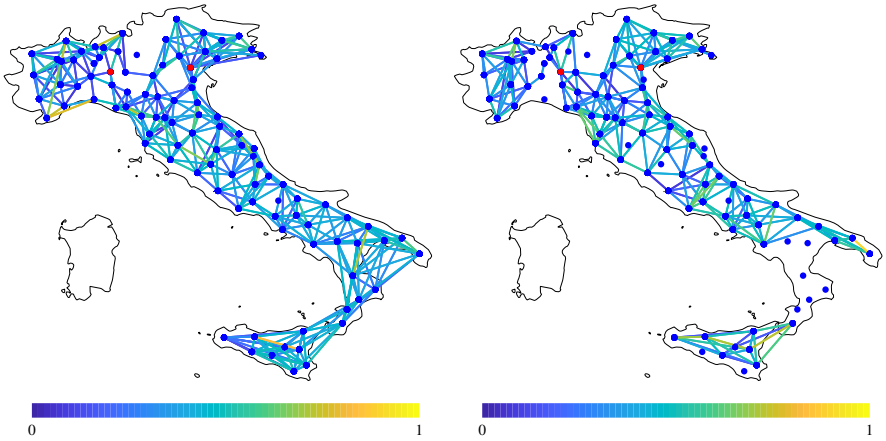
As shown in Figs. 4.15(a), 4.15(c), and 4.15(e), all graphs have sparser edges in northern Italy. Especially, TGL-TH (TAAD=4) is very sparse. It is observed that the graph by TGL-TH (TAAD=4) has few edges around Lodi and Padua. In Figs. 4.15(b), 4.15(d), and 4.15(f), all the graphs become sparser than those before lockdown. TGL-TH (TAAD=4) has sparser edges in the southern regions than the other two, that would reflect the effect of the nation-wide lockdown: It would weaken the relationship among neighboring regions.

4.7 Summary

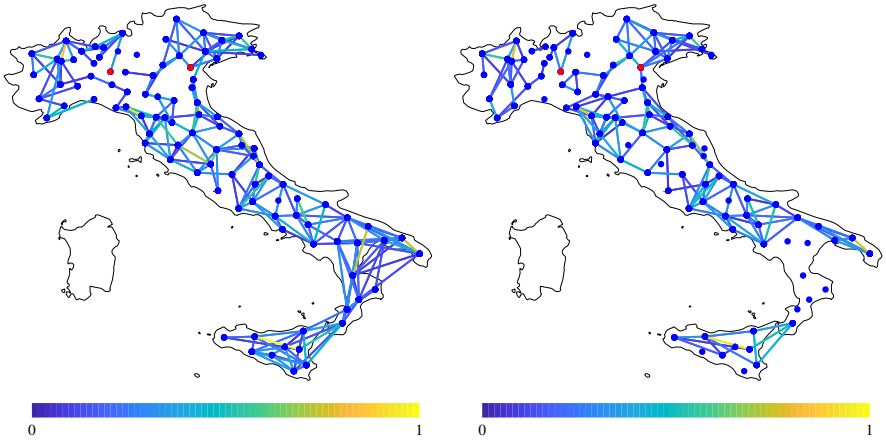
In this paper, we have presented a framework for learning time-varying graphs suitable for analysis of spatiotemporal data and change detection. The framework formulates a time-varying graph learning problem as convex optimization problems with the constraints on the temporal variation in the graph. Specifically, the framework introduces fused Lasso regularization and group Lasso regularization of the temporal variation to learn the graph having temporal homogeneity and one reflecting a sudden change in networks respectively. We also propose algorithms solving the optimization problems based on the primal-dual splitting algorithm. In the applications with synthetic and real data, we demonstrated that our proposed model can successfully learn the time-varying graphs, especially under the condition that the number of observations is small. Our future work includes implementing an automatic parameter tuning method and extending the algorithm for online graph learning.



(a) TGL-Tikhonov (b) TGL-Tikhonov



(c) TGL-TH (TAAD=6) (d) TGL-TH (TAAD=6)



(e) TGL-TH (TAAD=4) (f) TGL-TH (TAAD=4)

FIGURE 4.15: The visualization of the graph learned at a certain time slot from the COVID-19 confirmed cases in Italy. (a), (c), (e) Graphs learned for the period March 4th to March 10th (after the first lockdown and starting the expansion of the quarantine zone). (b), (d), (f) Graphs learned for the period between May 27 to June 2 (just before relaxing the travel restrictions)

Chapter 5

Temporal Multiresolution Graph Learning

To handle these dynamic behaviors, *time-varying graph learning* (TVGL) methods have been proposed [49,50,54,81]. Technically, a time-varying graph consists of multiple graphs where one graph corresponds to the relationship among vertices in a certain time slot. Typically, TVGL divides multivariate time-series data into consecutive (overlapped or nonoverlapped) data segments, and learns multiple graphs from these segments.

Time-varying graph learning (TVGL) methods often suffer from a trade-off in the temporal resolution. For example, a large time window allows the capture of a global structure but results in the loss of the local temporal behavior. In contrast, selecting a small time window enables the capture of fast-changing behaviors but may also result in noise sensitivity. To tackle this problem, most TVGL methods (including TVGL method introduced in Chapter 4) [49, 50, 54] impose constraints for temporal variations of graphs between neighboring time slots. However, they have two main limitations. First, they are not suitable when the data are not fitted to the prior assumption of temporal variations. Second, even if the temporal variation information is known, it is often difficult to determine the hyperparameter(s) such as those in the constraints on the temporal variation and the temporal window size.

In this chapter, we propose a TVGL method without using a specific prior for network evolution. Instead, we assume that the time-varying graphs have a *temporal multiresolution (TMR) structure*: They can be represented by the combination of graphs at different temporal resolutions from local (i.e., short time period) to global (i.e., static) ones. This is desirable because multivariate time-series data tend to have a multiresolution property. An example of such TMR data is temperature data observed in multiple sensor locations. The measurements in each location exhibit the interdependence relationships which

change hourly, daily, monthly, and even yearly, where the relationships correspond to structures localized at different temporal resolutions. Our proposed method automatically reveals which edge sets are localized at which temporal variation. In our problem setting, temporal resolution levels correspond to time window sizes in the time-varying graphs, but are not necessarily set as a hyperparameter.

The proposed TVGL is formulated as a convex optimization problem derived from the generation model. We also present an iterative algorithm for solving the optimization problem efficiently, which guarantees the convergence of the solution.

5.1 Graph Laplacian Operator

Let $\mathbf{w} \in \mathbb{R}_+^{N(N-1)/2}$ be a vector composed of edge weights. It corresponds to the vectorized version of the lower triangular part in $\mathbf{W}$ (excluding diagonal elements). The graph Laplacian operator $\mathcal{L} : \mathbf{w} \in \mathbb{R}_+^{N(N-1)/2} \to \mathcal{L}\mathbf{w} \in \mathbb{R}^{N\times N}$ is defined as [53]

$$[\mathcal{L}\mathbf{w}]_{i,j} = \begin{cases} -w_{i+d_j} & i > j \\ [\mathcal{L}\mathbf{w}]_{j,i} & i < j \\ \sum_{i\neq j}[\mathcal{L}\mathbf{w}]_{i,j} & i = j \end{cases} \tag{5.1}$$

where $d_j = -j + \frac{j-1}{2}(2N - j)$. Simply speaking, $\mathcal{L}$ transforms $\mathbf{w}$ into $\mathbf{L}$. Its adjoint operator $\mathcal{L}^* : \mathbf{Y} \in \mathbb{R}^{N\times N} \to \mathcal{L}^*\mathbf{Y} \in \mathbb{R}_+^{N(N-1)/2}$ is given by

$$\begin{aligned} [\mathcal{L}^*\mathbf{Y}]_k &= y_{ii} - y_{ij} - y_{ji} + y_{jj}, \\ k &= i - j + \frac{j-1}{2}(2N - j), \quad (i > j). \end{aligned} \tag{5.2}$$

5.2 Learning Graphs with LGMRF from Multivariate Time-Series Data

Here, we present an overview of a generic formulation of SGL and (single-resolution) TVGL methods based on LGMRF in the literature because our formulation also leverages it. The formulation is also useful to distinguish the proposed method from existing methods.

Several existing approaches can be written using this generic formulation, although they have been proposed independently. We derive some representative graph learning methods from the generic formulation and reveal the relationship between them.

5.2.1 General Formulation

Suppose that multivariate time-series data $\{\mathbf{x}_t\}_{t=0}^{T-1}$ are given where $\mathbf{x}_t \in \mathbb{R}^N$ and T is the duration of the time-series. We also assume that $\mathbf{x}_t$ is generated by LGMRF as follows:

$$\mathrm{p}(\mathbf{x}_t|\mathbf{L}_t) = (2\pi)^{-N/2} \left(\mathrm{gdet}(\mathbf{L}_t^\dagger)\right)^{-1/2} \exp\left(-\frac{1}{2}\mathbf{x}_t^\mathsf{T}\mathbf{L}_t\mathbf{x}_t\right), \tag{5.3}$$

where $\mathbf{L}_t$ is the graph Laplacian at time t that corresponds to the underlying graph of $\mathbf{x}_t$, and gdet represents the generalized determinant [82]. Letting $\mathbf{L}_t = \mathcal{L}\mathbf{w}_t$, $\mathbf{w}_t \in \mathbb{R}^E$, and $E = N(N-1)/2$, (5.3) can be rewritten as

$$\mathrm{p}(\mathbf{x}_t|\mathbf{w}_t) = (2\pi)^{-N/2} \left(\mathrm{gdet}((\mathcal{L}\mathbf{w}_t)^\dagger)\right)^{-1/2} \exp\left(-\frac{1}{2}\mathbf{x}_t^\mathsf{T}\mathcal{L}\mathbf{w}_t\mathbf{x}_t\right). \tag{5.4}$$

The edge weight vector $\mathbf{w}_t$ will be sparse and nonnegative. Assume that the prior distribution of $\mathbf{w}_t$ is the following E-variate exponential distribution [39]:

$$p(\mathbf{w}_t) = \left(\frac{\alpha}{2}\right)^E \exp\left(-\frac{\alpha}{2}\mathbf{1}^\mathsf{T}\mathbf{w}_t\right) \quad \text{for } \mathbf{w}_t \geq \mathbf{0}, \tag{5.5}$$

The maximum a posteriori (MAP) estimation of $\mathrm{p}(\mathbf{x}_t|\mathbf{w}_t)$ leads to the following optimization problem:

$$\min_{\mathbf{w}_0 \ldots \mathbf{w}_{T-1} \geq 0} \sum_{t=0}^{T-1} -\log \mathrm{gdet}(\mathcal{L}\mathbf{w}_t) + \alpha\|\mathbf{w}_t\|_1 + \mathbf{x}_t^\mathsf{T}\mathcal{L}\mathbf{w}_t\mathbf{x}_t, \tag{5.6}$$

where α controls the sparsity of the graph.

The optimization problem in (5.6) is the general form for the SGL and TVGL problems based on LGMRF. We also utilize (5.6) for the multiresolution TVGL proposed in this study.

While the problem itself is generic, directly solving this problem is impractical because it needs to learn one graph from one data sample due to overfitting. Therefore, we often need to divide the data into multiple segments. By assuming that the data within the same segment have one common graph, (5.6) is reduced to well-known SGL and TVGL problems. In the following subsections, we derive representative SGL and TVGL settings from (5.6).

5.2.2 Static Graph Learning

Suppose that $\mathbf{w}_t$ is constant for all time instances t, i.e., the graph is static over time. This leads to $\mathbf{w}_t = \mathbf{w}$, $t = 0, \ldots T-1$. Then, (5.6) is reduced to the

following problem:

$$\min_{\mathbf{w}\geq 0} \; -\log \operatorname{gdet}(\mathcal{L}\mathbf{w}) + \alpha\|\mathbf{w}\|_1 + \frac{1}{T}\sum_{t=0}^{T-1} \mathbf{x}_t^\top \mathcal{L}\mathbf{w}\mathbf{x}_t. \tag{5.7}$$

The third term in (5.7) represents the smoothness of the signal as the Laplacian quadratic form $\mathbf{x}_t^\top \mathbf{L}\mathbf{x}_t$. This term can be rewritten by using the sample covariance matrix $\mathbf{S}$ as

$$\sum_{t=0}^{T-1} \mathbf{x}_t^\top \mathcal{L}\mathbf{w}\mathbf{x}_t = \operatorname{tr}(\sum_{t=0}^{T-1} \mathbf{x}_t\mathbf{x}_t^\top \mathcal{L}\mathbf{w}) = T\operatorname{tr}(\mathbf{S}\mathcal{L}\mathbf{w}). \tag{5.8}$$

As a result, (5.7) can be rewritten using (5.8) as

$$\min_{\mathbf{w}\geq 0} \; -\log \operatorname{gdet}(\mathcal{L}\mathbf{w}) + \alpha\|\mathbf{w}\|_1 + \operatorname{tr}(\mathbf{S}\mathcal{L}\mathbf{w}). \tag{5.9}$$

It is equivalent to the graphical Lasso problem with graph Laplacian constraints [39].

5.2.3 Time-varying Graph Learning

As previously mentioned, learning one graph from one sample from (5.6) causes overfitting. Instead, we consider a TVGL problem by dividing the time-series data with nonoverlapping time windows in the same manner as [49, 50, 54].

Let $\mathbf{X}^{(k)} = [\mathbf{x}_{kr}, \dots \mathbf{x}_{(k+1)r-1}] \quad (k = 0, \dots, K-1)$ be the kth data chunk where r is the time window size and k is the index for the time window. We denote $\mathbf{w}^{(k)}$ as the edge weight vector corresponding to the underlying graph of $\mathbf{X}^{(k)}$.

Under the assumption that the graph within the same time window is fixed, i.e., $\mathbf{w}_t = \mathbf{w}^{(k)}, (t = kr, \dots, (k+1)r-1)$, TVGL is formulated as follows:

$$\begin{aligned} \min_{\mathbf{w}^{(0)}\dots\mathbf{w}^{(K-1)}\geq 0} \; & \sum_{k=0}^{K-1} -\log \operatorname{gdet}(\mathcal{L}\mathbf{w}^{(k)}) + \alpha\|\mathbf{w}^{(k)}\|_1 \\ + \frac{1}{r}\operatorname{tr}((\mathbf{X}^{(k)})^\top \mathcal{L}\mathbf{w}^{(k)}\mathbf{X}^{(k)}) & + \beta\sum_{k=1}^{K-1} \psi(\mathbf{w}^{(k)} - \mathbf{w}^{(k-1)}) \end{aligned} \tag{5.10}$$

where $\psi(\cdot)$ is an additional regularizer that characterizes the temporal evolution based on the prior knowledge of time-varying graphs and β is its parameter. Note that the problem is identical to the SGL in (5.7) if $\psi(\cdot) = 0$. The algorithm to solve (5.10) is described in Appendix A.

As possible regularizers, $\psi(\cdot) = \|\cdot\|_2^2$ reflects a time-varying graph whose edge weights change smoothly over time, and $\psi(\cdot) = \|\cdot\|_1$ leads to the graph wherein only a small number of edges change at any given time. A similar problem to (5.10) is proposed in [81].

This approach is effective as long as we have appropriate prior knowledge of the temporal evolution, i.e., $\psi(\cdot)$, and the accurate window size r. However, an inappropriate choice of $\psi(\cdot)$ or r leads to inappropriate graphs. To tackle this problem, we propose TMR TVGL in the next section.

5.3 Multiresolution Time-Varying Graph Learning

In this section, we present the formulation of the TMR TVGL and an algorithm for solving it.

5.3.1 Formulation

Here, we introduce a TVGL method that learns $\{\mathbf{w}_t\}_{t=0}^{T-1}$ based on a multiresolution assumption. For simplicity, suppose that T is divisible by 2^L, however, this method is applicable to general values of T.

Suppose that $\mathbf{W}_t$ can be represented as a combination of graphs localized at a temporal resolution $\mathbf{W}_{l,m}$, as illustrated in Fig. 5.1. We refer to $\mathbf{W}_{l,m}$ as the TMR graph at the temporal resolution l and the segment index m. Therefore, the multiscale representation of $\{\mathbf{w}_t\}_{t=0}^{T-1}$ is given by the sum of TMR graphs corresponding to time t as

$$\mathbf{w}_t \approx \bar{\mathbf{w}}_t = \sum_{i=0}^{L} \mathbf{w}_{i,\lfloor q(t)/2^{(L-i)} \rfloor}, \tag{5.11}$$

where $q(t) = \left\lfloor \frac{t}{T} 2^L \right\rfloor$ and L is the maximum temporal resolution level.

This TMR representation has two advantages. First, it reduces the number of parameters to learn. For TMR TVGL, we need $E(2^{(L+1)} - 1)$ parameters, whereas the number of parameters in a single-resolution TVGL is ET. $E(2^{(L+1)} - 1) \leq ET$ when $L \leq \log_2 T - 1$. It is beneficial if we only have a limited amount of available data. Second, the TMR representation enables the capture of the edges localized in an arbitrary temporal resolution, without specifying the temporal window size.

