T2R2 東京科学大学 リサーチリポジトリ
Science Tokyo Research Repository

論文 / 著書情報
Article / Book Information

題目(和文)	
Title(English)	Estimating Time-Varying Signals over Dynamic Graphs via Proximal Splitting Techniques
著者(和文)	山縣英介
Author(English)	Eisuke Yamagata
出典(和文)	学位:博士(工学), 学位授与機関:東京科学大学, 報告番号:甲第370号, 授与年月日:2025年3月26日, 学位の種別:課程博士, 審査員:小野 峻佑,小野 功,高安 美佐子,村田 剛志,横田 理央
Citation(English)	Degree:Doctor (Engineering), Conferring organization: Institute of Science Tokyo, Report number:甲第370号, Conferred date:2025/3/26, Degree Type:Course doctor, Examiner:,,,,
学位種別(和文)	博士論文
Type(English)	Doctoral Thesis

Powered by T2R2 (Science Tokyo Research Repository)

Doctoral Dissertation

Estimating Time-Varying Signals over Dynamic Graphs via Proximal Splitting Techniques

Eisuke Yamagata

Graduate Major in Artificial Intelligence
School of Computing
Institute of Science Tokyo

Supervisor: Shunsuke Ono

January, 2025

Abstract

This dissertation proposes optimization-based methods for estimating time-varying signals over dynamic graph topologies by leveraging graph signal processing and proximal splitting techniques. With the increasing demand for sophisticated data analysis across various applications, the importance of effective data collection and utilization has undergone significant transformation. Extracting meaningful information from collected data is crucial for enhancing conventional analyses and enabling unprecedented data utilization.

Time-varying signal processing focuses on extracting temporal relationships within data, while graph signal processing leverages spatial relationships by interpreting data points as nodes in a graph. By combining these techniques, time-varying graph signal processing has emerged as a promising framework for handling data that can be represented as time-varying graph signals, with applications in environmental monitoring, traffic optimization, finance, and more.

While data-driven methods—particularly those based on machine learning—are increasingly common for handling complex data structures, they have notable drawbacks. They often require large amounts of training data, which may be infeasible in domains like environmental monitoring, where data collection is expensive. Additionally, their black-box nature can limit transparency and interpretability.

To address these challenges, we focus on an optimization-based approach suitable for limited time-varying graph signal data—one that does not require extensive training datasets and maintains high interpretability. Conventional optimization-based methods for time-varying graph signal processing have primarily assumed static graph topologies. However, in applications involving mobile sensor networks—such as sensor-equipped drones or vehicles—it is more realistic to assume that the network topology changes over time.

Accordingly, this dissertation proposes time-varying graph signal estimation methods over dynamic graph topologies using proximal splitting techniques. We demonstrate the effectiveness of dynamic graph assumptions over static ones in both physical and non-physical sensing domains, namely environmental monitoring and finance.

In Chapter 3, we introduce a time-varying graph signal recovery method capable of handling graph signals corrupted by various types of noise. With advancements in mobile devices equipped with high-quality sensors, using networks of such devices for sensing has become practical. Consequently, methods to recover such signals from various types of noise and corruption have become increasingly important. Upon considering network signal recovery, we leverage graph signal processing to effectively utilize the information inherent in the network topology. Unlike conventional methods that assume static graphs, we consider dynamic network topologies, which are more appropriate for mobile device networks. Our proposed method effectively leverages the network's dynamics by formulating an optimization problem under the assumption that the graph is time-varying. In addition to leveraging priors in the vertex domain, we also utilize priors in the temporal domain to enhance recovery performance. The proposed formulation can handle Gaussian noise, sparse noise, and missing values, making the method highly robust to corruption. We then solve the problem using a primal-splitting-based algorithm. We evaluate the

proposed method on both a synthetic dataset and a real-world dataset to show the effectiveness of dynamic graph representations over static graph representations in time-varying graph signal recovery.

In Chapter 4, we propose a sparse index tracking method that leverages market graphs to reduce investment risk. Financial indices like the S&P 500 and NASDAQ 100 are known for their stable growth and are attractive investment targets. However, investing directly in an index requires distributing capital across all its constituent assets, and the true weighting schemes used by index providers are often unknown. Investors must, therefore, construct their own portfolios to track the performance of the target index.

Ideally, the portfolio should be sparse to reduce transaction costs. Conventional methods impose sparsity using ℓ_1-norm regularizations, but these methods lack the ability to precisely control the number of nonzero weights in the portfolio. We introduce a sparse index tracking method based on an ℓ_0-norm constraint, allowing easy control of the number of nonzero weights by adjusting a single parameter. Furthermore, to mitigate risks associated with concentrated investments, we impose market graph neutrality in our formulation to diversify the investment. We represent relationships among assets using a market graph and apply graph clustering to partition them into groups. Our tracking method distributes capital across all groups, effectively dispersing investments and helping investors avoid the risks of concentrated portfolios. By reconstructing the groups frequently, we effectively capture the dynamics of the market graph in our method.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In this chapter, we will introduce the motivations for using proximal splitting optimization based on graph signal processing to estimate time-varying graph signals over dynamic graph topologies. This chapter will also establish the general objective of this Ph.D. study and clarify the contributions of this study with its outline.

1.1 Background

Time-varying signal processing is a fundamental instrument for extracting temporal relationships within data to enhance signal estimation. Many approaches and methods have been researched in the area to handle a wide variety of problems and data, e.g., the moving-average method for market analysis [6, 90], spectral analysis for speech processing [83], the Kalman filter for tracking [22,80], and the autoregressive model for time-series prediction [3], etc.

However, in light of recent advancements in sensor technology, data collected from sensors are often spatially distributed, and the spatial relationships between data points are essential for understanding the data. In the realm of physical sensing, a network of sensors has been deployed to monitor environmental changes [59,63,72], traffic conditions [34,110], and other phenomena. In the realm of non-physical sensing, SNS analysis [56] and financial analysis [8, 18] are some examples where the data points can be distributed unevenly in their respective spaces, which can be interpreted as spatial relationships. Utilizing information inherent in these spatial relationships along with the temporal relationships is essential for understanding the data and performing effective estimations.

Upon leveraging the spatial relationships between data points, graph signal processing [69, 87, 92] has emerged as a powerful tool for analyzing data. While conventional signal processing mainly targets data with regular structures, e.g., image processing that targets spatially regular data and speech processing that targets temporally regular data, graph signal processing can handle data with irregular structures, such as data observed on a network of sensors unevenly distributed in space, using edges and weights to represent the relationships between the data points. By interpreting signals with spatial relationships as graph signals and a time series of such signals as time-varying graph signals, time-varying graph signal processing has emerged as a promising framework for handling data that included complex spatial and temporal relationships. Graph signal processing can also be interpreted as a fusion of graph theory and signal processing, where graph theory is used to represent the spatial relationships (graph structure) of the data, and signal processing is used to analyze the data or signal itself. Graph signal processing fuses the two and handles signals that are defined on the vertices of the graph.

There are two fundamental approaches when applying time-varying graph signal processing: learning-based and optimization-based approaches. The learning-based approach

learns graph neural network models to capture the spatial and temporal relationships of the data and then uses them for analysis [42, 48, 101]. For instance, graph convolutional networks (GCNs) have been extended to handle time-varying data by incorporating recurrent or attention mechanisms, leading to architectures such as graph attention networks (GATs) and spatio-temporal graph neural networks [97, 106]. These methods excel at learning complex nonlinear relationships in dynamic data, making them particularly effective for applications like traffic flow prediction, social network evolution analysis, and temporal anomaly detection. However, the learning-based approach requires a large amount of training data to effectively train models, which is often unavailable in many applications. For example, when considering data observed on a sensor network, the data collection is often expensive due to the cost of deploying a large number of sensors and maintaining a large number of sensors. Furthermore, because the training of GNN models is often specific to each graph topology and its characteristics, transferring these trained models to other datasets can be difficult.

On the other hand, the optimization-based approach has been studied as a powerful tool that can handle a wide range of problems and data without the need for a large amount of training data. In the optimization-based approach, the estimation tasks are formulated as optimization problems, and then the estimation results can be obtained by solving the optimization problems using optimization algorithms [38, 85]. By incorporating prior knowledge, such as spatial and temporal priors, signal corruption models, etc., into the optimization problems, the optimization-based approach can provide effective estimation results without having to extract such information by training with a large amount of data.

Optimization problems that leverage numerous priors are often highly complex and composed of a wide variety of regularizations and constraints that are not necessarily differentiable. Due to the often non-differentiable nature of these problems, closed-form solutions or simple gradient-based optimization algorithms cannot be used. To this end, proximal splitting optimization algorithms have been applied in many studies to effectively handle and solve such complex and non-differentiable problems.

1.2 Prior Arts

In the context of time-varying graph signal processing, there are two important steps to effectively leveraging the spatial priors of the data: the preparation of the graph topology and the incorporation of that topology into the optimization problem.

1.2.1 Graph Construction

In graph signal processing, graphs are used as the mathematical representation of the spatial relationships between data points. Each data point is represented as a vertex in the graph, and the relationships between the nodes are represented as edges with weights. The construction of a graph that accurately reflects the spatial relationships of the data is essential for effective analysis. Similarly to the two approaches to time-varying graph signal processing, the preparation of the graph topology can be divided into two approaches, namely, learning-based and algorithm-based approaches.