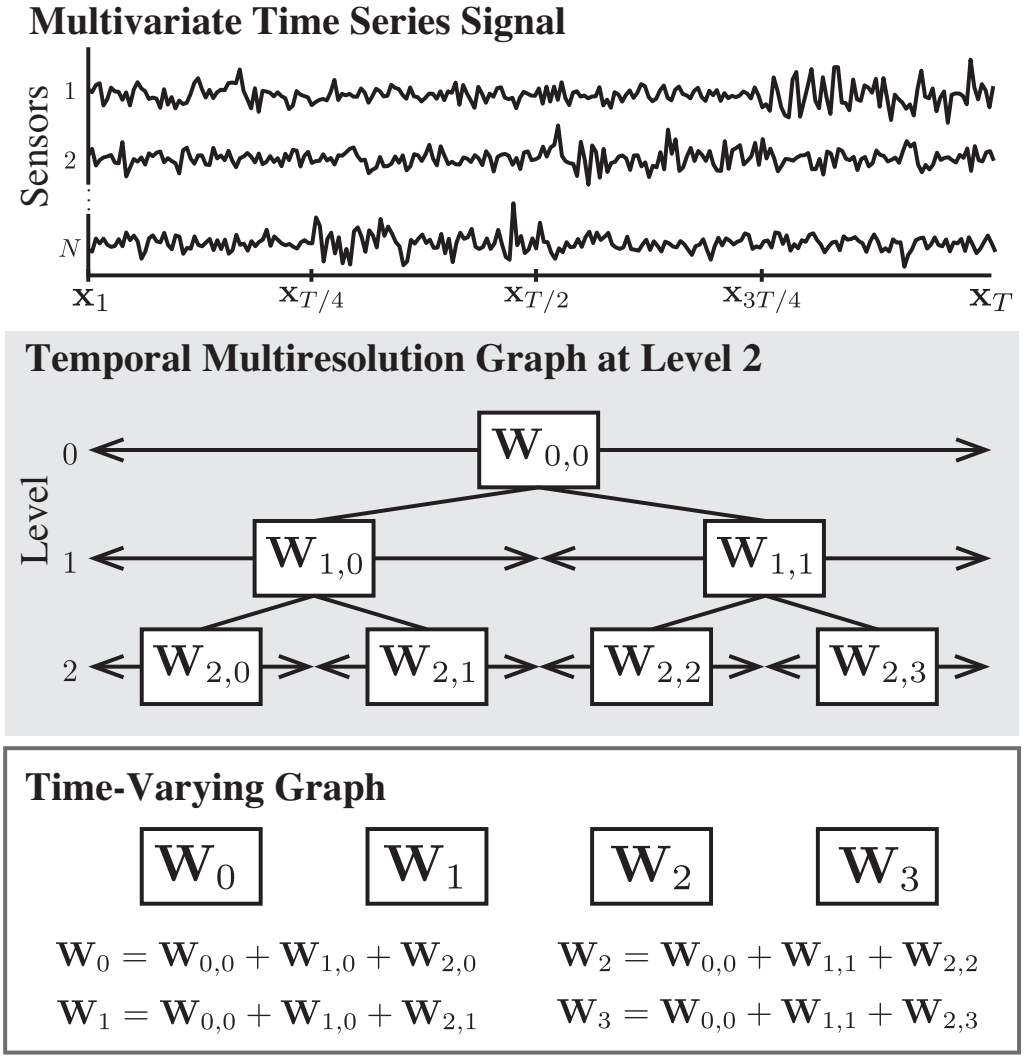


FIGURE 5.1: Overview of multiresolution graph learning.

Now, we consider the detailed formulation of the proposed TVGL. The goal is to learn $\mathbf{w}_{l,m}$ from $\{\mathbf{x}_t\}_{t=0}^{T-1}$. Substituting (5.11) into (5.6) leads to the following problem:

$$\begin{aligned}\min_{\mathbf{w}_{0,0},\ldots,\mathbf{w}_{L,2^L-1}\geq 0} \sum_{t=0}^{T-1} -\log \mathrm{gdet}\left(\mathcal{L}(\sum_{k=0}^{L}\mathbf{w}_{k,\lfloor q(t)/2^{(L-k)}\rfloor})\right)\\ +\alpha\|\sum_{k=0}^{L}\mathbf{w}_{k,\lfloor q(t)/2^{(L-k)}\rfloor}\|_1+\mathbf{x}_t^{\mathsf{T}}\mathcal{L}(\sum_{k=0}^{L}\mathbf{w}_{k,\lfloor q(t)/2^{(L-k)}\rfloor})\mathbf{x}_t.\end{aligned} \tag{5.12}$$

Letting

$$\begin{aligned}\mathbf{F}&=[\mathbf{w}_{0,0},\mathbf{w}_{1,0},\mathbf{w}_{1,1},\ldots,\mathbf{w}_{L,2^L-1}]\in\mathbb{R}^{E\times 2^{(L+1)}-1},\\ \mathbf{X}_k&=[\mathbf{x}_{kr},\ldots\mathbf{x}_{kR+R-1}]\quad(k=0,\ldots,2^L-1,R=T/2^L),\end{aligned} \tag{5.13}$$

(5.12) can be rewritten as:

$$\begin{aligned}\min_{\mathbf{F}\geq 0}\alpha\|\mathbf{FM}\|_1+\sum_{k=0}^{2^L-1}\frac{1}{R}\mathrm{tr}((\mathbf{X}_k)^{\mathsf{T}}\mathcal{L}([\mathbf{FM}]_k)\mathbf{X}_k)\\ -\log\mathrm{gdet}(\mathcal{L}([\mathbf{FM}]_k)),\end{aligned} \tag{5.14}$$

where $\mathbf{M}\in\mathbb{R}^{2^{(L+1)}-1\times 2^L}$ is given by

$$[\mathbf{M}]_{i,j}=\begin{cases}1 & 2^{L-l}m\leq j\leq 2^{L-l}(m+1)-1\\ 0 & \text{otherwise,}\end{cases}$$

in which $l=\lfloor\log_2(i+1)\rfloor$ and $m=\mathrm{mod}(i+1,2^l)$. Note that $[\mathbf{FM}]_k=\bar{\mathbf{w}}_{kR}=\cdots=\bar{\mathbf{w}}_{kR+R-1}$.

In (5.14), we need to obtain a sparse $\mathbf{w}_{l,m}$ to capture the temporally localized structure. However, the direct constraint on the sparseness of $\mathbf{FM}$ does not result in a sparse $\mathbf{F}$. Therefore, we replace the first term in (5.14) with the sparse constraint on $\mathbf{F}$ as follows:

$$\begin{aligned}\min_{\mathbf{F}\geq 0}\alpha\|\mathbf{F}\|_1+\sum_{k=0}^{2^L-1}\frac{1}{R}\mathrm{tr}((\mathbf{X}_k)^{\mathsf{T}}\mathcal{L}([\mathbf{FM}]_k)\mathbf{X}_k)\\ -\log\mathrm{gdet}(\mathcal{L}([\mathbf{FM}]_k).\end{aligned} \tag{5.15}$$

This is the proposed TVGL formulation for learning TMR graphs. In the following subsection, we describe an algorithm to solve (5.15).

5.3.2 Algorithm

The optimization problem in (5.15) is convex and can be solved using the PDS algorithm. Here, we reformulate (5.15) to the PDS applicable form.

Let $\mathbf{Z}_k \in \mathbb{R}^{N\times N}$ be a pairwise distance matrix computed from

$$[\mathbf{Z}_k]_{i,j} = \sum_{n=0}^{R-1} \|[\mathbf{X}_k]_{i,n} - [\mathbf{X}_k]_{j,n}\|^2, \tag{5.16}$$

and $\mathbf{z}_k \in \mathbb{R}^E$ be the vector form representation of $\mathbf{Z}_k$. The third term of (5.15) can then be rewritten as

$$\begin{aligned} &\sum_{k=0}^{2^L-1} \mathrm{tr}((\mathbf{X}_k)^\mathsf{T}\mathcal{L}([\mathbf{FM}]_k)\mathbf{X}_k) \\ &= \sum_{k=0}^{2^L-1} \mathbf{z}_k^\mathsf{T}[\mathbf{FM}]_k = \|\mathbf{Z}_{\mathrm{all}} \circ (\mathbf{FM})\|_1 = \|(\mathbf{MZ}_{\mathrm{all}}^\mathsf{T}) \circ \mathbf{F}^\mathsf{T}\|_1 \end{aligned} \tag{5.17}$$

where $\mathbf{Z}_{\mathrm{all}} = [\mathbf{z}_0, \ldots, \mathbf{z}_{2^L-1}]$. Here, we denote $\bar{\mathbf{F}} = \mathbf{F}^\mathsf{T}$ and $\bar{\mathcal{L}}_i\mathbf{X} = \mathcal{L}(\mathbf{X}^\mathsf{T})_i$ for notation simplicity. By using the indicator function, (5.15) can be reduced to the following optimization problem:

$$\begin{aligned} \min_{\bar{\mathbf{F}}} \alpha\|\bar{\mathbf{F}}\|_1 + \frac{1}{R}\|\mathbf{MZ}_{\mathrm{all}}^\mathsf{T} \circ \bar{\mathbf{F}}\|_1 + \iota(\bar{\mathbf{F}}) \\ - \sum_{k=0}^{2^L-1} \log \mathrm{gdet}(\bar{\mathcal{L}}_i(\mathbf{M}^\mathsf{T}\bar{\mathbf{F}})). \end{aligned} \tag{5.18}$$

where ι is defined by

$$\iota(\bar{\mathbf{F}}) = \begin{cases} 0 & \bar{\mathbf{F}} \geq 0 \\ \infty & \text{otherwise.} \end{cases} \tag{5.19}$$

Owing to the nonnegative constraint on $\bar{\mathbf{F}}$, the first and second terms in (5.18) can be merged as

$$\begin{aligned} \min_{\bar{\mathbf{F}}} \frac{1}{R}\|(\alpha\mathbf{H} + \mathbf{MZ}_{\mathrm{all}}^\mathsf{T}) \circ \bar{\mathbf{F}}\|_1 + \iota(\bar{\mathbf{F}}) \\ - \sum_{k=0}^{2^L-1} \log \mathrm{gdet}(\bar{\mathcal{L}}_i(\mathbf{M}^\mathsf{T}\bar{\mathbf{F}})), \end{aligned} \tag{5.20}$$

where $\mathbf{H} = \mathbf{1}\mathbf{1}^\mathsf{T} \in \mathbb{R}^{E \times 2^{(L+1)}-1}$. By introducing the linear operator $\bar{\mathcal{L}} : \mathbb{R}^{E \times 2^L} \to \mathbb{R}^{2^L N \times N}$ defined as

$$\bar{\mathcal{L}}(\mathbf{X}) = [\bar{\mathcal{L}}_0(\mathbf{X}), \ldots, \bar{\mathcal{L}}_{2^L-1}(\mathbf{X})]^\mathsf{T}, \tag{5.21}$$

and a dual variable $\mathbf{V} := [\mathbf{V}_0^\mathsf{T}, \ldots, \mathbf{V}_{2^L-1}^\mathsf{T}]^\mathsf{T} = \bar{\mathcal{L}}(\mathbf{M}^\mathsf{T}\bar{\mathbf{F}})$, we can convert (5.20) into the form in (2.35) as follows:

$$\begin{aligned} f(\bar{\mathbf{F}}) &= 0, \\ g(\bar{\mathbf{F}}) &= \frac{1}{R}\|(\alpha\mathbf{H} + \mathbf{M}\mathbf{Z}_{\text{all}}^\mathsf{T}) \circ \bar{\mathbf{F}}\|_1 + \iota(\bar{\mathbf{F}}), \\ h(\mathbf{V}) &= -\sum_{k=0}^{2^L-1} \log \operatorname{gdet}(\mathbf{V}_k). \end{aligned} \tag{5.22}$$

The proximal operator for the function g corresponds to that of the weighted ℓ_1 norm with the nonnegative constraint, and it is given by

$$\left[\operatorname{prox}_{\gamma g}(\mathbf{A})\right]_{i,j} = \begin{cases} 0 & [\mathbf{A}]_{i,j} \le \gamma[\mathbf{B}]_{i,j} \\ [\mathbf{A}]_{i,j} - \gamma[\mathbf{B}]_{i,j} & \text{otherwise,} \end{cases} \tag{5.23}$$

where $\mathbf{B} = \alpha\mathbf{H} + \mathbf{M}\mathbf{Z}_{\text{all}}^\mathsf{T}$.

The proximal operator of h can be computed as follows. In general, the logarithm of a generalized determinant is a nonconvex function. Under the assumption that the learned graph is connected (which is often the case), it can be replaced with a convex function as follows [39, Proposition 1]:

$$\log \operatorname{gdet}(\mathbf{A}) = \log \det(\mathbf{A} + \frac{1}{N}\mathbf{1}\mathbf{1}^\mathsf{T}). \tag{5.24}$$

Then, the proximal operator is given by

$$\operatorname{prox}_{\gamma(-\log \operatorname{gdet}(\cdot))}(\mathbf{A}) = \mathbf{U} \begin{bmatrix} \phi(\lambda_0) & & 0 \\ & \ddots & \\ 0 & & \phi(\lambda_{N-1}) \end{bmatrix} \mathbf{U}^\mathsf{T}, \tag{5.25}$$

where $\phi(\lambda_i) = \frac{\lambda_i + \sqrt{\lambda_i^2 + 4\gamma}}{2}$ and $\mathbf{U}$ and λ_i are the eigenvector matrix and the eigenvalue of $\mathbf{A} + \frac{1}{N}\mathbf{1}\mathbf{1}^\mathsf{T}$, respectively. The eignvalues are ordered as $\lambda_0 \le \lambda_1 \le$

$\lambda_2 \cdots \leq \lambda_{N-1}$ [1].

Finally, we present the algorithm for the multiresolution TVGL in Algorithm 4. The condition of convergence is given by

$$\gamma_1\gamma_2\|\mathbf{M}\bar{\mathcal{L}}^*\bar{\mathcal{L}}\mathbf{M}^\mathsf{T}\| < 1. \tag{5.27}$$

Based on the submultiplicativity of the operator norm, the upper-bound of $\|\mathbf{M}\bar{\mathcal{L}}^*\bar{\mathcal{L}}\mathbf{M}^\mathsf{T}\|$ can be computed from

$$\begin{aligned}\|\mathbf{M}\bar{\mathcal{L}}^*\bar{\mathcal{L}}\mathbf{M}^\mathsf{T}\| \leq \|\mathbf{M}\|\|\bar{\mathcal{L}}^*\bar{\mathcal{L}}\|\|\mathbf{M}^\mathsf{T}\| &= \|\bar{\mathcal{L}}^*\bar{\mathcal{L}}\|\|\mathbf{M}\mathbf{M}^\mathsf{T}\| \\ &= N(2^{L+2}-2)\end{aligned} \tag{5.28}$$

because of $\|\bar{\mathcal{L}}^*\bar{\mathcal{L}}\| = 2N$ and $\|\mathbf{M}\mathbf{M}^\mathsf{T}\| = 2^{L+1}-1$. Consequently, the convergence condition in (5.27) can be rewritten as

$$\gamma_1 < \frac{1}{\gamma_2 N(2^{L+2}-2)}. \tag{5.29}$$

The computational complexity of our algorithm is $\mathcal{O}(2^L N^3)$ per iteration.

Algorithm 4 Temporal multiresolution graph learning

Input: $\bar{\mathbf{F}}^{(0)}$, $\mathbf{V}^{(0)}$, $\{\mathbf{x}_t\}_{t=0}^{T-1}$, L, ϵ
Output: $\mathbf{F}$
  Divide $\{\mathbf{x}_t\}_{t=0}^{\mathsf{T}}$ into 2^L data segments $\mathbf{X}_0 \ldots \mathbf{X}_{2^L-1}$
  Compute $\{\mathbf{Z}_k\}_{k=0}^{2^L-1}$ from $\{\mathbf{X}_k\}_{k=0}^{2^L-1}$
  while $\|\bar{\mathbf{F}}^{(i+1)} - \bar{\mathbf{F}}^{(i)}\|/\|\bar{\mathbf{F}}^{(i)}\| > \epsilon$ **do**
    $\bar{\mathbf{F}}^{(i+1)} := \mathrm{prox}_{\gamma_1 g}(\bar{\mathbf{F}}^{(i)} - \gamma_1\mathbf{M}\bar{\mathcal{L}}^*\mathbf{V}^{(i)})$
    $\mathbf{V}^{(i)} \leftarrow \mathbf{V}^{(i)} + \gamma_2\bar{\mathcal{L}}\mathbf{M}(2\bar{\mathbf{F}}^{(i+1)} - \bar{\mathbf{F}}^{(i)})$
    for $k = 0, \ldots, 2^L - 1$ **do**
      $\mathbf{V}_k^{(i+1)} := \mathbf{V}_k^{(i)} - \gamma_2\,\mathrm{prox}_{\frac{1}{\gamma_2}(-\log\mathrm{gdet}(\cdot))}\left(\frac{\mathbf{V}_k^{(i)}}{\gamma_2}\right)$
    end for
    $\mathbf{V}^{(i+1)} = \left[\mathbf{V}_0^{(i+1)\mathsf{T}}, \ldots, \mathbf{V}_{2^L-1}^{(i+1)\mathsf{T}}\right]^\mathsf{T}$
    $i \leftarrow i + 1$
  end while

[1] Even if the original graph has disconnected components, we can avoid the problem of the calculation of the proximal operator by adding a small regularizing parameter c to the input as follows [58]:

$$\log\mathrm{gdet}(\mathbf{A}) \approx \log\det(\mathbf{A} + c^2\mathbf{I}), \tag{5.26}$$

The proximal operator of this approximation also can be computed in the same manner as (5.25).

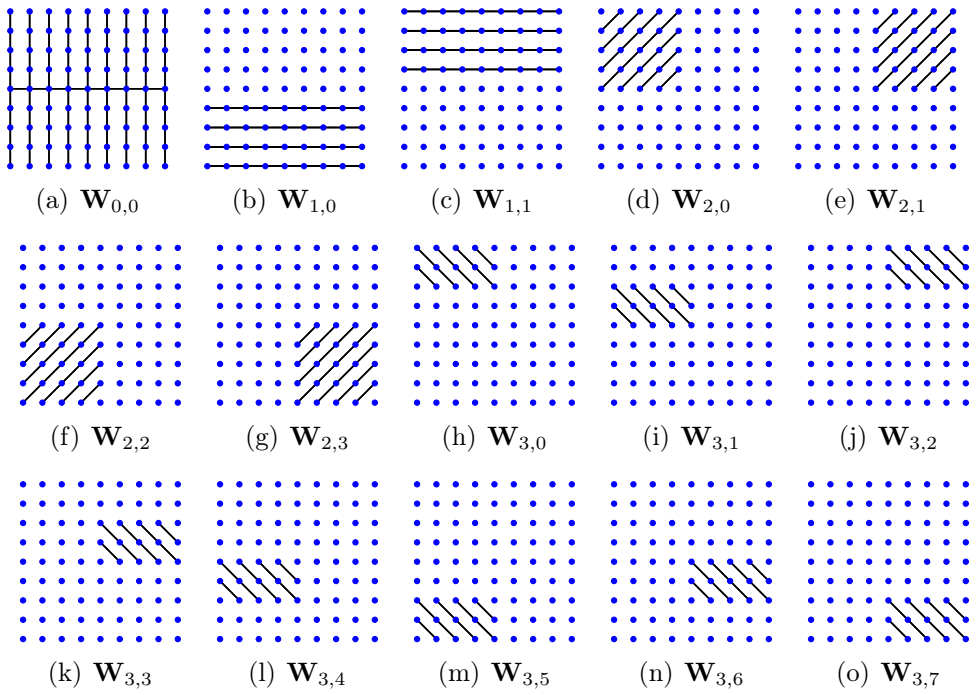


FIGURE 5.2: Visualization of the ground-truth graphs.

5.4 Experimental Results

In this section, we present experimental results on synthetic and real datasets. The existing and proposed methods are abbreviated as follows:

- SGL based on smoothness criterion (SGL-S) [37].
- SGL with LGMRF (SGL-LG) [39].
- TVGL introduced in Section 4 (TVGL-S) [49, 50, 54].
- TVGL with LGMRF incorporating the temporal variation constraint (TVGL-LG) [81].
- Proposed TMR TVGL (TVGL-MR) described in Section 5.3.

The stopping criterion of the iterations for each methods is set to $\|\mathbf{w}^{(n+1)} - \mathbf{w}^{(n)}\|/\|\mathbf{w}^{(n)}\| < 1.0 \times 10^{-3}$.

5.4.1 Experiments on Temporal Multiresolution Graphs

To demonstrate the concept of TVGL for TMR graphs, we first present the results by constructing a simple TMR graph dataset.

TABLE 5.1: Comparison of the performance for learning time-varying graph.

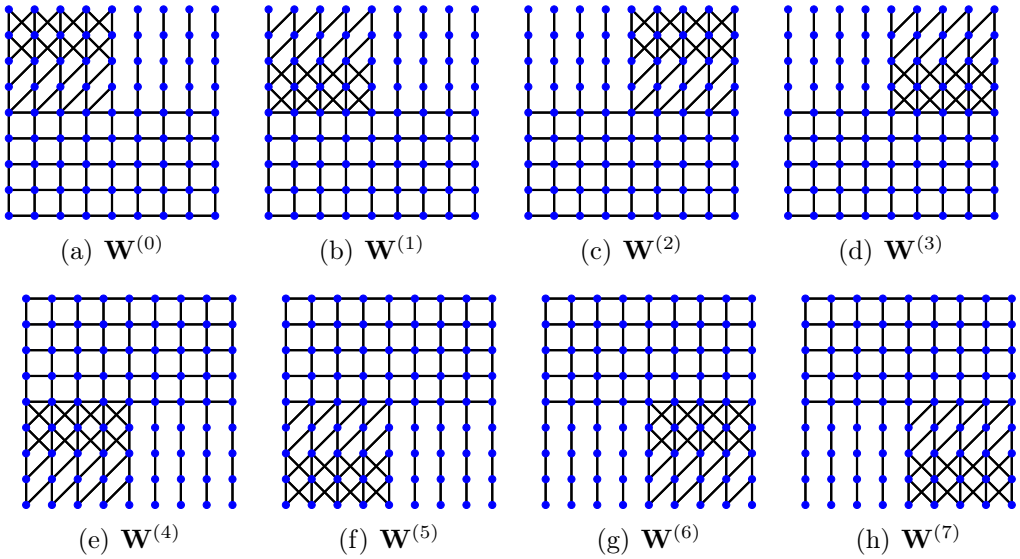
Methods	Window size	F-measure	Relative Error
SGL-S	20	0.571	0.710
	40	0.717	0.624
	80	0.790	0.581
	avg.	0.693	0.638
SGL-LG	20	0.596	0.730
	40	0.726	0.576
	80	0.831	0.431
	avg.	0.718	0.579
TVGL-S	20	0.798	0.479
	40	0.817	0.508
	80	0.855	0.431
	avg.	0.823	0.472
TVGL-LG	20	**0.890**	**0.370**
	40	0.888	0.419
	80	0.871	0.513
	avg.	0.883	0.434
TVGL-MR	20	0.842	0.415
	40	**0.895**	**0.341**
	80	**0.915**	**0.314**
	avg.	**0.884**	**0.357**

Dataset

The dataset is constructed in two sequential steps: 1) construction of time-varying graphs and 2) generation of data samples based on the time-varying graphs.

First, we construct TMR graphs with four levels ($l = 0, \ldots, 3$) as shown in Fig. 5.2. The number of vertices N is set to $N = 81$ and the edge weights between vertices are random values drawn from a uniform distribution from the interval $[0.1, 3]$. The lowest resolution graph, i.e., the graph reflecting the global structure, is $\mathbf{W}_{0,0}$, as shown in Fig. 5.2(a), where the graph has a grid-like structure while the edges only run vertically, except for the horizontal edges at the center of the grid. As shown in Figs. 5.2(b)–(o), the graphs at levels 1 to 3 have horizontal edges, diagonal edges from the upper right to lower left, and diagonal edges from the upper left to lower right, respectively. By combining $\mathbf{W}_{l,m}$'s, we obtain prototype graphs $\mathbf{W}^{(0)}, \ldots, \mathbf{W}^{(7)}$, as shown in Fig. 5.3.

From the prototype graphs, we then construct time-varying graphs $\{\mathbf{W}_0, \ldots, \mathbf{W}_{T-1}\}$. We set $T = 640$ in this experiment. As the number of mutiresolution graphs in the highest resolution is eight, each of them has been duplicated 80 times and



(a) $\mathbf{W}^{(0)}$ (b) $\mathbf{W}^{(1)}$ (c) $\mathbf{W}^{(2)}$ (d) $\mathbf{W}^{(3)}$

(e) $\mathbf{W}^{(4)}$ (f) $\mathbf{W}^{(5)}$ (g) $\mathbf{W}^{(6)}$ (h) $\mathbf{W}^{(7)}$

FIGURE 5.3: Time-varying graphs obtained from the multiresolution graphs in Fig. 5.2.

then they are concatenated, i.e., $\mathbf{W}_t := \mathbf{W}^{(\lfloor t/80 \rfloor)}$ $(t = 0, \ldots, T-1)$.

Second, multivariate time-series signals $\mathbf{X}$ are generated from the following GMRF:

$$\mathbf{x}_t \sim \mathcal{N}(0, (\mathbf{L}_t + \sigma^2 \mathbf{I})^\dagger), \tag{5.30}$$

where $\mathbf{L}_t$ is the graph Laplacian associated with $\mathbf{W}_t$. We set σ to 0.5.

Experimental Condition

We evaluate the performance in terms of relative error and F-measure, each averaged over all time slots. Relative error is given by

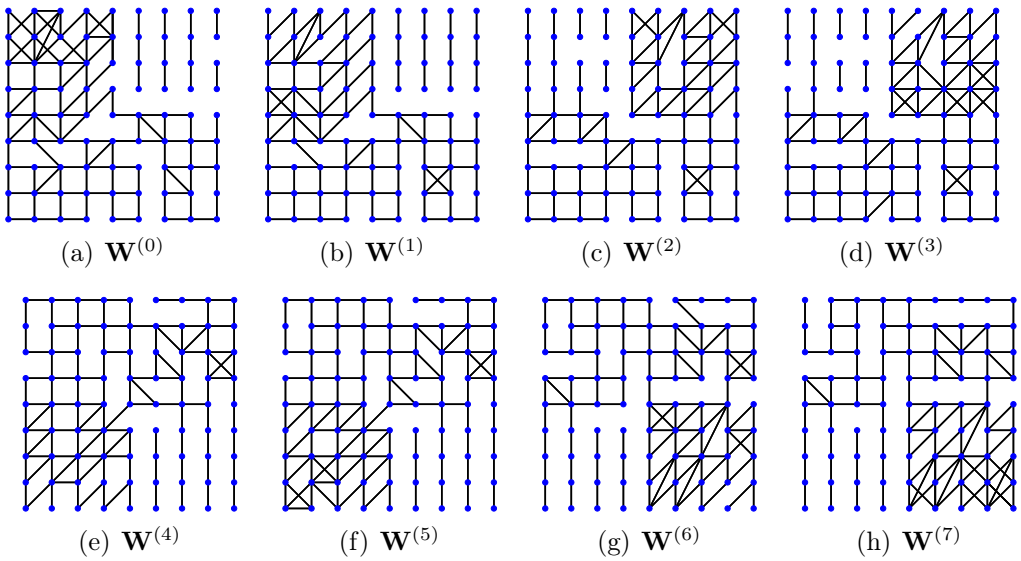
$$\text{Relative error} = \frac{\|\widehat{\mathbf{W}} - \mathbf{W}^*\|_F}{\|\mathbf{W}^*\|_F}, \tag{5.31}$$

where $\widehat{\mathbf{W}}$ is the estimated weighted adjacency matrix, and $\mathbf{W}^*$ is the ground-truth. It reflects the accuracy of edge weights on the estimated graph.

The F-measure is given by

$$\text{F-measure} = \frac{2\text{tp}}{2\text{tp} + \text{fn} + \text{fp}}, \tag{5.32}$$

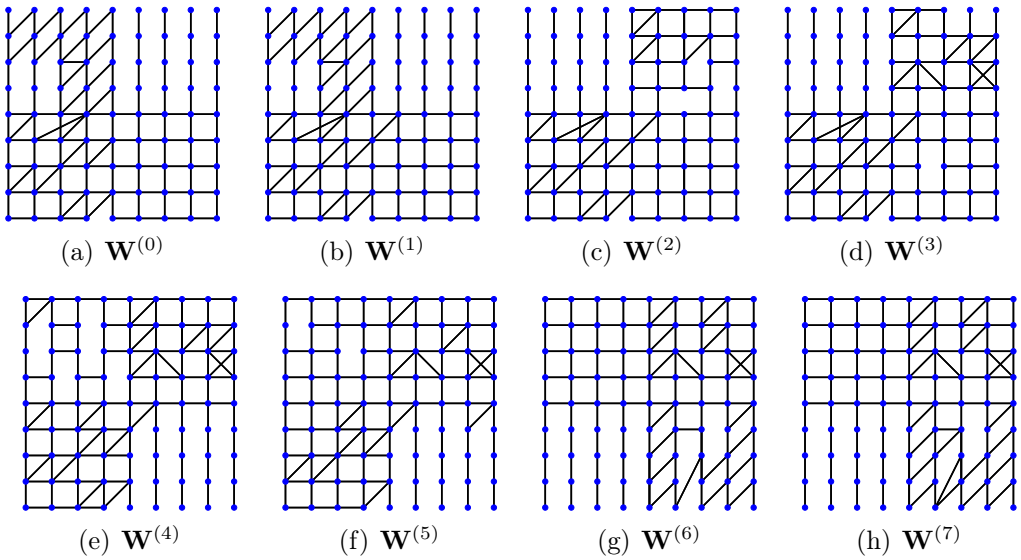
where the true positive (tp) is the number of edges that are included both in $\widehat{\mathbf{W}}$ and $\mathbf{W}^*$, the false positive (fn) is the number of edges that are not included in $\widehat{\mathbf{W}}$ but are included in $\mathbf{W}^*$, and the false positive (fp) is the number of edges that are included in $\widehat{\mathbf{W}}$ but are not included in $\mathbf{W}^*$. The F-measure, which



(a) $\mathbf{W}^{(0)}$ (b) $\mathbf{W}^{(1)}$ (c) $\mathbf{W}^{(2)}$ (d) $\mathbf{W}^{(3)}$

(e) $\mathbf{W}^{(4)}$ (f) $\mathbf{W}^{(5)}$ (g) $\mathbf{W}^{(6)}$ (h) $\mathbf{W}^{(7)}$

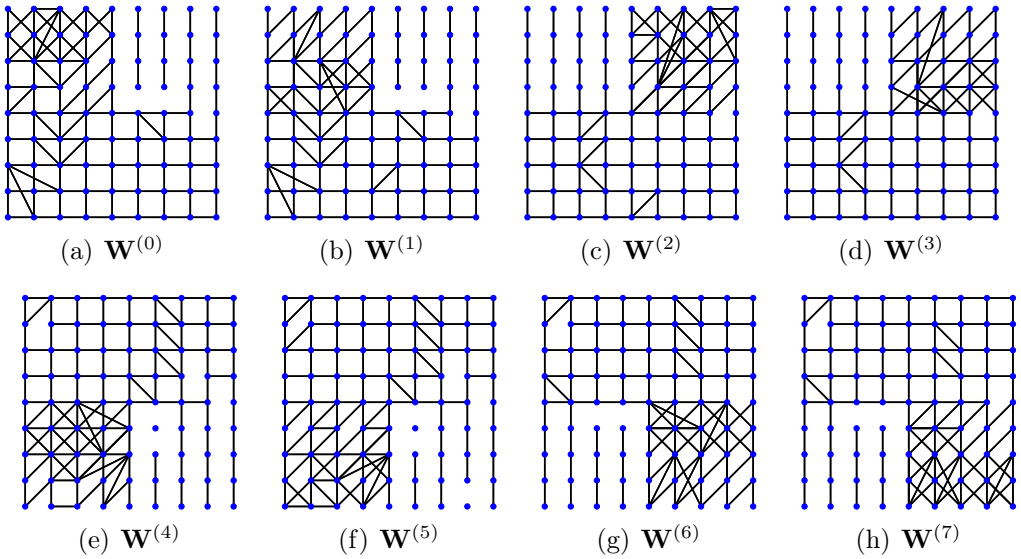
FIGURE 5.4: Visualization of time-varying graphs learned by TVGL-S



(a) $\mathbf{W}^{(0)}$ (b) $\mathbf{W}^{(1)}$ (c) $\mathbf{W}^{(2)}$ (d) $\mathbf{W}^{(3)}$

(e) $\mathbf{W}^{(4)}$ (f) $\mathbf{W}^{(5)}$ (g) $\mathbf{W}^{(6)}$ (h) $\mathbf{W}^{(7)}$

FIGURE 5.5: Visualization of time-varying graphs learned by TVGL-LG



(a) $\mathbf{W}^{(0)}$ (b) $\mathbf{W}^{(1)}$ (c) $\mathbf{W}^{(2)}$ (d) $\mathbf{W}^{(3)}$

(e) $\mathbf{W}^{(4)}$ (f) $\mathbf{W}^{(5)}$ (g) $\mathbf{W}^{(6)}$ (h) $\mathbf{W}^{(7)}$

FIGURE 5.6: Visualization of time-varying graphs learned by TVGL-MR

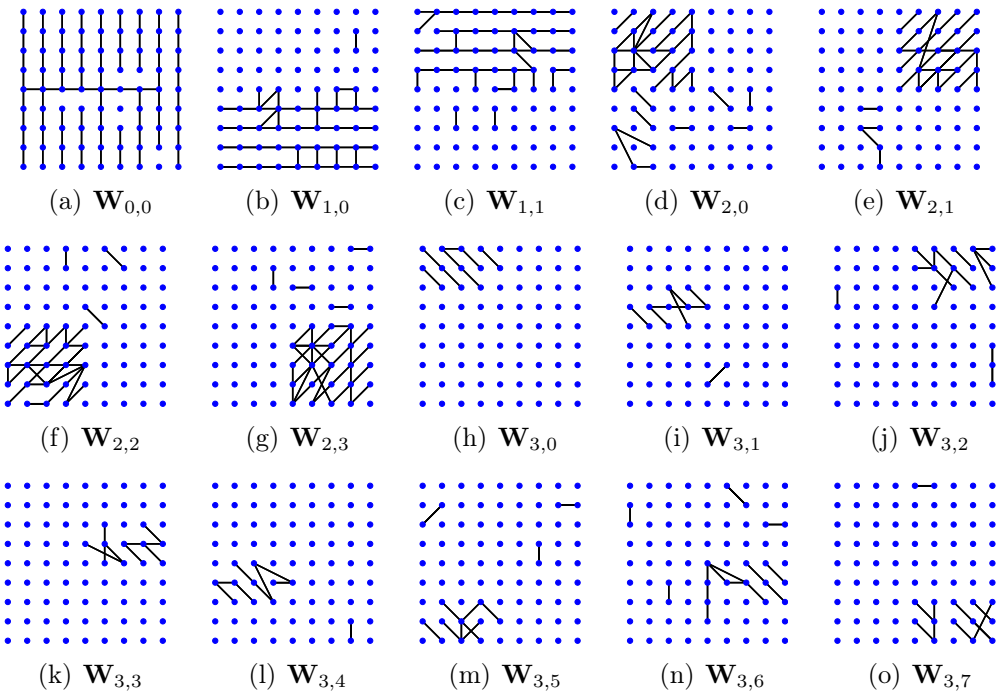
is the harmonic average of the precision and recall, represents the accuracy of the estimated graph topology. The F-measure takes values between 0 and 1. The higher the F-measure, the higher the performance of capturing the graph topology.