In the learning-based approach [35, 47, 84], the graph topology is learned from the data to construct a graph that exhibits desirable traits for later use in graph signal processing. For example, in physical sensing scenarios, graph topologies are often used to leverage signal smoothness priors in the spatial (vertex) domain to denoise corrupted data; therefore, graphs that can house the observed data smoothly are often constructed. However, this often falls into a chicken-and-egg problem since the graph is desired to estimate the cor-

rupted data, but it must be constructed from the already-denoised data. To handle this problem, some studies have proposed methods to learn the graph from corrupted data by simultaneously estimating the true data. Despite this problem, the learning-based approach can deliver graphs that are tailored to the data and can be effective in many scenarios.

On the other hand, algorithm-based graph construction methods have been widely used in many studies for their simple implementations and relatively accurate approximations of the true graph. Some examples that are used in physical sensing are the k-nearest neighbor method, where the k-nearest neighbors of each data point are connected with edges, and the ε-radius method, where data points within a certain distance (ε) are connected with edges. After the edges are defined, the weights of the edges are often defined by the distance of the data points, e.g., the inverse of the distance or the Gaussian kernel of the distance (physical coordinates of the sensors are often paired with the observations in physical sensor network data).

1.2.2 Leveraging Graph Topology

Once the graph topology is constructed, it becomes a powerful tool for graph signal estimation by directly incorporating spatial relationships into data processing algorithms. Conventional studies often leverage the graph topology to promote the smoothness of the signal across the graph. This approach is based on the assumption that connected vertices are likely to have similar signal values, which holds true in many physical sensing scenarios where measurements at nearby locations are correlated [92].

One common method is to utilize the graph Laplacian, a matrix representation that encapsulates the connectivity and edge weights of the graph. By incorporating the graph Laplacian into optimization frameworks, researchers impose smoothness constraints on the estimated signals. This is achieved by penalizing large differences between signal values of connected vertices, effectively suppressing noise while preserving the intrinsic structure of the data [32]. Such regularization techniques enhance the signal quality by aligning the estimated signal with the underlying graph topology.

Another vertex-domain approach is total variation minimization on graphs, which aims to reduce the total variation of the signal over the graph. This method balances the need for smoothness with the preservation of important features, such as edges or abrupt changes in the signal [21]. For time-varying signals, this approach can be extended by incorporating temporal total variation, promoting smoothness not only across the graph but also over time.

In the context of time-varying graph signals, where the signal values change over time, leveraging the graph topology becomes even more crucial. A notable example is the work by [85], who proposed a method for reconstructing time-varying graph signals by incorporating both spatial and temporal smoothness. Their approach models the time-varying signal as a sequence of graph signals and employs a joint optimization framework that enforces smoothness over the graph at each time instance as well as smoothness over time. This method effectively captures the dynamic nature of the signals while leveraging the underlying graph structure, leading to improved estimation and denoising performance.

1.3 Limitations of Existing Methods

As mentioned in the previous section, constructing the graph and incorporating it into the optimization problem to leverage spatial priors jointly with temporal priors is essential for effective time-varying graph signal processing. This framework allows us to effectively handle complex data structures and relationships and can be applied to a wide range of

applications, such as environmental monitoring, traffic optimization, and financial analysis. However, the existing methods have several limitations that hinder their effectiveness in handling complex data structures and relationships.

Existing time-varying graph signal estimation methods are mostly established based on the assumption that the graph topology is static over time. This is due to the conventional sensing scheme where the sensor networks were often fixed, e.g., a network of climate observatories. However, in light of recent developments in sensor downsizing and mobile devices that can house such sensors (e.g., smartphones, drones, 3D tracking suits), the assumption of a static graph topology has become unrealistic in many cases. For example, in environmental monitoring, drones equipped with various sensors are often used to monitor environmental changes. In such a scenario, the network composed of drones is constantly moving, causing the network topology to change over time as well. In non-physical settings, for example, SNS analysis and financial analysis, the graph is often defined in correspondence to the signals, in which case the graph should be updated as the signals change over time.

When considering time-varying graph signal estimation on a dynamic graph, existing methods often involve applying the conventional static-graph-based methods or disregarding the temporal priors and applying spatial-prior-based methods to individual time instances. Both cases fail to effectively leverage the dynamics of the graph topology, leaving substantial room for improvement in estimation performance.

In the realm of learning-based approaches, a Graph Convolution Network (EvolveGCN) [79] has been proposed as an extension of GCNs that can handle dynamic graphs by updating the GCN parameters over time using recurrent neural networks. This allows the model to adapt to changes in the graph structure while learning from evolving signals. However, like conventional learning-based methods, this method also requires a large amount of training data to train the model effectively, and because the model is trained on specific datasets or graph types, transferring it to other datasets or graph types is often difficult. For example, in environmental monitoring or surveillance applications that use drones and satellites for data collection, the collection process is often expensive, and the data is often limited, making it difficult to train the model effectively. In financial domains, training models with historical data is often difficult due to the non-stationarity of the data and often failure to generalize to future data.

The concept of graph signal processing is relatively new in the signal processing domain, and time-varying graph signal processing is even more recent. The idea and application of time-varying graph signal processing on dynamic graphs is still in its infancy.

1.4 General Objective of This Study

In this dissertation, we propose optimization-based methods for estimating time-varying signals over dynamic graph topologies by leveraging graph signal processing and proximal splitting techniques. The discussion in the previous section brings us to the following research question: *How can we exploit the dynamics of the graph topology to enhance time-varying graph signal estimation in situations where the underlying graph topologies are explicitly dynamic?* To this end, we take the following approaches.

First, we construct a time-varying graph that can reasonably approximate the true dynamic graph. In doing so, we disregard overly sophisticated methods and focus on simple algorithms like the k-nearest neighbor method. Complex graph construction methods often require large amounts of data or uncorrupted data for accurate graph construction, which is often unavailable in realistic applications.

Next, we use the constructed graph to effectively leverage the spatial relationships of the data in the optimization problem. Unlike conventional methods, we utilize the dynamic

graphs so that the graph signals at each time instance are estimated in correspondence with the graph of that time instance. Furthermore, the temporal priors of the data are jointly leveraged in the optimization problem. Finally, we solve the optimization problem using proximal splitting optimization algorithms, which are effective in handling complex and non-differentiable problems. Since the methodology of leveraging these priors can greatly differ depending on the application, we explore various tasks and applications to demonstrate the effectiveness of the proposed methods.

This dissertation is organized as follows. First, Chapter 2 provides mathematical preliminaries for convex optimization and graph signal processing, which form the basis of the proposed methods. Next, Chapter 3 and Chapter 4 explore the possibilities of applying time-varying graph signal estimation over dynamic graphs to various tasks and applications. Chapter 3 focuses on physical sensing applications (i.e., environmental monitoring), while Chapter 4 expands the scope to non-physical sensing applications (i.e., financial analysis). The details are as follows.

- Chapter 3: We introduce a time-varying graph signal recovery method capable of handling graph signals corrupted by various types of noise. With advancements in mobile devices equipped with high-quality sensors, using networks of such devices for sensing has become practical. Consequently, methods to recover such signals from various types of noise and corruption have become increasingly important. In considering network signal recovery, we leverage graph signal processing to effectively utilize the information inherent in the network topology. Unlike conventional methods that assume static graphs, we consider dynamic network topologies, which are more appropriate for mobile device networks. Our proposed method effectively leverages the dynamics of the network topology by formulating an optimization problem based on the assumption that the graph is time-varying. In addition to leveraging priors in the vertex domain, we also leverage priors in the temporal domain to enhance the recovery performance. The proposed formulation is generalized to accommodate a wide variety of regularization terms, allowing us to explore the effects of these terms on various data types. The proposed formulation can also handle Gaussian noise, sparse noise, and missing values, making the proposed method highly robust to corruption. We then solve the problem using a primal-splitting-based algorithm. We evaluate the proposed method on both a synthetic and a real-world dataset to demonstrate the effectiveness of the dynamic graph representation over static graph representation in time-varying graph signal recovery. We also discuss the effects of the various regularization terms on estimation performance.

- Chapter 4: We propose a sparse index tracking method that leverages market graphs to reduce investment risk. Financial indices like the S&P 500 and NASDAQ 100 are known for their stable growth and are attractive investment targets. However, investing directly in an index requires distributing capital across all its constituent assets, and the true weighting schemes used by index providers are often unknown. Investors must, therefore, construct their own portfolios to track the performance of the target index. Before introducing the market graph, we first propose a novel sparse index tracking method that will serve as the basis of the proposed market graph-based method. Ideally, the portfolio should be sparse to reduce transaction costs. Conventional sparse index tracking methods impose sparsity using ℓ_1-norm regularization, but these methods lack the ability to precisely control the number of non-zero weights in the portfolio. We introduce a sparse index tracking method based on an ℓ_0-norm constraint that allows easy control of the number of non-zero weights by adjusting a single parameter. Based on the introduced method that uses the ℓ_0-norm, we propose a sparse index tracking method that can mitigate risk

associated with concentrated investments by imposing market graph neutrality in our formulation to diversify the investment. Conventional methods use fixed sectors to diversify the investment. On the other hand, we represent the relationships between assets using a market graph and apply graph clustering to partition them into groups. Our tracking method distributes capital across all groups, effectively dispersing investments and helping investors avoid the risks of concentrated portfolios. By reconstructing the groups frequently, we effectively reflect the dynamics of the market graph in our method. We evaluate the proposed method on real-world financial datasets to demonstrate its improvements over existing methods.