In this experiment, we construct training and test data and evaluate the performance of graph learning on the test data using the hyperparameters that minimize the relative error on the training data. We search for optimal hyperparameters using Bayesian optimization [83]. Additionally, ℓ_1 norm is used for the temporal variation regularization of the existing TVGL approaches.

We evaluate the performance with different window sizes to study the robustness of each method for the choice of the window size K. The existing methods use $K = 20$, 40, or 80, and the proposed method uses the maximum temporal resolution level $L = 5$. The proposed method can reconstruct time-varying graphs corresponding to $K = \{20, 40, 80\}$ from a set of TMR graphs. Note that the existing methods need to fix K before running their algorithms, whereas the proposed TVGL method simultaneously estimates time-varying graphs in the different window sizes.

Results

Table 5.1 summarizes the average performance of the learned graphs. As shown in the table, TVGL-MR nearly outperforms the other methods both in terms of F-measure and relative error. This indicates that the TVGL performances



(a) $\mathbf{W}_{0,0}$ (b) $\mathbf{W}_{1,0}$ (c) $\mathbf{W}_{1,1}$ (d) $\mathbf{W}_{2,0}$ (e) $\mathbf{W}_{2,1}$

(f) $\mathbf{W}_{2,2}$ (g) $\mathbf{W}_{2,3}$ (h) $\mathbf{W}_{3,0}$ (i) $\mathbf{W}_{3,1}$ (j) $\mathbf{W}_{3,2}$

(k) $\mathbf{W}_{3,3}$ (l) $\mathbf{W}_{3,4}$ (m) $\mathbf{W}_{3,5}$ (n) $\mathbf{W}_{3,6}$ (o) $\mathbf{W}_{3,7}$

FIGURE 5.7: Visualization of the TMR graphs learned by TVGL-MR.

can be improved by TVGL-MR if time-varying graphs can be assumed to have multiresolution characteristics.

Figs. 5.4, 5.5 and 5.6 visualizes the time-varying graphs learned by TVGL-S, TVGL-LG, and TVGL-MR. As shown in these figures, the alternative TVGL methods fail to capture temporal multiresolution structures, particularly those at the high-resolution level. In contrast, the proposed method captures edges localized at various temporal resolutions.

Furthermore, Fig. 5.7 shows the TMR graphs learned by TVGL-MR. The figure also demonstrates that the proposed method can successfully learn TMR graphs.

5.4.2 Experiments on Single Resolution Graphs

The previous experiment demonstrates the effectiveness of the proposed method for TMR graphs. While the proposed method is not specifically designed for learning single resolution TVGL, here, we compare TVGL performances with the other methods for some single resolution time-varying graphs.

Datasets

The dataset is constructed with the same steps described in Section 5.4.1. In this experiment, we construct two types of time-varying graphs as follows:

Edge-Markovian Evolving Graph (EMEG): EMEG is a stochastic time dependency evolving graph [84]. Each edge in EMEG follows the Markovian process. EMEG $\mathcal{G}_s = \{\mathcal{G}_t = (\mathcal{V}_t, \mathcal{E}_t, \mathbf{W}_t)\}$ is satisfied as

$$\begin{cases} p\left(e \in \mathcal{E}_{t+1} \mid e \notin \mathcal{E}_t\right) = q_1 \\ p\left(e \notin \mathcal{E}_{t+1} \mid e \in \mathcal{E}_t\right) = q_2 \end{cases} \tag{5.33}$$

where q_1 and q_2 are called *birth rate* and *death rate*, respectively. We generate an Erdős–Rényi graph with $N = 36$, $p = 0.1$ as the initial graph $\mathcal{G}_0$. The edge weights of the initial graph are selected from the uniform distribution with the interval $[0.1, 3]$, and the weights of the newborn edges are also selected from the same distribution. We set $q_1 = 0.001$ and $q_2 = 0.01$.

Switching Behavior Graph (SBG): SBG is a time-varying graph that exhibits the transition of connectivity states. It often appears in brain connectivity dynamics [75, 76]. We construct an SBG using the following procedure. We generate six static graphs used as the connectivity states. Each of the graphs is initialized to an Erdős–Rényi graph with $N = 36$, an edge connection probability $p = 0.05$, and edge weights drawn from a uniform distribution in the interval $[0.1, 3]$. The initial state is selected randomly from the six connectivity states, and its state remains with a 98% probability and transits to another connectivity state with the 2% probability at each time.

Generating Graph Signals: Given graph Laplacians $\mathbf{L}^{(0)}, \ldots, \mathbf{L}^{(127)}$ of the constructed time-varying graphs, we generate multivariate time-series signal $\mathbf{x}_0, \ldots, \mathbf{x}_{5119}$ from the following GMRF:

$$\mathbf{x}_t \sim \mathcal{N}(0, (\mathbf{L}^{(\lfloor t/40 \rfloor)} + \sigma^2 \mathbf{I})^\dagger), \tag{5.34}$$

where σ^2 is the variance of the white Gaussian noise. We set $\sigma = 0.5$ in this experiment.

Regularization Functions for Alternative TVGL Methods: TVGL methods, i.e., TVGL-S and TVGL-LG, require choosing the regularization function based on the prior knowledge of temporal graph evolution. For EMEG and SBG, we adopt ℓ_1 and $\ell_{2,1}$-norm as the possible regularization functions, respectively.

TABLE 5.2: Comparison of the F-measure and relative error for learning time-varying graph. The bold and underlined values represent the best and second-best performance among the methods, respectively.

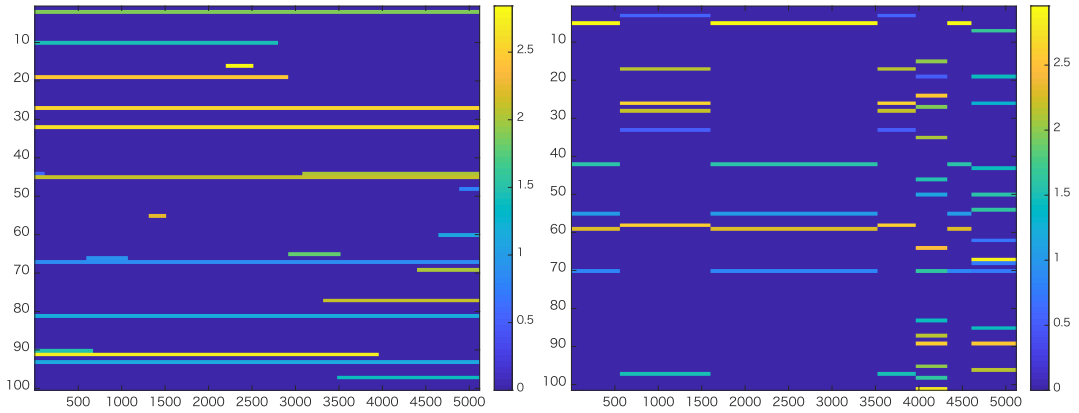
Dataset	WS	SGL-S	SGL-LG	TVGL-S	TVGL-LG	TVGL-MR
F-measure						
EMEG	10	0.320	0.367	0.525	<u>0.710</u>	**0.800**
	20	0.389	0.448	0.523	<u>0.748</u>	**0.776**
	40	0.495	0.530	0.518	<u>0.727</u>	**0.772**
	avg.	0.401	0.448	0.522	<u>0.601</u>	**0.783**
SBG	10	0.303	0.313	**0.804**	0.418	<u>0.754</u>
	20	0.449	0.382	**0.800**	0.579	<u>0.722</u>
	40	0.524	0.436	**0.818**	0.664	<u>0.696</u>
	avg.	0.425	0.377	**0.807**	0.557	<u>0.724</u>
Relative error						
EMEG	10	0.885	0.906	0.537	**0.191**	<u>0.296</u>
	20	0.874	0.832	0.532	**0.196**	<u>0.262</u>
	40	0.720	0.713	0.527	<u>0.269</u>	**0.254**
	avg.	0.826	0.817	0.532	**0.219**	<u>0.271</u>
SBG	10	0.882	0.888	<u>0.419</u>	0.571	**0.410**
	20	0.777	0.774	0.399	**0.349**	<u>0.376</u>
	40	0.658	0.618	0.402	**0.354**	<u>0.359</u>
	avg.	0.772	0.760	<u>0.406</u>	0.424	**0.382**

Results

Table 5.2 summarizes the performances of SGL/TVGL methods on different datasets. TVGL methods outperform the static methods on all datasets. This implies that the regularization for the temporal graph evolution or TMR assumption improves the graph learning performance.

Among the TVGL methods, TVGL-MR ranks first or second in this experiment. This suggests the effectiveness and robustness of the proposed method even for single resolution time-varying graphs. It is also worth noting that, TVGL-MR can exhibit performance comparable to that of time-varying methods without the prior knowledge of the graph evolution over time, i.e., the regularization function. Typically, existing TVGL approaches require *both* prior knowledge and hyperparameter(s). In contrast, the only assumption in the proposed method is that time-varying graphs are characterized by the multiresolution property, which is a natural assumption of signal processing. This implies the flexibility of the proposed method.

Figs. 5.8, 5.9 and 5.10 show the visualization of the temporal variation in the ground-truth graphs and the learned graphs with a window size of 40. The



(a) Edge-Markovian evolving graph (b) Switching behavior graph

FIGURE 5.8: Visualization of the temporal variations in the ground-truth time-varying graph of each dataset.

vertical and horizontal axes of these figures represent the edge and time slot indices of the time-varying graph, and the color represents the intensity of the edge weights. For simple visualization, the first 100 edge indices are visualized.

As can be seen in Figs. 5.9 and 5.10, SGL-S and SGL-LG lose the temporal relations, whereas TVGL-S, TVGL-LG, and TVGL-MR can capture the original structures more precisely than static methods. Time-varying graphs by TVGL-S, TVGL-LG, and TVGL-MR are similar, but the proposed method tends to yield larger edge weights.

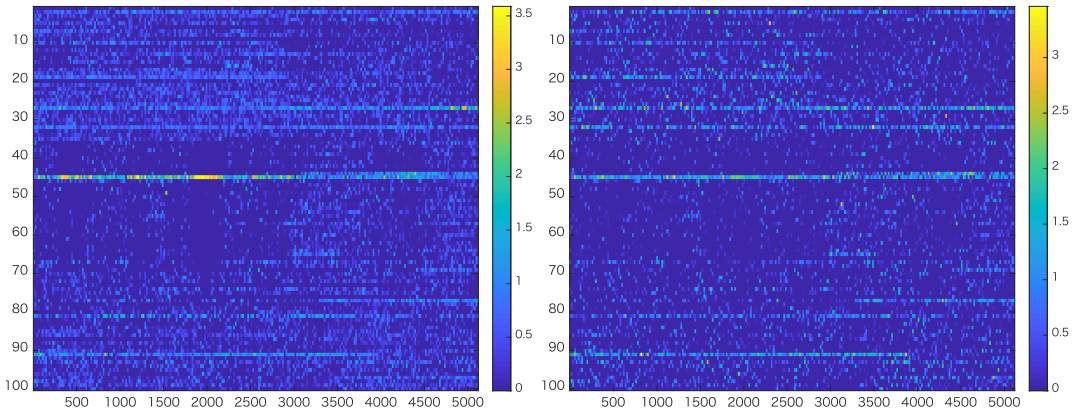
5.4.3 Learning Temporal Multiresolution Graphs From Real Temperature Data

Finally, we apply TVGL-MR to the real temperature data in Hokkaido, the northernmost island in Japan. The goal of this experiment is to explore the common (time-invariant) and seasonal relationships among geographical regions using the proposed method.

We use the average temperature data[2] measured at 172 recording locations in Hokkaido from March 2014 to February 2015. We perform TVGL-MR with $L = 3$ (i.e., the number of graphs is four at the highest level).

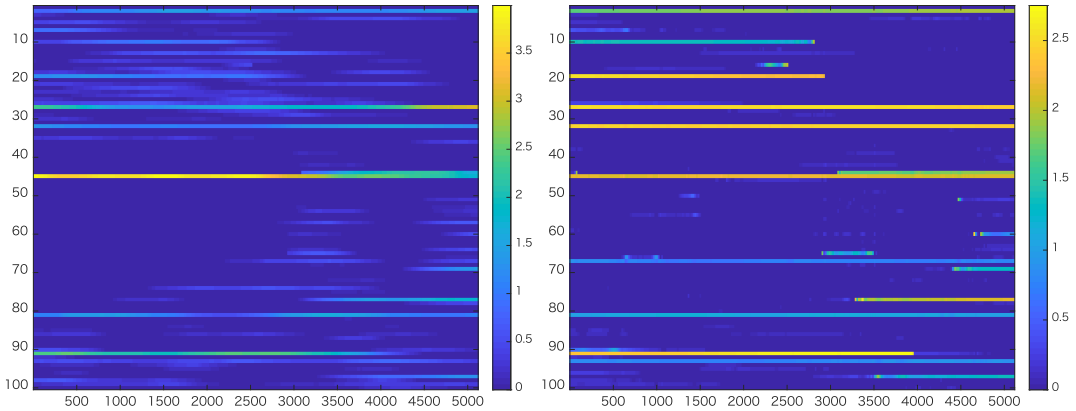
Fig. 5.11 shows the lowest resolution graph $\mathbf{W}_{0,0}$ obtained by TVGL-MR and the graph obtained by SGL-LG from data of all time slots. Note that both of them can be regarded as static graphs. Focusing on the graph learned by TVGL-MR, the following characteristics are observed:

[2]The Japan Meteorological Agency provided the daily temperature data from their website at https://www.jma.go.jp/jma/index.html



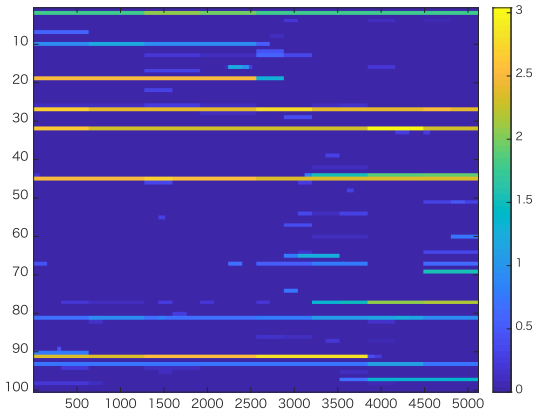
(a) SGL-S

(b) SGL-LG



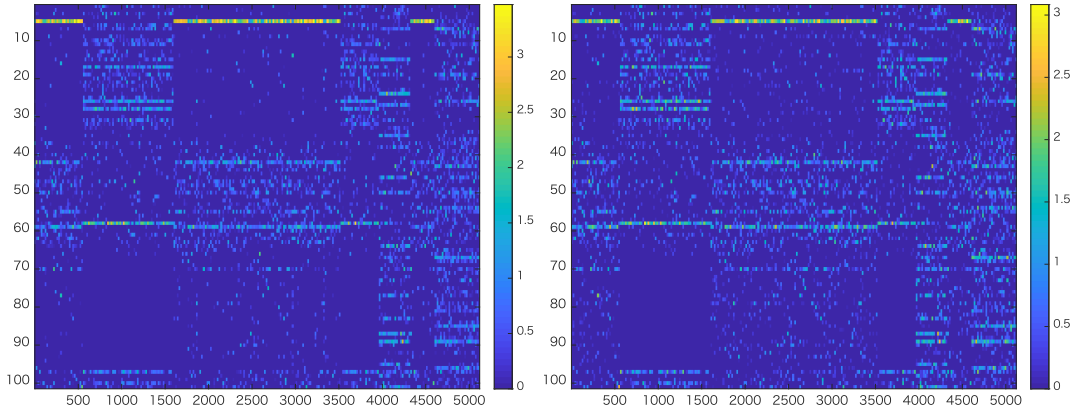
(c) TVGL-S

(d) TVGL-LG



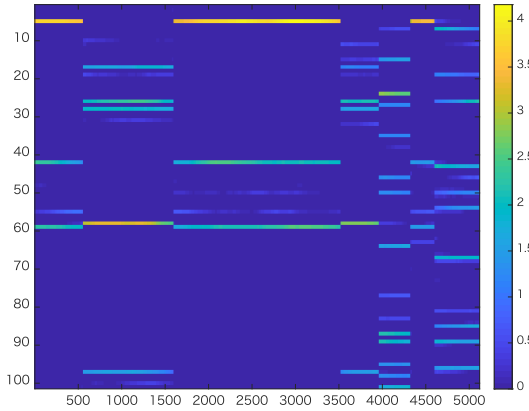
(e) TVGL-MR

FIGURE 5.9: Visualization of the temporal variations in the learned time-varying graph for EMEG dataset. The colors in these figures represent the weights of the edges.

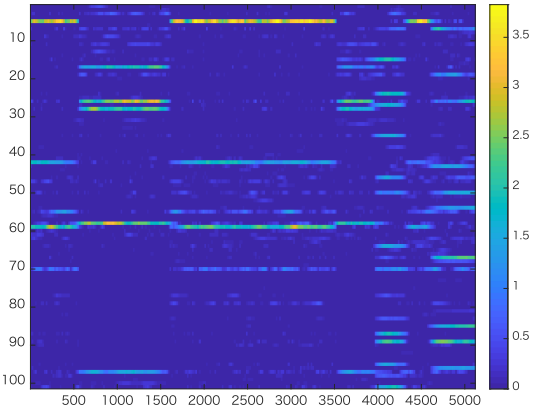


(a) SGL-S

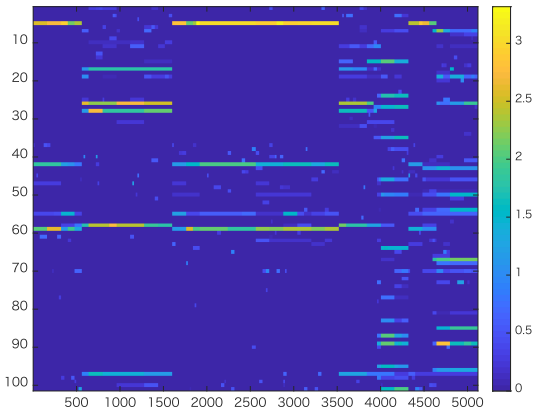
(b) SGL-LG



(c) TVGL-S



(d) TVGL-LG



(e) TVGL-MR

FIGURE 5.10: Visualization of the temporal variations in the learned time-varying graph for SBG dataset. The colors in these figures represent the weights of the edges.

- Vertices close to each other are basically connected, and edges between closer nodes tend to have large weights. However, if the recording locations are separated by a mountain (brown-ish area), nodes may not be connected even if they are geographically close.
- Vertices with similar geographic features are often connected, i.e., ones along the coast are connected to each other, and the similar characteristic is observed for inland vertices.

The above-mentioned characteristics seem reasonable because the relationship based on the distance between nodes or geographic features is static.

In contrast, the graph learned by SGL-LG is denser than that by TVGL-MR and includes many edges connecting distant nodes. Such edges may be derived from the seasonal behavior, which is described later. As SGL-LG learns a static graph from all the time slots without separating structures localized at various temporal resolutions, the learned graph may include both common and seasonal edges.

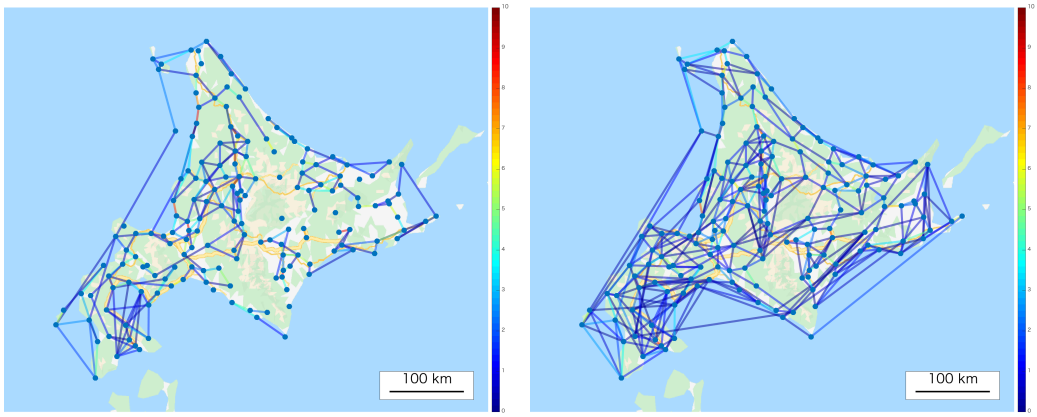
Fig. 5.12 shows $\mathbf{W}_{2,0}, \ldots, \mathbf{W}_{2,3}$ learned by TVGL-MR, which corresponds to season-specific graphs. In contrast to the static graph, these seasonal graphs have few edges connecting nodes close to each other. This suggests that the distance-based relationship would have a weak effect on the seasonal behavior. Furthermore, the summer- and winter-specific graphs have more edges than those of the spring and autumn-specific graphs. This seems intuitive because the seasonal effects in summer and winter are expected to be stronger than those in mild , such as spring and fall.

Furthermore, edges connecting distant coastal nodes in the summer and winter-specific graphs (which are also observed in SGL-LG in Fig. 5.11(b)) can be attributed to the effects of seasonal sea currents. Fig. 5.13 shows the sea surface temperature (SST) [3] on August 7, 2014, and January 8, 2015. As can be seen in Figs. 5.12(b), 5.12(d), and 5.13, vertices connected along coasts in the summer- and winter-specific graphs reflect SST behaviors for the two seasons.

5.5 Summary

This chapter demonstrated a temporal multiresolution graph learning method from multivariate time-series data. The proposed method is designed based on a signal generation model in accordance with an LGMRF, and enables the capture of time-varying structures having a multiresolution property in one single

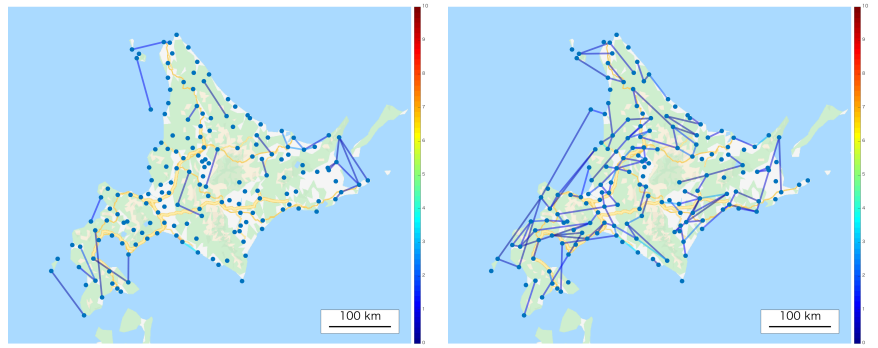
[3] The daily SST was provided by Japan Meteorological Agency, from their website at https://www.jma.go.jp/jma/index.html



(a) (b)

FIGURE 5.11: Visualization of learned graphs. (a) $\mathbf{W}_{0,0}$ learned by the TVGL-MR. (b) Graph learned by SGL-LG from data of all time slots.

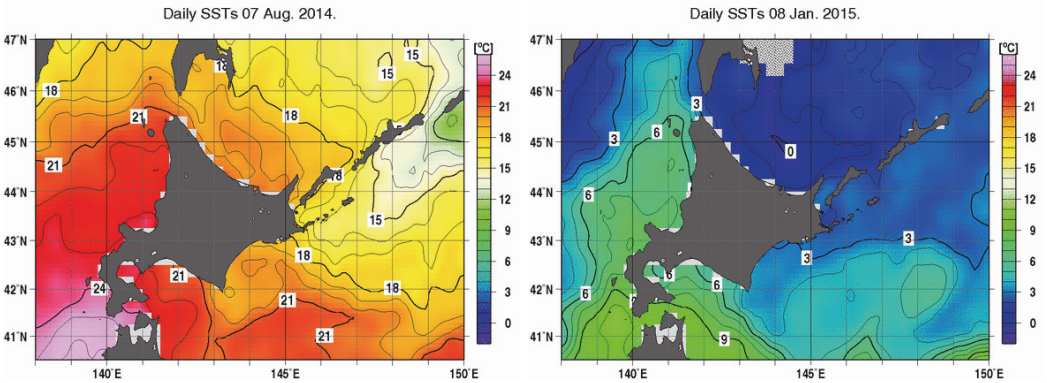
framework. The TVGL is formulated as a convex optimization problem and can be solved efficiently using a primal-dual splitting algorithm. The experiments on synthetic and real datasets demonstrate that the proposed method outperforms the existing static and time-varying graph learning methods.



(a) Spring-specific graph (b) Summer-specific graph

(c) Autumn-specific graph (d) Winter-specific graph

FIGURE 5.12: Visualization of the season-specific graphs learned by TVGL-MR: (a)–(b) corresponds to $\mathbf{W}_{2,0}, \ldots, \mathbf{W}_{2,3}$, respectively.



(a) (b)

FIGURE 5.13: Daily sea surface temperature. (a) August 7, 2014. (b) January 8, 2015.

Chapter 6

Graph Learning Information Criterion

Most studies on graph learning formulate their problem as a convex optimization that minimizes some criteria (e.g., signal smoothness on graphs) [37–39,45, 49,50]. Since their formulations generally contain regularization terms based on prior knowledge, we need to tune hyperparameters that control the strength of the regularization to obtain the desired graph. A typical example of the regularization used in graph learning is the ℓ_1 regularization, and its hyperparameter controls the edge sparsity of the learned graph. However, we do not know how sparse the actual graph is. As a result, it is difficult to determine the optimal hyperparameters in real applications and we often determine the parameters in an ad hoc manner.

The question we consider in this chapter is: *How should we choose the optimal hyperparameter(s) for graph learning?* Although the choice of hyperparameters strongly affects the performance of graph learning, the research on hyperparameter selection still remains insufficient. When graph learning is applied for supervised learning problems such as prediction and classification, the hyperparameters could be determined by cross-validation (but they do not have a theoretical guarantee in general). In contrast, for unsupervised learning such as clustering, there is few methods to determine hyperparameters.

Some studies of graph learning determine hyperparameters using Bayesian information criterion (BIC) [85,86]. BIC is a criterion based on *model evidence* and is used for model selection among a finite set of models, i.e., a set of graphs learned under different parameters. However, BIC is only a rough approximation of model evidence and fails to choose a good model in graph learning (we later show it in the experiment in Section 6.3).

In this chapter, we propose a new model selection criterion specifically designed for graph learning: *Graph learning information criterion* (GLIC). The strategy of our method is to introduce the Bayesian model equivalent to the

formulation based on Laplacian constrained Gaussian Markov random field (LGMRF) model [39] and to compute the approximation of the model evidence of this Bayesian model using the result in [87]. Since this approximation computation requires sampling from the posterior distribution of the Bayesian model, we also present an efficient sampling method using block Gibbs sampling scheme.

6.1 Graph Learning and Bayesian Information Criterion

6.1.1 Graph Learning with LGMRF

In this chapter, we assume the following signal observation model based on Laplacian constrained Gaussian Markov random field (LGMRF) [38, 39, 45]:

$$p(\mathbf{x} \mid \mathbf{\Theta}) = \frac{1}{(2\pi)^{K/2} \left(\operatorname{gdet}(\mathbf{\Theta}^{\dagger})\right)^{1/2}} \exp\left(-\frac{1}{2}\mathbf{x}^{T}\mathbf{\Theta}\mathbf{x}\right), \tag{6.1}$$

where $\mathbf{\Theta} \in \mathcal{L}$ is the precision matrix satisfying graph Laplacian constraints. The negative log-likelihood function $L(\mathbf{\Theta})$ of (6.1) is given by

$$\begin{aligned} L(\mathbf{\Theta}) &= -\log(\prod_{k=1}^{K} p\left(\mathbf{x}_k \mid \mathbf{\Theta}\right)) \\ &= \frac{1}{2}\sum_{k=1}^{K} \operatorname{Tr}\left(\mathbf{x}_k^{\intercal}\mathbf{\Theta}\mathbf{x}_k\right) - \frac{K}{2}\log \operatorname{gdet}(\mathbf{\Theta}). \end{aligned} \tag{6.2}$$

Furthermore, suppose that the prior distribution $p(\mathbf{\Theta})$ is the following Laplace distribution.

$$p(\mathbf{\Theta} \mid \lambda) = \prod_{i<j} p(\theta_{ij}) = \prod_{i<j} \frac{\lambda}{2} \exp(-\lambda|\theta_{ij}|), \tag{6.3}$$

where λ is a scale parameter of the Laplace distribution. The maximum a posteriori (MAP) estimation of $\mathbf{\Theta}$ with (6.2) and (6.3) leads to the following the optimization problem [39]:

$$\underset{\mathbf{\Theta}\in\mathcal{L}}{\text{minimize}} \ \frac{1}{K}\operatorname{Tr}(\mathbf{\Theta}\mathbf{S}) - \log \operatorname{gdet}(\mathbf{\Theta}) + \alpha\|\mathbf{\Theta}\|_{1,\text{off}} \tag{6.4}$$

where $\mathbf{S} = \mathbf{X}\mathbf{X}^\mathsf{T}$, $\alpha = \lambda/K$, $\|\mathbf{\Theta}\|_{1,\text{off}}$ represents the absolute sum of off-diagonal elements in $\mathbf{\Theta}$, and $\mathcal{L}$ is the set of valid graph Laplacians given by:

$$\mathcal{L} = \left\{ \mathbf{L} \in \mathbb{R}^{N\times N} : L_{ij} = L_{ji} \leq 0 \ (i \neq j), L_{ii} = \sum_{i\neq j} L_{ij} \right\}. \tag{6.5}$$

The optimization problem in (6.4) can be solved using the block coordinate descent algorithm [39]. In (6.4), the hyperparameter α controls the sparsity of the learned graph Laplacian $\mathbf{\Theta}$. If we do not have prior information on the graph sparsity, selecting the optimal α becomes a difficult problem.

6.1.2 Bayesian Information Criterion

In Bayesian statistics, *model evidence* is often used for model selection among a finite set of models [88]. The model evidence is a likelihood function in which some parameters are marginalized.

The model evidence of the graph learning model introduced in Section 6.1.1 is given by

$$p(\mathbf{X} \mid \lambda) = \int_{\mathbf{\Theta}} p(\mathbf{X} \mid \mathbf{\Theta}) p(\mathbf{\Theta} \mid \lambda) \mathrm{d}\mathbf{\Theta}. \tag{6.6}$$

Unfortunately, (6.6) cannot be analytically computed. Hence, the approximate computation of the model evidence is required.

Bayesian information criterion (BIC) is a criterion for model selection based on the model evidence [86]. BIC is an approximation of the negative logarithmic model evidence, which is computed using the Laplace method. It is defined as follows:

$$\mathrm{BIC} = L(\mathbf{\Theta}) + \frac{d}{2} \log K, \tag{6.7}$$

where $L(\cdot)$ is the negative log-likelihood function of the estimated parameter, d is the number of estimated parameters in the model, and K is the number of observations.

The BIC of the graph learning in (6.4) can be computed as follows:

$$\mathrm{BIC}(\alpha) = \frac{1}{2}(\mathrm{Tr}(\widehat{\mathbf{\Theta}}\mathbf{S}) - K \log \mathrm{gdet}(\widehat{\mathbf{\Theta}}) + \|\widehat{\mathbf{\Theta}}\|_0 \log K), \tag{6.8}$$

where $\widehat{\mathbf{\Theta}}$ is the solution of (6.4) with the parameter α and $\|\widehat{\mathbf{\Theta}}\|_0$ is the number of nonzero elements in $\widehat{\mathbf{\Theta}}$.

When selecting a model among a finite model set, the ones with small BIC are preferred. Thus, the hyperparameter α of (6.4) can be determined by the

following steps: 1) Estimating $\mathbf{\Theta}$ with different α's; 2) Computing BIC(α) of the estimated $\mathbf{\Theta}$; and 3) Selecting the one with the smallest BIC.

However, BIC is only a rough approximation of model evidence [87] and often fails to choose a good model for graph learning. We will show it later in the experiment of Section 6.3.

6.2 GLIC

In this section, we present a computation method of GLIC, which is a new criterion for the model selection of graph learning. The strategy of the proposed method is to estimate the negative logarithmic model evidence in a different way from BIC.