In Chapter 5, we conclude the dissertation by summarizing the results obtained in the earlier chapters and giving an outlook on future research directions.

Chapter 2

Preliminaries

In this section, we present the notation, definitions, and fundamental concepts in linear algebra and convex optimization that will be used throughout this dissertation. Our goal is to provide a self-contained reference, emphasizing properties that will be crucial in the subsequent analyses and algorithmic developments. By elaborating on these foundational elements, we aim to enhance understanding and ensure clarity for readers who may not be familiar with all aspects of the subject matter.

2.1 Notation and Conventions

We work in finite-dimensional Euclidean spaces throughout this dissertation. Let $\mathbb{R}^n$ denote the n-dimensional real vector space. We adopt the following notational conventions:

- **Vectors and Matrices**: Bold lowercase letters (e.g., $\mathbf{x} \in \mathbb{R}^n$) denote vectors, and bold uppercase letters (e.g., $\mathbf{A} \in \mathbb{R}^{m\times n}$) denote matrices. For a vector $\mathbf{x} = (x_1, x_2, \ldots, x_n)$, its components are denoted by x_i for $i = 1, \ldots, n$. For a matrix $\mathbf{A} = [A_{ij}] \in \mathbb{R}^{m\times n}$, the element in the i-th row and j-th column is denoted by A_{ij} or $[\mathbf{A}]_{ij}$.
- **Zero Vectors and Matrices**: The symbol $\mathbf{0}$ denotes zero vectors or zero matrices of suitable dimensions; the specific dimension will be clear from the context.
- **Identity Matrix**: The identity matrix in $\mathbb{R}^{n\times n}$ is denoted by $\mathbf{I}_n$ or simply $\mathbf{I}$ when the dimension is clear from the context.

Matrix and Block Matrix Representations

It is often convenient to represent matrices in block form, especially when dealing with structured problems. Suppose $\mathbf{A} \in \mathbb{R}^{m\times n}$ is partitioned into blocks $\mathbf{A}_{ij} \in \mathbb{R}^{m_i\times n_j}$, where the row and column dimensions satisfy $\sum_{i=1}^{p} m_i = m$ and $\sum_{j=1}^{q} n_j = n$. Then we can write $\mathbf{A}$ as:

$$\mathbf{A} = [\mathbf{A}_{ij}]_{1\le i\le p,\, 1\le j\le q} = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} & \cdots & \mathbf{A}_{1q} \\ \mathbf{A}_{21} & \mathbf{A}_{22} & \cdots & \mathbf{A}_{2q} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{A}_{p1} & \mathbf{A}_{p2} & \cdots & \mathbf{A}_{pq} \end{bmatrix}. \tag{2.1}$$

Block matrix representation is particularly useful in simplifying computations, exploiting sparsity, and understanding the underlying structure of linear transformations.

Transpose and Diagonal Structures

Transpose: For a matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$, the transpose $\mathbf{A}^\top \in \mathbb{R}^{n\times m}$ is defined by

$$[\mathbf{A}^\top]_{ji} = A_{ij}, \quad \text{for } i = 1,\dots,m;\ j = 1,\dots,n. \tag{2.2}$$

The transpose operation reflects the matrix across its main diagonal, interchanging rows and columns.

Diagonal Matrices: A diagonal matrix is a square matrix where all off-diagonal elements are zero.

Given a vector $\mathbf{x} = (x_1, x_2, \dots, x_n)$, the corresponding diagonal matrix is

$$\operatorname{diag}(\mathbf{x}) := \begin{bmatrix} x_1 & 0 & \cdots & 0 \\ 0 & x_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & x_n \end{bmatrix}. \tag{2.3}$$

2.1.1 Selected Elements in Linear Algebra

Difinition 1 (Inner Product)**.** The *inner product* is a fundamental operation that generalizes the dot product to higher dimensions and is central to many concepts in linear algebra.

1. **Vectors**: For two vectors $\mathbf{x}_1, \mathbf{x}_2 \in \mathbb{R}^n$, their inner product is defined as

$$\langle \mathbf{x}_1, \mathbf{x}_2 \rangle := \sum_{i=1}^{n} [\mathbf{x}_1]_i [\mathbf{x}_2]_i = \mathbf{x}_1^\top \mathbf{x}_2. \tag{2.4}$$

   This operation measures the degree of alignment between two vectors.

2. **Matrices**: For matrices $\mathbf{X}_1, \mathbf{X}_2 \in \mathbb{R}^{n\times m}$, their inner product is defined by

$$\langle \mathbf{X}_1, \mathbf{X}_2 \rangle := \sum_{i=1}^{n}\sum_{j=1}^{m} [\mathbf{X}_1]_{ij} [\mathbf{X}_2]_{ij} = \operatorname{Tr}(\mathbf{X}_1^\top \mathbf{X}_2). \tag{2.5}$$

   The inner product for matrices extends the concept to higher dimensions and is essential in deriving many matrix-related properties.

Difinition 2 (Norms Induced by Inner Product)**.** Norms provide a measure of the size or length of vectors and matrices.

Vector Norms: The *ℓ_2-norm* (Euclidean norm) of a vector $\mathbf{x} \in \mathbb{R}^n$, induced by the inner product (2.4), is defined as

$$\|\mathbf{x}\|_2 := \sqrt{\langle \mathbf{x}, \mathbf{x} \rangle} = \sqrt{\sum_{i=1}^{n} x_i^2}. \tag{2.6}$$

This norm measures the length of the vector in Euclidean space.

Matrix Norms: The *Frobenius norm* of a matrix $\mathbf{X} \in \mathbb{R}^{n\times m}$, derived from the inner product (2.5), is defined by

$$\|\mathbf{X}\|_F := \sqrt{\langle \mathbf{X}, \mathbf{X}\rangle} = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{m} X_{ij}^2}. \tag{2.7}$$

This norm is analogous to the Euclidean norm for vectors and provides a measure of the magnitude of a matrix.

2.2 Proximal Tools and Algorithms

Proximal operators and algorithms are powerful tools in convex optimization, particularly for problems involving non-smooth functions. They generalize the concept of projections and play a central role in designing efficient algorithms.

2.2.1 Proximity Operators

Difinition 3 (Proximity Operator)**.** For a function $f \in \Gamma_0(\mathbb{R}^n)$ and a parameter $\gamma > 0$, the *proximity operator* of f is the mapping $\mathrm{prox}_{\gamma f} : \mathbb{R}^n \to \mathbb{R}^n$ defined by

$$\mathrm{prox}_{\gamma f}(\mathbf{x}) := \underset{\mathbf{y}\in\mathbb{R}^n}{\arg\min} \left\{ f(\mathbf{y}) + \frac{1}{2\gamma}\|\mathbf{y}-\mathbf{x}\|_2^2 \right\}. \tag{2.8}$$

The proximity operator can be interpreted as a generalized projection that balances proximity to $\mathbf{x}$ and the function value of f. It is a central concept in proximal algorithms for optimization.

If the proximity operator $\mathrm{prox}_{\gamma f}$ can be computed efficiently, we say that f is *proximable*.

Proposition 2.2.1 (Moreau's Identity)**.** *For a function $f \in \Gamma_0(\mathbb{R}^n)$ and $\gamma > 0$, the following identity holds between the proximity operators of f and its Fenchel conjugate f^*:*

$$\mathrm{prox}_{\gamma f^*}(\mathbf{x}) = \mathbf{x} - \gamma \mathrm{prox}_{\frac{1}{\gamma} f}\left(\frac{\mathbf{x}}{\gamma}\right). \tag{2.9}$$

Moreau's identity provides a valuable relationship that allows computation of one proximity operator from another and is often used in algorithm derivations.

Example 1 (Proximity Operator of the ℓ_1-Norm)**.** Let $f(\mathbf{x}) = \|\mathbf{x}\|_1$, the ℓ_1-norm, and $\gamma > 0$. The proximity operator of γf corresponds to the *soft-thresholding* operation. For each component $j = 1, \ldots, n$,

$$[\mathrm{prox}_{\gamma f}(\mathbf{x})]_j = \mathrm{sign}(x_j)\max\{|x_j| - \gamma, 0\}.$$

Here, $\mathrm{sign}(x_j)$ denotes the sign of x_j, which is 1 if $x_j > 0$, -1 if $x_j < 0$, and 0 if $x_j = 0$. The soft-thresholding operator shrinks the components of $\mathbf{x}$ towards zero by γ, setting them to zero if they are within γ of zero. This operation is fundamental in algorithms for sparse recovery and regularization.

Difinition 4 (Metric Projection onto a Closed Convex Set)**.** For any nonempty closed convex set $C \subset \mathbb{R}^n$, the *metric projection* onto C is the mapping $\mathrm{P}_C : \mathbb{R}^n \to C$ defined by

$$\mathrm{P}_C(\mathbf{x}) := \underset{\mathbf{y}\in C}{\arg\min} \|\mathbf{x}-\mathbf{y}\|_2. \tag{2.10}$$

This means that $\mathrm{P}_C(\mathbf{x})$ is the closest point in C to $\mathbf{x}$ with respect to the Euclidean norm. In the context of convex sets, the projection is unique and nonexpansive, making it a valuable tool in optimization algorithms.

Example 2 (Proximity Operator of an Indicator Function)**.** For $f = \iota_C$, the indicator function of a closed convex set $C \subset \mathbb{R}^n$, the proximity operator reduces to the metric projection onto C:

$$\text{prox}_{\gamma\iota_C}(\mathbf{x}) = \text{P}_C(\mathbf{x}), \quad \forall\gamma > 0. \tag{2.11}$$

This means that the proximity operator of the indicator function is independent of γ and is simply the closest point in C to $\mathbf{x}$. This relationship bridges the concepts of projections and proximal operators.

2.2.2 Primal-Dual Splitting Methods

Primal-dual splitting methods are iterative algorithms designed to solve convex optimization problems that involve both differentiable and non-differentiable functions, as well as operations involving given matrices. These methods enable decomposing complex problems into simpler, more manageable subproblems.

Consider the convex optimization problem:

$$\min_{\mathbf{u}\in\mathbb{R}^n} f_1(\mathbf{u}) + f_2(\mathbf{u}) + f_3(\mathbf{A}\mathbf{u}), \tag{2.12}$$

where:

- $f_1 : \mathbb{R}^n \to \mathbb{R}$ is differentiable with a β-Lipschitz continuous gradient, i.e., $\|\nabla f_1(\mathbf{x}) - \nabla f_1(\mathbf{y})\|_2 \leq \beta\|\mathbf{x} - \mathbf{y}\|_2$ for all $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$.
- $f_2 \in \Gamma_0(\mathbb{R}^n)$ is a proximable convex function.
- $f_3 \in \Gamma_0(\mathbb{R}^m)$ is a proximable convex function defined on $\mathbb{R}^m$.
- $\mathbf{A} \in \mathbb{R}^{m\times n}$ is a given matrix.

The presence of non-differentiable functions and a matrix transformation $\mathbf{A}\mathbf{u}$ can pose challenges in direct optimization. Primal-dual splitting methods address this by maintaining a pair of variables that represent the primal and dual problems, respectively.

A typical primal-dual splitting iteration generates sequences $\{\mathbf{u}^{(i)}, \mathbf{v}^{(i)}\}_{i\geq 0}$ according to:

$$\mathbf{u}^{(i+1)} = \text{prox}_{\gamma_1 f_2}\left(\mathbf{u}^{(i)} - \gamma_1\left(\nabla f_1(\mathbf{u}^{(i)}) + \mathbf{A}^\top\mathbf{v}^{(i)}\right)\right), \tag{2.13}$$

$$\mathbf{v}^{(i+1)} = \text{prox}_{\gamma_2 f_3^*}\left(\mathbf{v}^{(i)} + \gamma_2\mathbf{A}(2\mathbf{u}^{(i+1)} - \mathbf{u}^{(i)})\right), \tag{2.14}$$

where $\gamma_1, \gamma_2 > 0$ are step sizes and f_3^* is the Fenchel conjugate of f_3.

These updates alternate between the primal variable $\mathbf{u}$ and the dual variable $\mathbf{v}$, leveraging proximal operators to handle non-differentiable terms. Under appropriate conditions—such as convexity, lower semicontinuity, and properness of the involved functions—along with suitable choices of the step sizes (which depend on β and the norm of the matrix $\mathbf{A}$), these iterations converge to a solution of the original problem.

PDS has played a central role in various signal estimation methods, e.g., [13, 25, 51, 67, 74, 75, 93, 105]. A comprehensive review on PDS can be found in [28, 50].

2.3 Fundamentals of Graph Signals

Difinition 5 (Graph and Graph Signals)**.** Let $\mathcal{G} = (\mathcal{V}, \mathcal{E}, w)$ denote an undirected, weighted graph:

- $\mathcal{V} = \{1, 2, \ldots, n\}$ is the set of n vertices (nodes).

- $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of edges, representing connections between nodes.
- $w : \mathcal{E} \to \mathbb{R}_+$ is a weight function assigning a nonnegative weight w_{ij} to each edge $(i, j) \in \mathcal{E}$, quantifying the strength of the connection.

We assume that the graph is undirected, meaning that if $(i, j) \in \mathcal{E}$, then $(j, i) \in \mathcal{E}$, and that there are no self-loops (edges from a node to itself) or multiple edges between the same pair of nodes.

A *graph signal* is a function defined on the vertices of the graph, represented as a vector $\mathbf{x} \in \mathbb{R}^n$. Each entry x_i corresponds to the signal value at vertex $i \in \mathcal{V}$. Graph signals generalize classical signals by considering data that is indexed by nodes in a graph rather than time or spatial coordinates.

Difinition 6 (Adjacency and Degree Matrices)**.** The structure of the graph $\mathcal{G}$ can be represented algebraically using matrices:

Adjacency Matrix: The *adjacency matrix* $\mathbf{W} \in \mathbb{R}^{n \times n}$ encodes the connectivity and weights of the graph:

$$W_{ij} = \begin{cases} w_{ij}, & \text{if } (i, j) \in \mathcal{E}, \\ 0, & \text{otherwise.} \end{cases} \tag{2.15}$$

Since the graph is undirected and has no self-loops, the adjacency matrix is symmetric ($W_{ij} = W_{ji}$) and has zeros on the diagonal ($W_{ii} = 0$).

An explicit representation of $\mathbf{W}$ is:

$$\mathbf{W} = \begin{bmatrix} 0 & W_{12} & \cdots & W_{1n} \\ W_{21} & 0 & \cdots & W_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ W_{n1} & W_{n2} & \cdots & 0 \end{bmatrix}. \tag{2.16}$$

Degree Matrix: The *degree matrix* $\mathbf{D} \in \mathbb{R}^{n \times n}$ is a diagonal matrix reflecting the total connection strength of each node:

$$D_{ii} = \sum_{j=1}^{n} W_{ij}, \quad \text{and } D_{ij} = 0 \text{ for } i \neq j. \tag{2.17}$$

This means that each diagonal entry D_{ii} represents the sum of the weights of all edges connected to node i.

An explicit representation of $\mathbf{D}$ is:

$$\mathbf{D} = \operatorname{diag}(D_{11}, D_{22}, \ldots, D_{nn}) = \begin{bmatrix} D_{11} & 0 & \cdots & 0 \\ 0 & D_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & D_{nn} \end{bmatrix}. \tag{2.18}$$

Difinition 7 (Graph Laplacian)**.** The *graph Laplacian* is a central operator in GSP, capturing the interplay between graph structure and signal variation.

Standard Graph Laplacian: The Laplacian matrix $\mathbf{L} \in \mathbb{R}^{n \times n}$ is defined as

$$\mathbf{L} := \mathbf{D} - \mathbf{W}. \tag{2.19}$$

This operator encapsulates the connectivity and weights of the graph, penalizing differences between connected nodes.

Alternatively, the entries of $\mathbf{L}$ can be expressed as

$$L_{ij} = \begin{cases} D_{ii}, & \text{if } i = j, \\ -W_{ij}, & \text{if } (i,j) \in \mathcal{E}, \\ 0, & \text{otherwise.} \end{cases} \tag{2.20}$$

This formulation highlights how each off-diagonal entry accounts for the negative connection between nodes i and j, while diagonal entries represent the "degree" of each node.

An explicit representation of $\mathbf{L}$ is:

$$\mathbf{L} = \begin{bmatrix} D_{11} & -W_{12} & \cdots & -W_{1n} \\ -W_{21} & D_{22} & \cdots & -W_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ -W_{n1} & -W_{n2} & \cdots & D_{nn} \end{bmatrix}. \tag{2.21}$$

The graph Laplacian $\mathbf{L}$ is symmetric ($\mathbf{L} = \mathbf{L}^\top$) and positive semidefinite, meaning that for any $\mathbf{x} \in \mathbb{R}^n$,

$$\mathbf{x}^\top \mathbf{L} \mathbf{x} \geq 0. \tag{2.22}$$

This property is essential in defining notions of smoothness and variation of graph signals.

2.3.1 Graph Signal Smoothness

A key concept in GSP is the smoothness of a graph signal with respect to the underlying graph structure.

A graph signal $\mathbf{x} \in \mathbb{R}^n$ is considered *smooth* if neighboring nodes (connected by edges with large weights) have similar signal values. In other words, the signal exhibits minimal variation across edges with strong connections.

Graph Laplacian Quadratic Form: The smoothness of a graph signal can be quantitatively measured using the graph Laplacian quadratic form:

$$\mathbf{x}^\top \mathbf{L} \mathbf{x} = \sum_{i=1}^{n} \sum_{j=1}^{n} W_{ij}(x_i - x_j)^2 = \sum_{(i,j) \in \mathcal{E}} W_{ij}(x_i - x_j)^2. \tag{2.23}$$

This expression sums the squared differences between signal values at connected nodes, weighted by the edge weights. A smaller value indicates a smoother signal over the graph.