For GLIC, we use an approximation of the negative logarithmic model evidence based on WBIC [87]:

$$-\log p(\mathbf{X} \mid \lambda) \approx \frac{\int L(\mathbf{\Theta}) \prod_{k=1}^{K} p\left(\mathbf{x}_k \mid \mathbf{\Theta}\right)^{\eta} p(\mathbf{\Theta}) \mathrm{d}\mathbf{\Theta}}{\int \prod_{k=1}^{K} p\left(\mathbf{x}_k \mid \mathbf{\Theta}\right)^{\eta} p(\mathbf{\Theta}) \mathrm{d}\mathbf{\Theta}}, \tag{6.9}$$

where $\eta = 1/\log(K)$. This indicates that the negative logarithmic model evidence can be estimated by the expectation of the negative log-likelihood over the posterior distribution $p(\mathbf{\Theta} \mid \mathbf{X}, \lambda, \eta)$ using Markov chain Monte Carlo (MCMC) method. Hence, (6.9) is rewritten as follows:

$$\begin{aligned} -\log p(\mathbf{X} \mid \lambda) &\approx \frac{1}{M} \sum_{m=1}^{M} L(\mathbf{\Theta}_m), \;\; \mathbf{\Theta}_m \sim p(\mathbf{\Theta} \mid \mathbf{X}, \lambda, \eta), \\ p(\mathbf{\Theta} \mid \mathbf{X}, \lambda, \eta) &= \frac{\prod_{k=1}^{K} p\left(\mathbf{x}_k \mid \mathbf{\Theta}\right)^{\eta} p(\mathbf{\Theta})}{\int \prod_{k=1}^{K} p\left(\mathbf{x}_k \mid \mathbf{\Theta}\right)^{\eta} p(\mathbf{\Theta}) \mathrm{d}\mathbf{\Theta}}. \end{aligned} \tag{6.10}$$

To compute the approximation of $-\log p(\mathbf{X} \mid \lambda)$ efficiently, we need to sample the the posterior distribution $p(\mathbf{\Theta} \mid \mathbf{X}, \lambda, \eta)$. In the following, we present an effective sampler of the posterior distribution.

6.2.1 Block Gibbs Sampler for GLIC

It is generally difficult to directly sample from the posterior distribution under the graph Laplacian constraint. Hence, we generate pseudo-samples $\{\mathbf{L}_m\}_{m=1}^{M}$ ($\mathbf{L}_m \in \mathcal{L}$) from $p(\mathbf{\Theta} \mid \mathbf{X}, \lambda, \eta)$ by the following steps:

1. Sampling $\{\mathbf{\Theta}_m\}_{m=1}^{M}$ from a posterior distribution using Gaussian Markov random field *without* the Laplacian constraint (i.e., $\mathrm{N}(0, \mathbf{\Theta}^{-1})$).

2. Constructing the graph Laplacian $\mathbf{L}_m$ closest to $\mathbf{\Theta}_m$.

Here, we describe the sampling method in Step 1 using block Gibbs sampling. First, the Laplace distribution in (6.3) is rewritten by scale mixture of Gaussians [89, 90] to derive a hierarchical representation to which block Gibbs sampling can be applied:

$$\frac{\lambda}{2}e^{-\lambda|\theta|} = \int_0^\infty \frac{1}{\sqrt{2\pi\tau}} e^{-\theta^2/(2\tau)} \frac{\lambda^2}{2} e^{-\lambda^2\tau/2} \mathrm{d}\tau, \quad \lambda > 0, \tag{6.11}$$

where τ_{ij} is a latent scale parameter. From $p(\mathbf{x} \mid \mathbf{\Theta}) = \mathrm{N}(0, \mathbf{\Theta}^{-1})$ and (6.11), it can be derived that $p(\mathbf{\Theta}, \boldsymbol{\tau} \mid \mathbf{X}, \lambda, \eta)$ is proportional to the following quantity:

$$\begin{aligned} p(\mathbf{\Theta}, \boldsymbol{\tau} \mid \mathbf{X}, \lambda, \eta) \propto & |\mathbf{\Theta}|^{\frac{\eta K}{2}} \exp\left\{-\eta \operatorname{Tr}\left(\frac{1}{2}\mathbf{S}\mathbf{\Theta}\right)\right\} \\ & \times \prod_{i<j} \left\{\tau_{ij}^{-\frac{1}{2}} \exp\left(-\frac{\theta_{ij}^2}{2\tau_{ij}}\right) \exp\left(-\frac{\lambda^2}{2}\tau_{ij}\right)\right\}. \end{aligned} \tag{6.12}$$

Let $\mathbf{T} \in \mathbb{R}^{N\times N}([\mathbf{T}]_{ij} = \tau_{ij})$ be a symmetric matrix whose all diagonal elements are zero. Then, the block matrix representations of $\mathbf{\Theta}$, $\mathbf{S}$, and $\mathbf{T}$ are respectively denoted by

$$\mathbf{\Theta} = \begin{bmatrix} \mathbf{\Theta}_{11} & \boldsymbol{\theta}_{12} \\ \boldsymbol{\theta}_{12}^\mathsf{T} & \theta_{22} \end{bmatrix}, \ \mathbf{S} = \begin{bmatrix} \mathbf{S}_{11}, \mathbf{s}_{12} \\ \mathbf{s}_{12}^\mathsf{T}, s_{22} \end{bmatrix}, \ \mathbf{T} = \begin{bmatrix} \mathbf{T}_{11}, \boldsymbol{\tau}_{12} \\ \boldsymbol{\tau}_{12}^\mathsf{T}, 0 \end{bmatrix}. \tag{6.13}$$

The block Gibbs sampler iterates sampling of $\boldsymbol{\theta}_{12}$, θ_{22} and τ_{ij} from their conditional posterior distributions to obtain samples $\{\mathbf{\Theta}_m\}_{m=1}^M$. From (6.12), (6.13), and the following Schur complement lemma [91]

$$\det\left(\begin{bmatrix} \mathbf{\Theta}_{11} & \boldsymbol{\theta}_{12} \\ \boldsymbol{\theta}_{12}^\mathsf{T} & \theta_{22} \end{bmatrix}\right) = \det(\mathbf{\Theta}_{11}) \det\left(\theta_{22} - \boldsymbol{\theta}_{12}^\mathsf{T}\mathbf{\Theta}_{11}^{-1}\boldsymbol{\theta}_{12}\right), \tag{6.14}$$

the conditional posterior of $\boldsymbol{\theta}_{12}$ and θ_{22} are given by

$$\begin{aligned} & p\left(\boldsymbol{\theta}_{12}, \theta_{22} \mid \mathbf{\Theta}_{11}, \mathbf{T}, \mathbf{X}, \lambda, \eta\right) \\ \propto & \left(\theta_{22} - \boldsymbol{\theta}_{12}^\mathsf{T}\mathbf{\Theta}_{11}^{-1}\boldsymbol{\theta}_{12}\right)^{\frac{\eta K}{2}} \\ & \times \exp\left[-\frac{1}{2}\left\{\boldsymbol{\theta}_{12}^\mathsf{T}\mathbf{D}_{\boldsymbol{\tau}}^{-1}\boldsymbol{\theta}_{12} + 2\eta\mathbf{s}_{12}^\mathsf{T}\boldsymbol{\theta}_{12} + (\eta s_{22} + \lambda)\,\theta_{22}\right\}\right], \end{aligned} \tag{6.15}$$

where $\mathbf{D}_{\boldsymbol{\tau}} = \mathrm{diag}(\boldsymbol{\tau}_{12})$. Making a change of variables as $\boldsymbol{\beta} := \boldsymbol{\theta}_{12}$ and $\gamma := \theta_{22} - \boldsymbol{\theta}_{12}^{\mathsf{T}}\boldsymbol{\Theta}_{11}^{-1}\boldsymbol{\theta}_{12}$, (6.15) is reduced to the following form:

$$\begin{aligned} & p\left(\boldsymbol{\beta}, \gamma \mid \boldsymbol{\Theta}_{11}, \mathbf{T}, \mathbf{X}, \lambda, \eta\right) \\ & \propto \gamma^{\frac{\eta K}{2}} \exp\left(-\frac{\eta s_{22} + \lambda}{2}\gamma\right) \\ & \times \exp\left(-\frac{1}{2}\left[\boldsymbol{\beta}^{\mathsf{T}}\left\{\mathbf{D}_{\boldsymbol{\tau}}^{-1} + \left(\eta s_{22} + \lambda\right)\boldsymbol{\Theta}_{11}^{-1}\right\}\boldsymbol{\beta} + 2\eta \mathbf{s}_{12}^{\mathsf{T}}\boldsymbol{\beta}\right]\right). \end{aligned} \tag{6.16}$$

This indicates that γ and $\boldsymbol{\beta}$ can be sampled from the following gamma and multivariate Gaussian distributions:

$$\begin{aligned} \gamma \mid \left(\boldsymbol{\Theta}_{11}, \mathbf{T}, \mathbf{X}, \lambda, \eta\right) &\sim \mathrm{Gam}\left(\frac{\eta K}{2} + 1, \frac{s_{22} + \lambda}{2}\right), \\ \boldsymbol{\beta} \mid \left(\boldsymbol{\Theta}_{11}, \mathbf{T}, \mathbf{X}, \lambda, \eta\right) &\sim \mathrm{N}\left(-\eta \mathbf{C}\mathbf{s}_{12}, \mathbf{C}\right), \end{aligned} \tag{6.17}$$

where $\mathbf{C} = ((\eta s_{22} + \lambda)\boldsymbol{\Theta}_{11} + \mathbf{D}_{\boldsymbol{\tau}})$. Focusing on τ_{ij} in (6.12), the conditonal distribution of $u_{ij} = 1/\tau_{ij}$ is given by

$$u_{ij} \mid (\theta_{ij}, \lambda) \sim \mathrm{IGau}(\sqrt{(\lambda^2/\theta_{ij}^2)}, \lambda^2). \tag{6.18}$$

As a result, the block Gibbs sampler procedure of (6.17) and (6.18) allows us to sample $\{\boldsymbol{\Theta}_m\}_{m=1}^M$ from the posterior distribution (6.12).

In Step 2, the sampled $\boldsymbol{\Theta}_m$ is replaced by the graph Laplacian $\mathbf{L}_m$ closest to $\boldsymbol{\Theta}_m$. This graph Laplacian $\mathbf{L}_m$ is obtained by solving the problem:

$$\underset{\mathbf{L}_m \in \mathcal{L}}{\text{minimize}} \ \|\boldsymbol{\Theta}_m - \mathbf{L}_m\|_1. \tag{6.19}$$

This problem can be easily solved using the result in [92].

6.2.2 Computation of GLIC

Let $\{\mathbf{L}_m\}_{m=1}^M$ be samples from the block Gibbs sampler presented in Section 6.2.1. Based on (6.10), we define the GLIC as follows:

$$\mathrm{GLIC} = \frac{1}{2M}\left(\sum_{m=1}^{M} \mathrm{Tr}(\mathbf{L}_m\mathbf{S}) - K \log \mathrm{gdet}(\mathbf{L}_m)\right). \tag{6.20}$$

Although GLIC seems similar to (6.8), it is computed based on WBIC and MCMC and can yield a more reliable model evidence approximation.

TABLE 6.1: Average performance of graph learning under hyperparameters selected by BIC and GLIC.

	Dataset	Relative error	F-measure
BIC	ER	0.867	0.518
	RM	0.968	0.231
GLIC	ER	**0.541**	**0.546**
	RM	**0.451**	**0.572**

The algorithm of the proposed method is summarized in Algorithm 5. For the hyperparameter selection in graph learning, the hyperparameter having the smallest GLIC is selected as an appropriate hyperparameter.

Algorithm 5 GLIC

Input: α, $\mathbf{S}$, M
Output: GLIC,
  Initialize $\boldsymbol{\Theta} = \mathbf{I}$
  for $m = 0$ to M **do**
    for $i = 0$ to N **do**
      Sample γ and $\boldsymbol{\beta}$ from (6.17)
      Update $\boldsymbol{\theta}_{12} = \boldsymbol{\beta}$ and $\theta_{22} = \gamma + \boldsymbol{\theta}_{12}^{\mathsf{T}} \boldsymbol{\Theta}_{11}^{-1} \boldsymbol{\theta}_{12}$
      Rearrange row/columns of $\boldsymbol{\Theta}$, $\mathbf{S}$, and $\mathbf{T}$
    end for
    for $i < j$ **do**
      Sample u_{ij} from (6.18)
      Update $\tau_{ij} = 1/u_{ij}$
    end for
    Obtain $\mathbf{L}_m$ by solving (6.19)
  end for
  Compute GLIC in (6.20)

6.3 Experiments

In this section, we conduct graph learning experiments using random graphs to validate the efficiency of the proposed method.

6.3.1 Dataset and Setup

We construct datasets by the following two steps: 1) Constructing random graphs and 2) generating signals from the constructed graph. In this experiment, we use two types of random graphs:

- Erdős–Rényi (ER) graph $\mathcal{G}_{\mathrm{ER}}^{(p)}$ where p is the edge connection probability.

- Random modular (RM) graph $\mathcal{G}_{\text{RM}}^{(p_1,p_2)}$ (also known as graphs with stochastic block model), where p_1 and p_2 are intra-cluster and inter-cluster edge connection probabilities, respectively.

All graphs have $N = 36$, and the edge weights are selected randomly from the uniform distribution $\text{U}(0.1, 3)$. We construct 30 graphs for each graph model: p of ER graph is chosen from $\text{U}(0.08, 0.12)$; p_1 and p_2 of RM graph are chosen from $\text{U}(0.25, 0.3)$ and $\text{U}(0.08, 0.12)$, respectively. Based on the constructed graph, we generate 100 graph signals from LGMRF in (6.1). For both datasets, graphs are learned with different α and compute BIC and GLIC.

The performance of graph learning is evaluated by F-measure and relative error. The F-measure, which is the harmonic average of the precision and recall, represents the accuracy of the estimated graph structure. Relative error is given by:

$$\text{RE}\left(\widehat{\boldsymbol{\Theta}}, \boldsymbol{\Theta}^*\right) = \frac{\left\|\widehat{\boldsymbol{\Theta}} - \boldsymbol{\Theta}^*\right\|_F}{\left\|\boldsymbol{\Theta}^*\right\|_F} \tag{6.21}$$

where $\widehat{\boldsymbol{\Theta}}$ is the estimated graph Laplacian, $\boldsymbol{\Theta}^*$ is the ground truth, and $\|\cdot\|_F$ is the Frobenius norm.

6.3.2 Results

Table 6.1 summarizes the average performance under the hyperparameters selected by BIC and GLIC. For both graphs, the graph selected with GLIC significantly outperforms that with BIC for the F-measure and relative error.

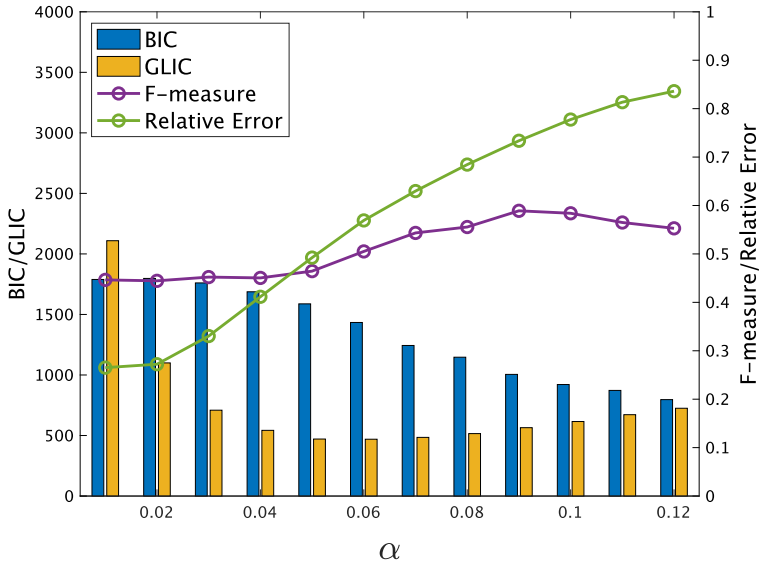
Fig. 6.1 shows the BIC and GLIC with different α and the objective performances. Let us denote $\hat{\alpha}$ as the α having the smallest information criterion. For the ER graph, the optimal $\hat{\alpha}$ with BIC is 0.12, and that with GLIC is 0.06; for the RM graph, $\hat{\alpha}$ with BIC is 0.12, and that with GLIC is 0.03. As observed from the figure, GLIC selects the parameter that indicates a good trade-off between F-measure and relative error, while BIC fails to do so.

Fig. 6.2 shows the visualization of the learned graph Laplacian with hyperparameters selected by BIC and GLIC. The graph selected by BIC is too sparse compared to the ground truth. In contrast, GLIC can select graphs with almost the same edge density as that of the ground truth.