Total Smoothness for Multiple Signals: For a collection of m graph signals, represented as columns of a matrix $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m] \in \mathbb{R}^{n \times m}$, the total smoothness is given by

$$\sum_{t=1}^{m} \mathbf{x}_t^\top \mathbf{L} \mathbf{x}_t = \operatorname{Tr}\left(\mathbf{X}^\top \mathbf{L} \mathbf{X}\right). \tag{2.24}$$

Here, $\operatorname{Tr}(\cdot)$ denotes the trace of a matrix. This expression measures the aggregate smoothness across all signals in $\mathbf{X}$.

Application in Signal Recovery: In optimization-based signal recovery, promoting signal smoothness can improve reconstruction from noisy or incomplete observations. For example, recovering a time-varying graph signal $\overline{\mathbf{X}}$ from observations $\mathbf{Y} = \Phi(\overline{\mathbf{X}}) + \mathbf{N}$, where $\mathbf{N}$ represents additive noise and Φ is a masking operator, can be formulated as the optimization problem

$$\min_{\mathbf{X}\in\mathbb{R}^{n\times m}} \operatorname{Tr}\left(\mathbf{X}^\top \mathbf{L}\mathbf{X}\right) + \frac{\alpha}{2}\left\|\mathbf{X}-\mathbf{Y}\right\|_F^2, \tag{2.25}$$

where $\alpha > 0$ is a regularization parameter balancing the smoothness term and data fidelity.

Remark 2.3.1. The optimization problem (2.25) aims to find a signal $\mathbf{X}$ that is both smooth over the graph (minimizing $\operatorname{Tr}(\mathbf{X}^\top \mathbf{L}\mathbf{X})$) and close to the observed data $\mathbf{Y}$ (minimizing $\|\mathbf{X}-\mathbf{Y}\|_F^2$). The parameter α controls the trade-off between these two objectives.

Example 3 (Graph Total Variation)**.** An alternative measure of signal variation is the *graph total variation* (GTV), defined as

$$\mathrm{TV}_G(\mathbf{x}) = \sum_{(i,j)\in\mathcal{E}} W_{ij}|x_i - x_j|. \tag{2.26}$$

Unlike the Laplacian quadratic form, which uses squared differences, the GTV uses absolute differences, making it more robust to abrupt changes or discontinuities in the signal.

The GTV promotes piecewise smoothness and is particularly useful in applications where preserving edges or sharp transitions is important, such as image denoising on graphs or community detection.

Chapter 3

Robust Time-Varying Graph Signal Recovery for Dynamic Physical Sensor Network Data

3.1 Introduction

The concept of a *graph signal*, defined by signal values observed on the vertex set $\mathcal{V}$ of a graph $\mathcal{G}$, has been intensely researched as an effective approach to representing irregularly structured data. Conventional signal processing is based on spatially or temporally regular structures (e.g., images and sounds), and thus, the relations between signal values are also regular, which provides no further information for us to leverage. On the other hand, graph signal representations and graph signal processing [69, 87, 92] explicitly represent relations between signal values with vertices and weighted edges, which we can exploit as priors in the vertex domain. Irregularly structured data such as traffic and sensor network data, geographical data, mesh data, and biomedical data all benefit from such representation.

Following the history of conventional signal processing, recovering the true graph signals from often corrupted observations is a necessity in processing the data for further use. Many algorithms have been proposed to address the problem, most methods based on either the vertex [10, 19, 29, 31, 77] or the graph spectral domain [20, 66, 78]. These methods are proposed to exploit the smoothness of the signals on the vertex domain. Graph signals are smooth when the signal values of two vertices connected by edges with large weights are similar. Such traits can often be observed on various real-world graph signals, such as temperature data observed on a network of sensors where the edges represent the physical distances of the sensors. Sensors that are close by would be connected by edges with large weights and would also most likely observe similar temperatures, which gives us a smooth graph signal.

Although leveraging the graph signal smoothness can successfully recover the true signals, these methods ignore the temporal domain. In real-life scenarios, many of the above-mentioned data can easily be sampled continuously to form time-coherent data, and therefore, priors based on the temporal domain should be effectively utilized to improve recovery results.

3.1.1 Related Work

Several time-varying graph signal recovery methods have been discussed to leverage signal smoothness priors in the temporal domain together with that of the vertex domain.

The earlier studies involve filtering and the Fourier transform of product graphs [88]. The more recent studies include spectral-based approaches like Joint time-vertex Fourier

Table 3.1: The Feature of Time-Varying Graph Signal Recovery Methods

Method	Spatial correlations	Temporal correlations		Sparse outliers	Topology	Temporal signal regularization	Data fidelity	Algorithm
		Graph	Signal					
JFT [39]	◯	×	◯	×	Known	(Spectral Filtering)	(Spectral Filtering)	Off-line
TGSR [85]	◯	×	◯	×	Known	Quadratic	Cost Function	Off-line
GTRSS [38]	◯	×	◯	×	Known	Quadratic	Cost Function	Off-line
LRGTS [57]	◯	×	◯	×	Known	Quadratic	Constraint	Off-line
DLSRA [44]	◯	×	◯	×	Known	Quadratic	Cost Function	On-line
DAMRA [44]	◯	×	◯	×	Known	ℓ_1	Cost Function	On-line
OLTVSG [89]	◯	△	△	×	Unknown	Quadratic	Cost Function	On-line
KKF [86]	◯	◯	◯	×	Known	Quadratic	Cost Function	On-line
Proposed	◯	◯	◯	◯	Known	Quadratic or ℓ_1	Constraint	Off-line

transform (JFT) [39,60,81] and vertex-domain-based approaches that leverage smoothness on the vertex and temporal domains [38, 49, 64, 85]. JFT jointly applies the Fourier transform to both the temporal direction and vertex direction to fully leverage the priors of the two domains. In [49, 85], the formulation is based on leveraging the smoothness of the temporal difference signal on the underlying graph, which effectively utilizes both domains for the signal recovery. This is generalized in [38] by leveraging the Sobolev smoothness of the time-varying graph signals. Low-rank-based methods [57] further enhance recovery performance by leveraging the low-rankness of the time-varying graph signals. Furthermore, distributed algorithms [44] have been proposed to avoid costly computations like eigenvalue decomposition and matrix inversion. Other than optimization-based methods, graph-neural-network-based methods [16, 17] have also been proposed as a more data-driven approach to time-varying graph signal recovery.

Generally speaking, time-varying graph signal recovery methods are often discussed under the assumption that the underlying graph is static, especially in cases of physical sensing. This is because, in practical applications, the graph (the weights of the edges) is pre-defined heuristically, usually by a Gaussian kernel of spatial coordinates. Whether the graph represents a geographical network, or a cranial nerve system, or any other physical network, as long as the physical coordinates of the vertices are static, the graph is also static. Therefore, very little work has been done in the area of time-varying graph signal recovery on a dynamic graph topology.

However, in the context of graph learning [35, 47, 84], where graphs are learned from the graph signals, it is much more natural to expect the graphs that house time-varying graph signals to be dynamic, just like the signals [103]. Under the assumption that dynamic graphs are better representations of the underlying structures that time-varying graph signals are observed on, some recovery methods have been proposed to exploit the dynamics of the graphs [86, 89].

The authors in [89] proposed a method to estimate time-varying graph signals while also estimating dynamic graphs, which are unknown, based on an online strategy to keep track of the temporal variation of both the graph and the graph signal. In [86], a space-time kernel is proposed as an extension from kernel-based methods on static graphs [36,98,99] to leverage both domains on a graph extended in the vertex domain to represent the temporal domain. Although these methods allow an online estimation of the graph signals, they force the following trade-offs to their formulations in order to consider online estimation: The inability to capture the global temporal correlations in [89] and the limitation to quadratic costs to adopt a Kalman filtering strategy in [86].

Outside of signal recovery, multiplex and multilayer graph signal processing have also been considered [107, 108] and applied in tasks such as image segmentation and graph learning. In multiplex and multilayer graph signal processing, the relation between the signals in the temporal domain is represented by edges and weights.

To summarize, most time-varying graph signal recovery methods assume static graphs

and the few that do consider dynamic graphs focus on online estimation and graph learning based on the assumption that all of the dynamic graphs are not available.

3.1.2 Contributions and Chapter Organization

In this chapter, we focus on scenarios where the entire dynamically changing graph structure can be fully exploited. Conventional studies on static time-varying graph signal recovery typically assume that the graph is static and available. However, situations where the complete dynamic graph structure is available have not been extensively studied.

With recent advances in sensor technology, dynamic sensor networks—such as those constructed from sensor-equipped drones and smartphones—are becoming increasingly relevant in real-world applications. Examples include: a) smart agriculture [68], where drones and IoT sensors monitor crop health, soil moisture, and weather conditions to optimize resource use and increase productivity; b) traffic and infrastructure management [12], where sensor-equipped vehicles and drones are used in smart cities for real-time traffic management and monitoring of road conditions; and c) environmental monitoring [45], where drones equipped with sensors are employed to monitor air quality, assess pollution levels, and track environmental changes. In these and other applications that handle time-varying data from physical sensor networks, appropriate dynamic graphs can often be generated using simple algorithms, such as k-nearest neighbors, by leveraging the time-varying spatial locations of the sensors, similar to the static problem settings.

We formulate dynamic graph signal recovery as a constrained convex optimization problem that simultaneously estimates both time-varying graph signals and sparsely modeled outliers. The formulation involves time-varying graph-Laplacian-based and temporal-difference-based regularizations that leverage the signal smoothness in both vertex and temporal domains. Although sparse outliers are rarely addressed in time-varying graph signal recovery, we consider it to be a realistic problem since physical sensors can easily be affected by local environmental factors, which can lead to outliers. We also restore missing values, which can represent sensor malfunctions or maintenance in real-world settings, to achieve robust estimation from highly degraded data. Data fidelity and outlier sparsity are imposed as hard constraints rather than as a part of the cost function to facilitate parameter tuning by decoupling the parameters, as has been addressed, e.g., in [2, 23, 73, 76]. A primal-dual splitting (PDS) [24, 100] method-based algorithm is developed to efficiently solve the optimization problem.

Table 3.1 is a simple representation of where our method lies in relation to the conventional methods. Although online estimation is a crucial task in some real-world applications, the global information of the signals that are disregarded for online estimation should be effectively utilized to enhance the graph signal recovery performance.

The main contributions of this study are as follows.

- We tackle an unaddressed problem setting, that is, recovering time-varying graph signals from noisy incomplete observations, possibly with outliers, when the underlying graph topology is dynamic.
- We establish an optimization-based recovery method that can handle both time-varying graph-Laplacian-based and temporal-difference-based regularizations.
- We design a framework for experiments on time-varying graph signal recovery over dynamic graphs based on synthetic data, which simulates signals observed using a network of sensor-loaded drones.
- We conduct extensive experiments not only to validate the proposed method but also for a comprehensive study of the performance of various regularization terms over both synthetic and real-world data.

We first the proposed method in Section 3.2 and then illustrate the experimental results both quantitively and visually and discuss the obtained results in Section 3.3.

3.2 Proposed Method

3.2.1 Problem Formulation

Consider the following graph signal observation model:

$$\mathbf{Y} = \Phi(\overline{\mathbf{X}} + \overline{\mathbf{S}} + \mathbf{N}), \tag{3.1}$$

where $\mathbf{Y} = [\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_p] \in \mathbb{R}^{n \times p}$ is the observation of an n vertices $\times$ p time-slots time-varying graph signal, and $\overline{\mathbf{X}} \in \mathbb{R}^{n \times p}$ is the true signal. For the noises, $\overline{\mathbf{S}} \in \mathbb{R}^{n \times p}$ and $\mathbf{N} \in \mathbb{R}^{n \times p}$ are sparse outliers and some random additive noise, respectively. Missing values are represented by a masking operator Φ, where $\Phi(\mathbf{X})$ projects the elements of $\mathbf{X}$ to 0 at a given probability (random vertices are masked every time-slot). Each graph signal $\mathbf{y}_k$ is observed on a graph corresponding to the kth graph Laplacian $\mathbf{L}_k$.

We propose the following optimization problem to estimate the true graph signal in the above model, under the assumption that the time-varying graph signals are smooth in the vertex domain, and that the outlier $\overline{\mathbf{S}}$ is sparse:

$$\begin{aligned} \min_{\mathbf{X} \in \mathbb{R}^{n \times p}, \mathbf{S} \in \mathbb{R}^{n \times p}} & \sum_{k=1}^{p} R_v^{(k)}(\mathrm{P}(\mathbf{X})) + \lambda R_t(\mathrm{D}(\mathbf{X})) \\ \text{s.t. } & \|\mathbf{Y} - \Phi(\mathbf{X} + \mathbf{S})\|_F \leq \varepsilon, \ \|\mathbf{S}\|_1 \leq \eta, \end{aligned} \tag{3.2}$$

where R_v and R_t indicate the graph-Laplacian-based and temporal-difference-based regularizations, respectively. Also, $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_p] \in \mathbb{R}^{n \times p}$ is the estimated time-varying graph signal, $\mathrm{P}(\mathbf{X}) := \mathbf{X}$or $\mathrm{D}(\mathbf{X})$, $\mathbf{S} \in \mathbb{R}^{n \times p}$ is the matrix of estimated outliers, $\mathbf{L}_k \in \mathbb{R}^{n \times n}$ is the graph Laplacian at time k, and $\lambda > 0$ is the temporal regularization parameter. The ℓ_1-norm of a matrix is denoted by $\|\cdot\|_1$, and ε and η are the radii of the Frobenius and ℓ_1-norm balls, respectively. The temporal difference operator D is defined by

$$\mathrm{D}(\mathbf{X}) = [\mathbf{x}_2 - \mathbf{x}_1, \mathbf{x}_3 - \mathbf{x}_2, ..., \mathbf{x}_p - \mathbf{x}_{p-1}]. \tag{3.3}$$

As for the first regularization term, which enforces the signal smoothness in the vertex domain, $R_v^{(k)}(\mathrm{P}(\mathbf{X}))$ is defined as

$$R_v^{(k)}(\mathrm{P}(\mathbf{X})) := \mathrm{P}(\mathbf{X})_k^\top \mathbf{L}_k \, \mathrm{P}(\mathbf{X})_k. \tag{3.4}$$

The above equation generalizes the following two regularizations:

$$R_v^{(k)}(\mathbf{X}) = \mathbf{x}_k^\top \mathbf{L}_k \mathbf{x}_k, \tag{3.5}$$

$$R_v^{(k)}(\mathrm{D}(\mathbf{X})) = (\mathbf{x}_{k+1} - \mathbf{x}_k)^\top \mathbf{L}_k (\mathbf{x}_{k+1} - \mathbf{x}_k). \tag{3.6}$$

Graph signal smoothness is leveraged by penalizing the above two graph Laplacian quadratic forms. Unlike many conventional works [21, 85] that measure the smoothness of the time-varying graph signal on a single graph (refer Eq. (2.24)), the proposed regularization measures the signal smoothness per time-slot ($\sum_{k=1}^{p} R_v^{(k)}(\mathrm{P}(\mathbf{X}))$). Such time-varying graph-Laplacian-based regularization allows us to integrate the dynamics of the vertex domain into the formulation. An alternative regularization $R_v^{(k)}(\mathrm{D}(\mathbf{X}))$ leverages the smoothness of the temporal difference signal on the vertex domain, proposed in [85] under the assumption that real-world time-varying graph signals are not as smooth over the graph as expected. On the contrary to [85], we consider a dynamic graph, which can question the

smoothness of $\mathbf{x}_{k+1} - \mathbf{x}_k$ on $\mathbf{L}_k$[1]. However, assuming that the graph changes smoothly over time, the difference between $\mathbf{L}_{k+1}$ and $\mathbf{L}_k$ would be small, and the regularization would be effective.

The main points of the proposed method are as follows:

- The time-varying graph-Laplacian-based regularization uses a different graph Laplacian for every time-slot, unlike the conventional static-based methods.
- We allow the choice between two types of graph-Laplacian-based regularizations.

The temporal-difference-based regularization $R_t(\mathrm{D}(\mathbf{X}))$ penalizes the temporal gradient of the graph signal $\mathbf{X}$ to leverage the signal smoothness in the temporal domain. We consider $R_t(\cdot)$ to be either $\|\cdot\|_1$ or $\|\cdot\|_F^2$ depending on the nature of the signal of interest.

We impose data fidelity and noise sparsity as hard constraints in (3.2) to facilitate parameter tuning by decoupling the parameters, as has been addressed, e.g., in [2, 23, 73, 76].

The main philosophy behind our formulation is that the underlying graph is often available in many real-world graph signal recovery situations, in which case our method would perform advantageously compared to the conventional methods [86, 89] that consider graph learning and online estimations. Especially in physical sensing situations, (e.g. using a dynamic network of sensor-loaded drones for physical sensing) the underlying dynamic graph topology can easily be generated per time-slot using the k-nearest neighbor algorithm and weights defined by the Gaussian kernel of spatial coordinates. Therefore, the conditions where our method can recover the graph signals well are when the target distribution that is being sampled is spatially and temporally smooth. The spatial smoothness is necessary for the graph constructed by the k-nearest neighbor algorithm to be a decent substitute for an ideal graph. The temporal smoothness is leveraged by $R_t(\cdot)$.

In cases where no dynamic graphs are available in any way (when $\mathbf{L}_1 = \mathbf{L}_2 = ... = \mathbf{L}_k$), our formulation would be the equivalent of the following formulation:

$$\begin{aligned} &\min_{\mathbf{X}\in\mathbb{R}^{n\times p},\mathbf{S}\in\mathbb{R}^{n\times p}} R_v(\mathrm{P}(\mathbf{X})) + \lambda R_t(\mathrm{D}(\mathbf{X})) \\ &\text{s.t. } \|\mathbf{Y} - \Phi(\mathbf{X}+\mathbf{S})\|_F \leq \varepsilon,\ \|\mathbf{S}\|_1 \leq \eta, \end{aligned} \tag{3.7}$$

where $R_v(\mathrm{P}(\mathbf{X})) := \mathrm{Tr}\,(\mathrm{P}(\mathbf{X})^\top \mathbf{L}\,\mathrm{P}(\mathbf{X}))$. Note that this is still a novel formulation for time-varying graph signal recovery on static graphs in terms of temporal-difference-based regularization and sparse noise estimation.