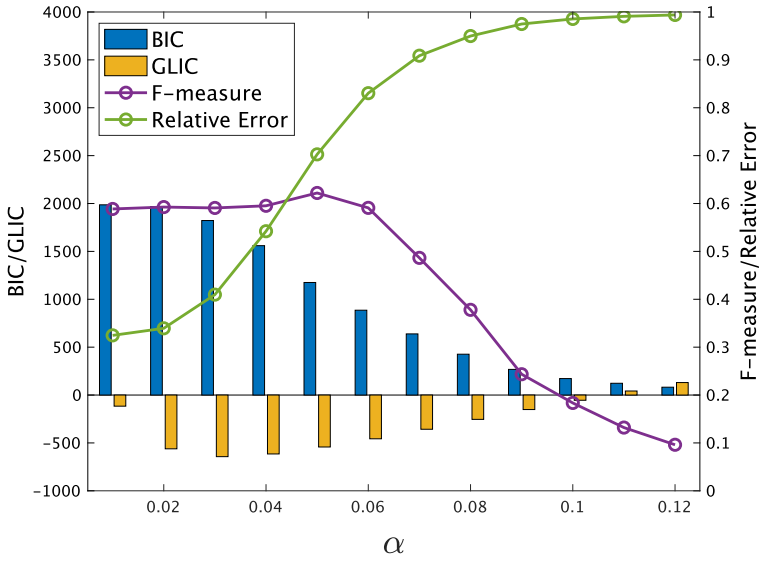
6.4 Summary

In this chapter, we propose a model selection criterion for graph learning using the approximation of the model evidence. It is based on WBIC and is

computed with an efficient sampler of the posterior distribution of the graph learning model based on LGMRF. The experimental results demonstrated that the proposed method can successfully select a valid parameter set among candidates.

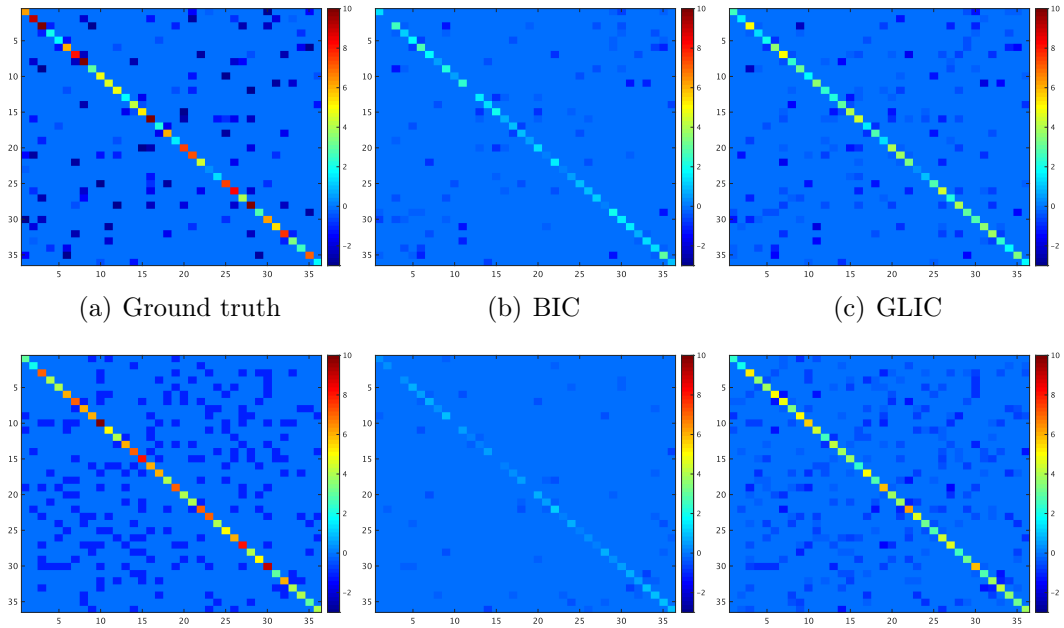


(a) ER graph



(b) RM graph

FIGURE 6.1: BIC and GLC with different α. (a) BIC: $\hat{\alpha} = 0.12$, GLIC: $\hat{\alpha} = 0.06$. (b) BIC: $\hat{\alpha} = 0.12$, GLIC: $\hat{\alpha} = 0.03$.



(a) Ground truth (b) BIC (c) GLIC

(d) Ground truth (e) BIC (f) GLIC

FIGURE 6.2: Visualization of the learned graph Laplacian with hyperparameters selected by BIC and GLIC. The top rows and bottom rows depict the learned graphs from ER graph dataset and RM graph dataset, respectively.

Chapter 7

Conclusion

This dissertation addressed the problem of learning time-varying graphs and the model selection problem that are a hot research topic in graph signal processing and a fundamental tool to utilize graph signal processing. Time-varying graph learning requires 1) estimating time-varying graphs from a small number of data and 2) taking into account the temporal relationships of the underlying time-varying graphs. To fulfill these requirements, the following methods were proposed:

1. **Time-varying graph learning with constraints on graph temporal variation (Chapter 4 and [49, 50])**
   A generic framework for learning time-varying graphs was proposed. This method imposes constraints for temporal variations of graphs between neighboring time slots based on prior knowledge of underlying time-varying graphs. Specifically, the framework introduces the fused Lasso regularization and group Lasso regularization of the temporal variation to capture graphs with temporal homogeneity and switching behavior. These constraints enable us to learn time-varying graphs successfully from small data samples and to consider the temporal relationship of time-varying graphs. This framework is formulated as convex optimization, and an efficient algorithm for solving the problem was presented.

2. **Temporal multiresolution graph learning (Chapter 5 and [51,52])**
   Temporal multiresolution graph learning was proposed. This method directly learns graphs localized at different temporal resolutions (which are called temporal multiresolution graphs), and reconstructs time-varying graphs by the linear combination of the temporal multiresolution graphs. Temporal multiresolution graphs are learned by the proposed algorithm based on a primal-dual splitting algorithm. The experiments using synthetic and real data demonstrated that the proposed method can learn efficiently temporal multiresolution graphs from data.

This dissertation also presented an answer to the question: *How should we choose the optimal hyperparameter(s) for graph learning?*

3. **Graph learning information criterion (Chapter 6)**
   Graph learning information criterion (GLIC), which is a tool to determine hyperparameter of graph learning, was proposed. GLIC selects optimal hyperparameters based on model evidence, the model selection criterion used in Bayesian statistics. GLIC computes the approximation of the model evidence with an efficient Gibbs sampler of the posterior distribution of the graph learning model based on LGMRF. The experimental results demonstrated that GLIC can successfully select a valid parameter.

Each of them or their combinations extend the scope of the application of graph signal processing. We expect that our work will be applied in a wide range of fields.

Appendix A

Algorithm for Time-varying Graph Learning with LGMRF

The TVGL problem (5.10) can be solved using a primal-dual splitting (PDS) algorithm while it cannot be directly applied to (5.10). Here, we reformulate (5.10) to the applicable form of the PDS algorithm. Let $\mathbf{Z}^{(k)} \in \mathbb{R}^{N\times N}$ be a pairwise distance matrix defined by

$$[\mathbf{Z}^{(k)}]_{i,j} = \sum_{n=0}^{r-1} \|[\mathbf{X}^{(k)}]_{i,n} - [\mathbf{X}^{(k)}]_{j,n}\|^2. \tag{A.1}$$

We also denote the vector form representation of $\mathbf{Z}^{(k)}$ as $\mathbf{z}^{(k)} \in \mathbb{R}^E$, which corresponds to the lower-triangular part of $\mathbf{Z}^{(k)}$. We further define the following two vectors:

$$\begin{aligned} \mathbf{w}_{\mathrm{all}} &= [\mathbf{w}^{(0)\mathsf{T}}, \ldots, \mathbf{w}^{(K-1)\mathsf{T}}]^{\mathsf{T}} \in \mathbb{R}^{ET}, \\ \mathbf{z}_{\mathrm{all}} &= [\mathbf{z}^{(0)\mathsf{T}}, \ldots, \mathbf{z}^{(K-1)\mathsf{T}}]^{\mathsf{T}} \in \mathbb{R}^{ET}. \end{aligned} \tag{A.2}$$

Then, the second term of (5.10) can be rewritten as

$$\sum_{k=0}^{K-1} \mathrm{tr}((\mathbf{X}^{(k)})^{\mathsf{T}} \mathcal{L}\mathbf{w}^{(k)} \mathbf{X}^{(k)}) = \sum_{k=0}^{K-1} (\mathbf{z}^{(k)})^{\mathsf{T}} \mathbf{w}^{(k)} = \mathbf{z}_{\mathrm{all}}^{\mathsf{T}} \mathbf{w}_{\mathrm{all}}. \tag{A.3}$$

Hereafter, we define $\hat{\mathcal{L}}^{(k)}\mathbf{w}_{\mathrm{all}} = \mathcal{L}(\mathbf{w}^{(k)})$ for notation simplicity. By introducing an indicator function ι, (5.10) can be rewritten as

$$\min_{\mathbf{w}_{\mathrm{all}}} \ \frac{1}{r}\mathbf{z}_{\mathrm{all}}^{\mathsf{T}}\mathbf{w}_{\mathrm{all}} + \alpha\|\mathbf{w}_{\mathrm{all}}\|_1 + \beta\psi(\mathbf{\Phi}\mathbf{w}_{\mathrm{all}}) + \iota(\mathbf{w}_{\mathrm{all}}) - \sum_{k=0}^{K-1} \log \mathrm{gdet}(\hat{\mathcal{L}}^{(k)}(\mathbf{w}_{\mathrm{all}})) \tag{A.4}$$

where $\mathbf{\Phi}$ is a linear operator that satisfies $\mathbf{\Phi}\mathbf{w}_{\text{all}} = \mathbf{w}_{\text{all}} - \hat{\mathbf{w}}_{\text{all}}$ in which $\hat{\mathbf{w}}_{\text{all}} = [\mathbf{w}^{(0)\mathsf{T}}, \mathbf{w}^{(0)\mathsf{T}}, \mathbf{w}^{(1)\mathsf{T}}, \ldots, \mathbf{w}^{(K-2)\mathsf{T}}]^{\mathsf{T}}$, and ι is defined by:

$$\iota(\mathbf{w}) = \begin{cases} 0 & \mathbf{w} \geq 0 \\ \infty & \text{otherwise.} \end{cases} \tag{A.5}$$

By introducing a linear operator $\hat{\mathcal{L}} : \mathbb{R}^{E \times 2^L} \to \mathbb{R}^{2^L N \times N}$ defined as

$$\hat{\mathcal{L}}(\mathbf{w}_{\text{all}}) := [\hat{\mathcal{L}}^{(0)}(\mathbf{w}_{\text{all}}), \ldots, \hat{\mathcal{L}}^{(K-1)}(\mathbf{w}_{\text{all}})]^{\mathsf{T}}, \tag{A.6}$$

dual variables $\mathbf{p} := \mathbf{\Phi}\mathbf{w}_{\text{all}}$, and $\mathbf{V} := [\mathbf{V}_0^{\mathsf{T}}, \ldots, \mathbf{V}_{K-1}^{\mathsf{T}}]^{\mathsf{T}} = \hat{\mathcal{L}}(\mathbf{w}_{\text{all}})$, we can convert (A.4) into the form of (2.35) as follows:

$$\begin{aligned} f(\mathbf{w}_{\text{all}}) &= 0, \\ g(\mathbf{w}_{\text{all}}) &= \frac{1}{r}(\alpha\mathbf{1} + \mathbf{z}_{\text{all}})^{\mathsf{T}}\mathbf{w}_{\text{all}} + \iota(\mathbf{w}_{\text{all}}), \\ h_1(\mathbf{p}) &= \beta\psi(\mathbf{p}), \\ h_2(\mathbf{V}) &= -\sum_{k=0}^{K-1} \log \operatorname{gdet}(\mathbf{V}_k). \end{aligned} \tag{A.7}$$

We then consider the proximal operators of g, h_1 and h_2 in (A.4). The proximal operator of g corresponds to that of the weighted ℓ_1 norm with the non-negative constraint. This proximal operator is known to be computed as follows [64]:

$$\left[\operatorname{prox}_{\gamma g}(\mathbf{x})\right]_{i,j} = \begin{cases} 0 & [\mathbf{x}]_{i,j} \leq \frac{\gamma}{r}[\alpha\mathbf{1} + \mathbf{z}_{\text{all}}]_{i,j} \\ [\mathbf{x}]_{i,j} - \frac{\gamma}{r}[\alpha\mathbf{1} + \mathbf{z}_{\text{all}}]_{i,j} & \text{otherwise.} \end{cases} \tag{A.8}$$

Typical examples of the regularizer in h_1 are $\psi(\cdot) = \|\cdot\|_2^2$ and $\psi(\cdot) = \|\cdot\|_1$. It is well known that their proximal operator can be computed efficiently [64].

In general, the logarithm of generalized determinant in h_2 is non-convex. Fortunately, under the assumption that the learned graph is connected (and it is often the case), it can be replaced with a following convex function [39, Proposition 1]:

$$\log \operatorname{gdet}(\mathbf{A}) = \log \det(\mathbf{A} + \frac{1}{N}\mathbf{1}\mathbf{1}^{\mathsf{T}}). \tag{A.9}$$

Its proximal operator is given by

$$\mathrm{prox}_{\gamma(-\log \mathrm{gdet}(\cdot))}(\mathbf{A}) = \mathbf{U} \begin{bmatrix} \phi(\lambda_0) & & 0 \\ & \ddots & \\ 0 & & \phi(\lambda_{N-1}) \end{bmatrix} \mathbf{U}^\mathsf{T}, \tag{A.10}$$

where $\phi(\lambda_i) = \frac{\lambda_i + \sqrt{\lambda_i^2 + 4\gamma}}{2}$, $\mathbf{U}$ and λ_i are the eigenvector matrix and the eigenvalue of $\mathbf{A} + \frac{1}{N}\mathbf{1}\mathbf{1}^\mathsf{T}$, respectively.

Even if the original graph has disconnected components, we can avoid the problem of the calculation of the proximal operator by adding a small regularizing parameter c to the input as follows [58]:

$$\log \mathrm{gdet}(\mathbf{A}) \approx \log \det(\mathbf{A} + c^2\mathbf{I}), \tag{A.11}$$

Its proximal operator can also be computed with the same manner as (A.10).

Finally, We summarize the entire iterative algorithm in Algorithm 6. The convergence of this algorithm is guaranteed by choosing step sizes γ_1 and γ_2 so that they satisfy

$$\gamma_1\gamma_2\|\mathbf{\Phi}^\mathsf{T}\mathbf{\Phi} + \hat{\mathcal{L}}^*\hat{\mathcal{L}}\| < 1. \tag{A.12}$$

Based on the submultiplicativity of the operator norm, the upper-bound of $\|\mathbf{\Phi}^\mathsf{T}\mathbf{\Phi} + \hat{\mathcal{L}}^*\hat{\mathcal{L}}\|$ is given by

$$\|\mathbf{\Phi}^\mathsf{T}\mathbf{\Phi} + \hat{\mathcal{L}}^*\hat{\mathcal{L}}\| \leq \|\mathbf{\Phi}^\mathsf{T}\mathbf{\Phi}\| + \|\hat{\mathcal{L}}^*\hat{\mathcal{L}}\| \leq 2(N+2) \tag{A.13}$$

because of $\|\hat{\mathcal{L}}^*\hat{\mathcal{L}}\| = 2N$ and $\|\mathbf{\Phi}^\mathsf{T}\mathbf{\Phi}\| \leq 4$. Consequently, the convergence condition in (A.12) can be rewritten as

$$\gamma_1 < \frac{1}{2\gamma_2(N+2)}. \tag{A.14}$$

In the proposed algorithm, most of the computation burden comes from the computation of the proximal operator of $\log \mathrm{gdet}(\cdot)$. It needs the eigendecomposition and its computation cost is $\mathcal{O}(N^3)$. Thus, the computational complexity of Algorithm 6 is $\mathcal{O}(KN^3)$ per iteration.

Algorithm 6 Time-varying graph learning

Input: $\mathbf{w}_{\text{all}}^{(0)}$, $\mathbf{p}^{(0)}$, $\mathbf{V}^{(0)}$, $\{\mathbf{x}_t\}_{t=0}^{T-1}$, K, ϵ
Output: $\mathbf{w}_{\text{all}}$

Divide $\{\mathbf{x}_t\}_{t=0}^{T}$ into K data segments $\mathbf{X}^{(0)} \dots \mathbf{X}^{(K-1)}$
Compute $\{\mathbf{Z}^{(k)}\}_{k=0}^{K-1}$ from $\{\mathbf{X}^{(k)}\}_{k=0}^{K-1}$
while $\|\mathbf{w}_{\text{all}}^{(i+1)} - \mathbf{w}_{\text{all}}^{(i)}\| / \|\mathbf{w}_{\text{all}}^{(i)}\| > \epsilon$ **do**
  $\mathbf{w}_{\text{all}}^{(n+1)} := \text{prox}_{\gamma_1 g}(\mathbf{w}_{\text{all}}^{(n)} - \gamma_1(\mathbf{\Phi}^{\mathsf{T}}\mathbf{p} + \hat{\mathcal{L}}^{*}\mathbf{V}))$
  $\mathbf{p}^{(n)} \leftarrow \mathbf{p}^{(n)} + \gamma_2 \mathbf{\Phi}(2\mathbf{w}_{\text{all}}^{(n+1)} - \mathbf{w}_{\text{all}}^{(n)})$
  $\mathbf{V}^{(n)} \leftarrow \mathbf{V}^{(n)} + \gamma_2 \hat{\mathcal{L}}(2\mathbf{w}_{\text{all}}^{(n+1)} - \mathbf{w}_{\text{all}}^{(n)})$
  $\mathbf{p}^{(n+1)} := \mathbf{p}^{(n)} - \gamma_2 \,\text{prox}_{\frac{1}{\gamma_2} h_1}\left(\frac{\mathbf{p}^{(n)}}{\gamma_2}\right)$
  for $k = 0, \dots, K-1$ **do**
    $\mathbf{V}_k^{(n+1)} := \mathbf{V}_k^{(n)} - \gamma_2 \,\text{prox}_{\frac{1}{\gamma_2}(-\log \text{gdet}(\cdot))}\left(\frac{\mathbf{V}_k^{(n)}}{\gamma_2}\right)$
  end for
  $\mathbf{V}^{(n+1)} = \left[\mathbf{V}_0^{(n+1)\mathsf{T}}, \dots, \mathbf{V}_{K-1}^{(n+1)\mathsf{T}}\right]^{\mathsf{T}}$
  $i \leftarrow i + 1$
end while

Bibliography

[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[2] T. N. Kipf and M. Welling, "Semi-Supervised Classification with Graph Convolutional Networks," *ArXiv160902907 Cs Stat*, 2016.