3.2.2 Algorithm

We use the primal-dual splitting method (PDS) [24] to solve (3.2). By vectorizing $\mathbf{Y}$ and using indicator functions $\iota_{B_{\ell 2}^{(\mathbf{y},\varepsilon)}}$ and $\iota_{B_{\ell 1}^{(\eta)}}$ of

$$B_{\ell 2}^{(\mathbf{y},\varepsilon)} := \{\mathbf{z}\in\mathbb{R}^{np} \mid \|\mathbf{z}-\mathbf{y}\|_2 \leq \varepsilon\}, \tag{3.8}$$

$$B_{\ell 1}^{(\eta)} := \{\mathbf{z}\in\mathbb{R}^{np} \mid \|\mathbf{z}\|_1 \leq \eta\}, \tag{3.9}$$

we can reformulate (3.2) as the following equivalent problem:

$$\begin{aligned} &\min_{\mathbf{X}\in\mathbb{R}^{n\times p},\mathbf{S}\in\mathbb{R}^{n\times p}} R_v(\mathrm{vec}(\mathrm{P}(\mathbf{X}))) + \lambda R_t(\mathbf{D}\,\mathrm{vec}\,(\mathbf{X})) \\ &+\iota_{B_{\ell 2}^{(\mathrm{vec}\,(\mathbf{Y}),\varepsilon)}}(\mathbf{\Phi}(\mathrm{vec}\,(\mathbf{X}) + \mathrm{vec}\,(\mathbf{S}))) + \iota_{B_{\ell 1}^{(\eta)}}(\mathrm{vec}\,(\mathbf{S})), \end{aligned} \tag{3.10}$$

[1] It should be also noted that although we are calculating $\mathbf{x}_{k+1} - \mathbf{x}_k$ on $\mathbf{L}_k$, there is no essential difference to calculating $\mathbf{x}_k - \mathbf{x}_{k-1}$ on $\mathbf{L}_k$ instead. If formulation applicable to online settings is desired, the latter can be used.

where $R_v(\mathbf{x}) = \mathbf{x}^\top \mathbf{L}\mathbf{x} = \mathbf{x}^\top \mathrm{diag}(\mathbf{L}_1, \cdots, \mathbf{L}_p)\mathbf{x}$.

The matrix $\mathbf{D} \in \mathbb{R}^{n(p-1)\times np}$ and diagonal matrix $\mathbf{\Phi} \in \mathbb{R}^{np\times np}$ are the temporal linear difference operator and a masking operator for vectorized variables, respectively.

By defining $\mathbf{u} := [\mathrm{vec}\,(\mathbf{X})^\top \mathrm{vec}\,(\mathbf{S})^\top]^\top$, and $\mathbf{v} := [\mathbf{v}_1^\top \ \mathbf{v}_2^\top]^\top$ ($\mathbf{v}_1 \in \mathbb{R}^{n(p-1)}, \mathbf{v}_2 \in \mathbb{R}^{np}$), and $f_1, f_2, f_3, \mathbf{A}$ as

$$\begin{aligned} f_1(\mathbf{u}) &:= R_v(\mathrm{vec}(\mathrm{P}(\mathbf{X}))) \\ &= R_v \circ \mathrm{vec} \circ \mathrm{P}(\mathbf{X}), \\ f_2(\mathbf{u}) &:= \iota_{B_{\ell 1}^{(\eta)}}(\mathrm{vec}\,(\mathbf{S})), \\ f_3(\mathbf{v}) &:= \lambda R_t(\mathbf{v}_1) + \iota_{B_{\ell 2}^{(\mathrm{vec}\,(\mathbf{Y}),\varepsilon)}}(\mathbf{v}_2), \\ \mathbf{A} &:= \begin{bmatrix} \mathbf{D} & \mathbf{0} \\ \mathbf{\Phi} & \mathbf{\Phi} \end{bmatrix}, \end{aligned} \tag{3.11}$$

since $f_1(\mathbf{u})$ is differentiable and the gradient is Lipschitz continuous and $\iota_{B_{\ell 1}^{(\eta)}}(\mathrm{vec}\,(\mathbf{S}))$ and $\lambda R_t(\mathbf{v}_1) + \iota_{B_{\ell 2}^{(\mathrm{vec}\,(\mathbf{Y}),\varepsilon)}}(\mathbf{v}_2)$ are proper lower-semicontinuous convex functions, the problem in (3.10) is reduced to (2.12), which can be solved by PDS (refer to Algorithm 1). Note that although the algorithm is written in the vectorized form, the actual implementation is written in the matrix form.

The proximity operators of f_2 and f_3 can be computed by the proximity operators of $\iota_{B_{\ell 1}^{(\eta)}}$, $\lambda R_t(\mathbf{v}_1)$ and $\iota_{B_{\ell 2}^{(\mathrm{vec}\,(\mathbf{Y}),\varepsilon)}}$.

The proximity operator of f_2 is given by

$$\mathrm{prox}_{\gamma \iota_{B_{\ell 1}^{(\eta)}}}(\mathbf{z}) = P_{B_{\ell 1}^{(\eta)}}(\mathbf{z}) = \mathbf{z} - \mathrm{prox}_{\eta\|\cdot\|_\infty}(\mathbf{z}) \tag{3.12}$$

$$= \mathbf{z} - [\mathrm{sgn}(z_j)\min\{|z_j|, s\}]_{1\le j\le J}, \tag{3.13}$$

where $[x_j]_{1\le j\le J}$ is a vector consisting x_j as the jth element and $s \in \mathbb{R}$ is such that $\sum_{j=1}^{J} \max\{0, |z_j| - s\} = \gamma$ (Eq. (6) in [26]).

For f_3, when R_t is $\|\cdot\|_1$, the proximity operator is equivalent to the soft-thresholding operation:

$$[\mathrm{prox}_{\gamma\|\cdot\|_1}(\mathbf{z})]_j := \mathrm{sgn}\,(z_j)\max\{0, |z_j| - \gamma\}, \tag{3.14}$$

and when R_t is $\|\cdot\|_F^2$, the proximity operator of f_3 is

$$\mathrm{prox}_{\gamma\|\cdot\|_2^2}(\mathbf{z}) = \frac{\mathbf{z}}{2\gamma + 1}. \tag{3.15}$$

The proximity operator of $\iota_{B_{\ell 2}^{(\mathbf{y},\varepsilon)}}$ is given by

$$\mathrm{prox}_{\gamma \iota_{B_{\ell 2}^{(\mathbf{y},\varepsilon)}}}(\mathbf{z}) = P_{B_{\ell 2}^{(\mathbf{y},\varepsilon)}}(\mathbf{z}) \tag{3.16}$$

$$= \begin{cases} \mathbf{z}, & \text{if } \mathbf{z} \in B_{\ell 2}^{(\mathbf{y},\varepsilon)}, \\ \mathbf{y} + \frac{\varepsilon(\mathbf{z}-\mathbf{y})}{\|\mathbf{z}-\mathbf{y}\|_2}, & \text{otherwise.} \end{cases} \tag{3.17}$$

3.2.3 Computational Complexity

In this section, we discuss the computational complexity per iteration of the proposed algorithm. The operations in our PDS-based algorithm (Algorithm 1) that potentially require complex computations are the three proximity operations and $\nabla R_v^{(k)}$ involved in line 2 of the algorithm. The two proximity operations in lines 6 and 7 can be computed in the order of $O(np)$. For the calculation of the ℓ_1-norm ball in line 3, it requires an

Algorithm 1 Algorithm for solving (3.2)

Input: Input signal $\mathbf{Y}$, graph Laplacian $\mathbf{L}_k (k = 1, 2, ..., p)$
Output: Output signal $\mathbf{X}^{(i)}$