[3] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Advances Neural Inf. Process. Syst.* Curran Associates, Inc., 2016, pp. 3844–3852.

[4] Y. Zhang and M. Rabbat, "A Graph-CNN for 3D Point Cloud Classification," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 6279–6283.

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

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

[7] C. Nowzari, V. M. Preciado, and G. J. Pappas, "Analysis and control of epidemics: A survey of spreading processes on complex networks," *IEEE Control Syst. Mag.*, vol. 36, no. 1, pp. 26–46, 2016.

[8] A. Ganesh, L. Massoulie, and D. Towsley, "The effect of network topology on the spread of epidemics," in *Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies.*, vol. 2, 2005, pp. 1455–1466 vol. 2.

[9] V. Colizza, A. Barrat, M. Barthélemy, and A. Vespignani, "The role of the airline transportation network in the prediction and predictability of global epidemics," *PNAS*, vol. 103, no. 7, pp. 2015–2020, 2006.

[10] A. Gadde, S. K. Narang, and A. Ortega, "Bilateral filter: Graph spectral interpretation and extensions," in *Proc. IEEE Int. Conf. Image Process.*, 2013, pp. 1222–1226.

[11] Y. Wang, A. Ortega, D. Tian, and A. Vetro, "A graph-based joint bilateral approach for depth enhancement," in *Proc. IEEE Int. Conf. Acoust. Speech. Signal Process.*, 2014, pp. 885–889.

[12] M. Onuki, S. Ono, M. Yamagishi, and Y. Tanaka, "Graph signal denoising via trilateral filter on graph spectral domain," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 2, no. 2, pp. 137–148, 2016.

[13] A. Elmoataz, O. Lezoray, and S. Bougleux, "Nonlocal discrete regularization on weighted graphs: A framework for image and manifold processing," *IEEE Trans. Image Process.*, vol. 17, no. 7, pp. 1047–1060, 2008.

[14] C. Couprie, L. Grady, L. Najman, J. Pesquet, and H. Talbot, "Dual constrained TV-based regularization on graphs," *SIAM J. Imaging Sci.*, vol. 6, no. 3, pp. 1246–1273, 2013.

[15] S. Ono, I. Yamada, and I. Kumazawa, "Total generalized variation for graph signals," in *Proc. IEEE Int. Conf. Acoust. Speech. Signal Process.*, 2015, pp. 5456–5460.

[16] N. Shahid, N. Perraudin, V. Kalofolias, G. Puy, and P. Vandergheynst, "Fast robust PCA on graphs," *IEEE J. Sel. Top. Signal Process.*, vol. 10, no. 4, pp. 740–756, 2016.

[17] F. Shang, L. C. Jiao, and F. Wang, "Graph dual regularization non-negative matrix factorization for co-clustering," *Pattern Recognition*, vol. 45, no. 6, pp. 2237–2250, 2012.

[18] Z. Zhang and K. Zhao, "Low-rank matrix approximation with manifold regularization," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 7, pp. 1717–1729, 2013.

[19] J. Zhang and J. M. F. Moura, "Diffusion in social networks as SIS epidemics: Beyond full mixing and complete graphs," *IEEE J. Sel. Top. Signal Process.*, vol. 8, no. 4, pp. 537–551, 2014.

[20] D. I. Shuman, P. Vandergheynst, D. Kressner, and P. Frossard, "Distributed Signal Processing via Chebyshev Polynomial Approximation," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 4, no. 4, pp. 736–751, 2018.

[21] J. Hara, K. Yamada, S. Ono, and Y. Tanaka, "Design of Graph Signal Sampling Matrices for Arbitrary Signal Subspaces," in *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021, pp. 5275–5279.

[22] C. Valenzuela and F. Tobar, "Low-pass Filtering as Bayesian Inference," in *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 3367–3371.

[23] Y. Tanaka, "Spectral Domain Sampling of Graph Signals," *IEEE Trans. Signal Process.*, vol. 66, no. 14, pp. 3752–3767, 2018.

[24] Y. Tanaka, Y. C. Eldar, A. Ortega, and G. Cheung, "Sampling signals on graphs: From theory to applications," *IEEE Signal Process. Mag.*, vol. 37, no. 6, pp. 14–30, 2020.

[25] Y. Tanaka and Y. C. Eldar, "Generalized Sampling on Graphs With Subspace and Smoothness Priors," *IEEE Trans. Signal Process.*, vol. 68, pp. 2272–2286, 2020.

[26] A. G. Marques, S. Segarra, G. Leus, and A. Ribeiro, "Sampling of Graph Signals With Successive Local Aggregations," *IEEE Trans. Signal Process.*, vol. 64, no. 7, pp. 1832–1843, 2016.

[27] A. Hashemi, R. Shafipour, H. Vikalo, and G. Mateos, "Accelerated Sampling of Bandlimited Graph Signals," *ArXiv180707222 Eess*, 2018.

[28] A. Anis, A. Gadde, and A. Ortega, "Efficient Sampling Set Selection for Bandlimited Graph Signals Using Graph Spectral Proxies," *IEEE Trans. Signal Process.*, vol. 64, no. 14, pp. 3775–3789, 2016.

[29] A. Sakiyama, Y. Tanaka, T. Tanaka, and A. Ortega, "Eigendecomposition-Free Sampling Set Selection for Graph Signals," *IEEE Trans. Signal Process.*, vol. 67, no. 10, pp. 2679–2692, 2019.

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

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

[32] G. B. Giannakis, Y. Shen, and G. V. Karanikolas, "Topology identification and learning over graphs: Accounting for nonlinearities and dynamics," *Proc. IEEE*, vol. 106, no. 5, pp. 787–807, 2018.

[33] S. K. Kadambari and S. Prabhakar Chepuri, "Learning Product Graphs from Multidomain Signals," in *Proc. IEEE Int. Conf. Acoust. Speech. Signal Process.*, 2020, pp. 5665–5669.

[34] Y. Liu, L. Yang, K. You, W. Guo, and W. Wang, "Graph Learning Based on Spatiotemporal Smoothness for Time-Varying Graph Signal," *IEEE Access*, vol. 7, pp. 62 372–62 386, 2019.

[35] M. A. Lodhi and W. U. Bajwa, "Learning Product Graphs Underlying Smooth Graph Signals," *ArXiv200211277 Cs Eess*, 2020.

[36] X. Chen, Y. Liu, H. Liu, and J. G. Carbonell, "Learning spatial-temporal varying graphs with applications to climate data analysis," in *Proc. AAAI Conf. Artif. Intell.* AAAI Press, 2010, pp. 425–430.

[37] V. Kalofolias, "How to learn a graph from smooth signals," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2016, pp. 920–929.

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

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

[40] B. Pasdeloup, V. Gripon, G. Mercier, D. Pastor, and M. G. Rabbat, "Characterization and inference of graph diffusion processes from observations of stationary signals," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 4, no. 3, pp. 481–496, 2018.

[41] D. Thanou, X. Dong, D. Kressner, and P. Frossard, "Learning heat diffusion graphs," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 3, no. 3, pp. 484–499, 2017.

[42] J. Mei and J. M. F. Moura, "Signal processing on graphs: Causal modeling of unstructured data," *IEEE Trans. Signal Process.*, vol. 65, no. 8, pp. 2077–2092, 2017.

[43] S. P. Chepuri, S. Liu, G. Leus, and A. O. Hero, "Learning sparse graphs under smoothness prior," in *Proc. IEEE Int. Conf. Acoust. Speech. Signal Process.*, 2017, pp. 6508–6512.

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

[45] H. E. Egilmez, E. Pavez, and A. Ortega, "Graph learning from filtered signals: Graph system and diffusion kernel identification," *IEEE Trans. Signal Inf. Process. Netw.*, pp. 1–1, 2018.

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

[47] Y. Kim, S. Han, S. Choi, and D. Hwang, "Inference of dynamic networks using time-course data," *Brief. Bioinformatics*, vol. 15, no. 2, pp. 212–228, 2014.

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

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

[50] ——, "Time-varying graph learning with constraints on graph temporal variation," *ArXiv200103346 Cs Eess*, 2020.

[51] K. Yamada and Y. Tanaka, "Learning graphs with multiple temporal resolutions," in *Proc. APSIPA ASC*, 2020, pp. 139–142.

[52] ——, "Temporal Multiresolution Graph Learning," *IEEE Access*, vol. 9, pp. 143 734–143 745, 2021.

[53] S. Kumar, J. Ying, J. V. de Miranda Cardoso, and D. Palomar, "Structured graph learning via Laplacian spectral constraints," in *Advances Neural Inf. Process. Syst.* Curran Associates, Inc., 2019, pp. 11 647–11 658.

[54] V. Kalofolias, A. Loukas, D. Thanou, and P. Frossard, "Learning time varying graphs," in *Proc. IEEE Int. Conf. Acoust. Speech. Signal Process.*, 2017, pp. 2826–2830.

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

[56] S. Segarra, Y. Wangt, C. Uhler, and A. G. Marques, "Joint inference of networks from stationary graph signals," in *2017 51st Asilomar Conference on Signals, Systems, and Computers*, 2017, pp. 975–979.

[57] M. Navarro, Y. Wang, A. G. Marques, C. Uhler, and S. Segarra, "Joint Inference of multiple graphs from matrix polynomials," *ArXiv201008120 Cs Eess Stat*, 2020.

[58] H. P. Maretic and P. Frossard, "Graph Laplacian mixture model," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 6, pp. 261–270, 2020.

[59] J. Kao, D. Tian, H. Mansour, A. Ortega, and A. Vetro, "Disc-GLasso: Discriminative graph learning with sparsity regularization," in *Proc. IEEE Int. Conf. Acoust. Speech. Signal Process.*, 2017, pp. 2956–2960.

[60] S. Yang, Z. Lu, X. Shen, P. Wonka, and J. Ye, "Fused multiple graphical Lasso," *SIAM J. Optim.*, vol. 25, no. 2, pp. 916–943, 2015.

[61] L. Gan, X. Yang, N. Narisetty, and F. Liang, "Bayesian joint stimation of multiple graphical models," *Adv. Neural Inf. Process. Syst.*, vol. 32, pp. 9802–9812, 2019.

[62] P. Danaher, P. Wang, and D. M. Witten, "The joint graphical lasso for inverse covariance estimation across multiple classes," *J R Stat Soc Series B Stat Methodol*, vol. 76, no. 2, pp. 373–397, 2014.

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

[64] N. Parikh and S. Boyd, "Proximal algorithms," *Found Trends Optim*, vol. 1, no. 3, pp. 127–239, 2014.

[65] N. Perraudin and P. Vandergheynst, "Stationary Signal Processing on Graphs," *IEEE Trans. Signal Process.*, vol. 65, no. 13, pp. 3462–3477, 2017.

[66] R. P. Monti, P. Hellyer, D. Sharp, R. Leech, C. Anagnostopoulos, and G. Montana, "Estimating time-varying brain connectivity networks from functional MRI time series," *NeuroImage*, vol. 103, pp. 427–443, 2014.

[67] J. Wang, S. Qiu, Y. Xu, Z. Liu, X. Wen, X. Hu, R. Zhang, M. Li, W. Wang, and R. Huang, "Graph theoretical analysis reveals disrupted topological properties of whole brain functional networks in temporal lobe epilepsy," *Clinical Neurophysiology*, vol. 125, no. 9, pp. 1744–1756, 2014.

[68] L. Condat, "A primal–dual splitting method for convex optimization involving Lipschitzian, proximable and linear composite terms," *J Optim Theory Appl*, vol. 158, no. 2, pp. 460–479, 2013.

[69] R. Tibshirani, M. Saunders, S. Rosset, J. Zhu, and K. Knight, "Sparsity and smoothness via the fused Lasso," *J. R. Stat. Soc. Ser. B Stat. Methodol.*, vol. 67, no. 1, pp. 91–108, 2005.

[70] L. Meier, S. V. D. Geer, and P. Bühlmann, "The group Lasso for logistic regression," *J. R. Stat. Soc. Ser. B Stat. Methodol.*, vol. 70, no. 1, pp. 53–71, 2008.

[71] N. Komodakis and J. Pesquet, "Playing with duality: An overview of recent primal-dual approaches for solving large-scale optimization problems," *IEEE Signal Process. Mag.*, vol. 32, no. 6, pp. 31–54, 2015.

[72] I. Daubechies, M. Defrise, and C. De Mol, "An iterative thresholding algorithm for linear inverse problems with a sparsity constraint," *Commun. Pure Appl. Math.*, vol. 57, no. 11, pp. 1413–1457, 2004.

[73] P. L. Combettes and J.-C. Pesquet, "Primal-dual splitting algorithm for solving inclusions with mixtures of composite, Lipschitzian, and parallel-sum type monotone operators," *Set-Valued Anal*, vol. 20, no. 2, pp. 307–330, 2012.

[74] D. B. Johnson and D. A. Maltz, "Dynamic source routing in ad hoc wireless networks," in *Mobile Computing*. Springer, 1996, pp. 153–181.

[75] E. A. Allen, E. Damaraju, S. M. Plis, E. B. Erhardt, T. Eichele, and V. D. Calhoun, "Tracking whole-brain connectivity dynamics in the resting state," *Cereb. Cortex*, vol. 24, no. 3, pp. 663–676, 2014.

[76] J. Taghia, S. Ryali, T. Chen, K. Supekar, W. Cai, and V. Menon, "Bayesian switching factor analysis for estimating time-varying functional connectivity in fMRI," *Neuroimage*, vol. 155, pp. 271–290, 2017.

[77] F. Zhang and E. R. Hancock, "Graph spectral image smoothing using the heat kernel," *Pattern Recogn*, vol. 41, no. 11, pp. 3328–3342, 2008.

[78] J. Gall, C. Stoll, E. de Aguiar, C. Theobalt, B. Rosenhahn, and H. Seidel, "Motion capture using joint skeleton tracking and surface estimation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2009, pp. 1746–1753.

[79] Italian Civil Protection Department, M. Morettini, A. Sbrollini, I. Marcantoni, and L. Burattini, "COVID-19 in Italy: Dataset of the Italian Civil Protection Department," *Data in Brief*, vol. 30, p. 105526, 2020.

[80] E. Dong, H. Du, and L. Gardner, "An interactive web-based dashboard to track COVID-19 in real time," *The Lancet Infectious Diseases*, vol. 20, no. 5, pp. 533–534, 2020.

[81] J. V. d. M. Cardoso and D. P. Palomar, "Learning Undirected Graphs in Financial Markets," *ArXiv200509958 Q-Fin Stat*, 2020.

[82] A. Holbrook, "Differentiating the pseudo determinant," *Linear Algebra and its Applications*, vol. 548, pp. 293–304, 2018.

[83] M. A. Gelbart, J. Snoek, and R. P. Adams, "Bayesian optimization with unknown constraints," in *Proc. Uncertainty in Artificial Intelligence*. AUAI Press, 2014, pp. 250–259.

[84] A. E. Clementi, C. Macci, A. Monti, F. Pasquale, and R. Silvestri, "Flooding time in edge-Markovian dynamic graphs," in *Proceedings of the Twenty-Seventh ACM Symposium on Principles of Distributed Computing*. Association for Computing Machinery, 2008, pp. 213–222.

[85] J. K. Tugnait, "Sparse-Group Lasso for Graph Learning From Multi-Attribute Data," *IEEE Trans. Signal Process.*, vol. 69, pp. 1771–1786, 2021.

[86] G. Schwarz, "Estimating the Dimension of a Model," *Ann. Stat.*, vol. 6, no. 2, pp. 461–464, 1978.

[87] S. Watanabe, "A Widely Applicable Bayesian Information Criterion," *J. Mach. Learn. Res.*, vol. 14, no. Mar, pp. 867–897, 2013.

[88] C. Robert, *The Bayesian Choice: From Decision-Theoretic Foundations to Computational Implementation*, 2nd ed. Springer-Verlag, 2007.

[89] T. Park and G. Casella, "The Bayesian Lasso," *J. Am. Stat. Assoc.*, vol. 103, no. 482, pp. 681–686, 2008.

[90] H. Wang, "Bayesian Graphical Lasso Models and Efficient Posterior Computation," *Bayesian Anal.*, vol. 7, no. 4, pp. 867–886, 2012.

[91] G. H. Golub and C. F. V. Loan, *Matrix Computations*. JHU Press, 2013.

[92] K. Sato, "Optimal graph Laplacian," *Automatica*, vol. 103, pp. 374–378, 2019.