Initialization: $\mathbf{X}^{(0)} = \mathbf{Y} \in \mathbb{R}^{n \times p}, \mathbf{S}^{(0)} = \mathbf{0}$
while *A stopping criterion is not satisfied* **do**
  $\mathrm{vec}\,(\mathbf{X})^{(i+1)} = \mathrm{vec}\,(\mathbf{X})^{(i)} - \gamma_1(\nabla(R_v \circ \mathrm{vec} \circ \mathrm{P})(\mathbf{X}) + \mathbf{D}^\top \mathbf{v}_1^{(i)} + \mathbf{\Phi}^\top \mathbf{v}_2^{(i)})$;
  $\mathrm{vec}\,(\mathbf{S})^{(i+1)} = P_{B_{\ell 1}^{(\eta)}}(\mathrm{vec}\,(\mathbf{S})^{(i)} - \gamma_1 \mathbf{\Phi}^\top \mathbf{v}_2^{(i)})$;
  $\mathbf{v}_1^{(i)} \leftarrow \mathbf{v}_1^{(i)} + \gamma_2 \mathbf{D}(2\,\mathrm{vec}\,(\mathbf{X})^{(i+1)} - \mathrm{vec}\,(\mathbf{X})^{(i)})$;
  $\mathbf{v}_2^{(i)} \leftarrow \mathbf{v}_2^{(i)} + \gamma_2 \mathbf{\Phi}((2\,\mathrm{vec}\,(\mathbf{X})^{(i+1)} - \mathrm{vec}\,(\mathbf{X})^{(i)}) + (2\,\mathrm{vec}\,(\mathbf{S})^{(i+1)} - \mathrm{vec}\,(\mathbf{S})^{(i)}))$;
  $\mathbf{v}_1^{(i+1)} = \mathbf{v}_1^{(i)} - \gamma_2 \mathrm{prox}_{\frac{\lambda}{\gamma_2} R_t}(\frac{1}{\gamma_2}\mathbf{v}_1^{(i)})$;
  $\mathbf{v}_2^{(i+1)} = \mathbf{v}_2^{(i)} - \gamma_2 P_{B_{\ell 2}^{(\mathrm{vec}\,(\mathbf{Y}),\varepsilon)}}(\frac{1}{\gamma_2}\mathbf{v}_2^{(i)})$;
  $i \leftarrow i + 1$;
end whilereturn $\mathbf{X}^{(i)}$

order of $O(np \log np)$. For $\nabla R_v^{(k)}$, the calculation of the gradient has to be performed for all time-slots, therefore, requires a computational complexity of $O(n^2 p)$. However, this is dependent on how dense the graph Laplacian is. If it is sufficiently sparse, like in our case where k-nearest neighbor ($k = 4$) is applied to construct the graph, the order is reduced to $O(np)$. Therefore, the overall computational complexity of our algorithm is $O(np \log np)$ per iteration.

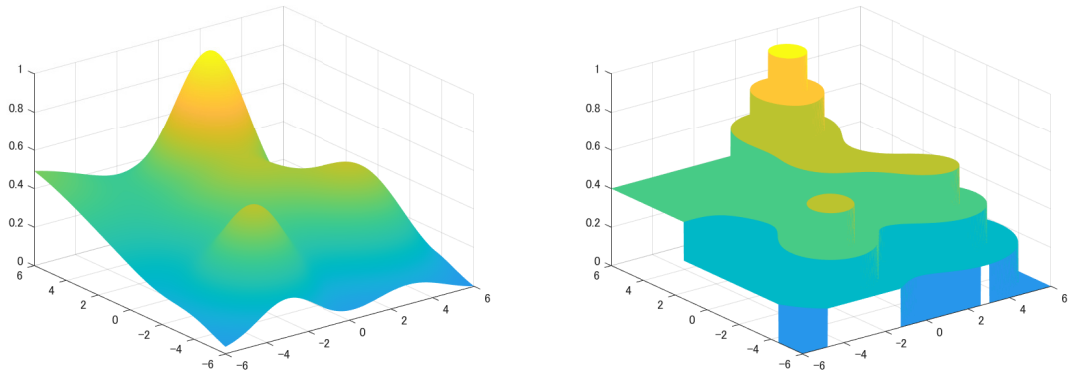
3.3 Experimental Results

3.3.1 Dataset

Synthetic Dataset

To test the methods on a dynamic graph Laplacian setting, we constructed a synthetic dataset that simulates a network of sensor-loaded drones remotely sensing a smooth landscape. In a 2D plane, 128 vertices, which represent drones, are generated randomly from a uniform distribution. They observe signal values calculated by inputting their coordinates to the underlying smooth distribution ($[0, 1]$), in this case, a combination of several multivariate normal distributions (Fig. 3.1 (a)). The vertices move across the 2D plane at random degrees and a pre-set velocity v for 100 time-slots to generate a time-varying graph signal $\overline{\mathbf{X}} \in \mathbb{R}^{128 \times 100}$. A dynamic graph Laplacian $\mathbf{L} \in \mathbb{R}^{128 \times 128 \times 100}$ is then constructed by applying the k-nearest neighbors algorithm ($k = 4$) to the time-varying graph signal $\overline{\mathbf{X}}$. The weights are decided by the Gaussian kernel of the spatial coordinates of the corresponding two vertices, where the variance in the Gaussian kernel is decided from the average nearest neighbor distances.

Then, $\overline{\mathbf{X}}$ is corrupted by adding $\overline{\mathbf{S}}$ and $\mathbf{N}_\sigma$ and being masked by Φ to generate an observation $\mathbf{Y}$. Outlier $\overline{\mathbf{S}}$ is a matrix of impulsive noise that appears at each vertex at a probability of P_s, where each value of the noise is uniformly distributed in the interval $[-1, 1]$. Noise $\mathbf{N}_\sigma$ is an additive white Gaussian noise of variance σ^2, and Φ masks the signals at a probability of P_p (e.g., the sampling rate is 80% when $P_p = 0.2$). The graph construction is implemented by using GSPBox [82]. We also constructed a piece-wise flat version of the dataset (Fig. 3.1 (b)) by rounding the signal values of Fig. 3.1 (a) to the nearest fifth of the signal range.



(a) (b)

Figure 3.1: (a): The smooth distribution constructed by a combination of several multivariate normal distributions. (b): A piece-wise flat version of (a).

Real-world Dataset

We use the time-varying sea surface temperature[2] for our real-world data experiments. The provided dataset is comprised of sea surface temperature data collected weekly at a spatial resolution of 1° latitude ×1° longitude. We first extract two subsets of the dataset, which originally covers a large part of the Earth's surface. The first subset represents the South Pacific Ocean, spanning 170° west to 90° west and 60° south to 10° north. The second subset represents the North Atlantic Ocean, spanning 60° west to 20° west and 50° north to 70° south. Then, we randomly select 100 and 64 initiate points from the respective cut-out data and simulate a dynamic graph by sequentially shifting these points to adjacent positions at each timeslot. Using this method, we create two datasets (100 nodes × 600 time-slots and 64 nodes × 300 time-slots, respectively) similar to the synthetic dataset where the underlying graph is dynamic. The graphs at each time-slots are constructed and the graph signals are corrupted in the same manner as the synthetic dataset. Note that unlike the synthetic dataset, where the underlying distribution is static, the sea temperature is also time-varying. The temperature data is scaled to [0 1].

3.3.2 Experimental Settings

In the experiments,we evaluate our method on synthetic and real-world datasets. Regarding the implementation for Algorithm 1, the stopping criterion is set to $\frac{\|\mathbf{X}^{(i)}-\mathbf{X}^{(i-1)}\|_2}{\|\mathbf{X}^{(i-1)}\|_2} \leq 1.0 \times 10^{-4} \wedge \frac{\|\mathbf{S}^{(i)}-\mathbf{S}^{(i-1)}\|_2}{\|\mathbf{S}^{(i-1)}\|_2} \leq 1.0 \times 10^{-4}$. The performance of the methods is measured in RMSE and MAE:

$$\text{RMSE} = \sqrt{\frac{1}{n \times p} \sum_{i=1}^{n} \sum_{j=1}^{p} (\overline{\mathbf{X}}(i,j) - \mathbf{X}(i,j))^2}, \tag{3.18}$$

$$\text{MAE} = \frac{1}{n \times p} \sum_{i=1}^{n} \sum_{j=1}^{p} |\overline{\mathbf{X}}(i,j) - \mathbf{X}(i,j)|. \tag{3.19}$$

[2]NOAA Physical Sciences Laboratory provides the weekly global sea temperature data from their website at https://psl.noaa.gov/data/gridded/data.noaa.oisst.v2.html.

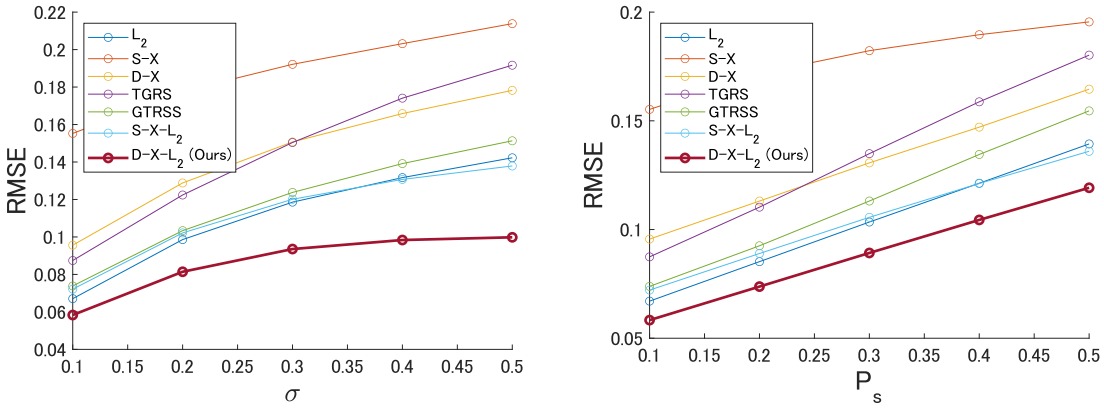
Table 3.2: The List of Methods Used in The Experiments. THe Methods That Leverage Static Graphs Are in Blue And Methods That Leverage Dynamic Graphs Are in Red.

Method	Vertex domain	Temporal domain
L_1	-	$\|\mathrm{D}(\mathbf{X})\|_1$
L_2	-	$\|\mathrm{D}(\mathbf{X})\|_F^2$
S-TV [10]	$\mathrm{TV}_G(\mathbf{X})$	-
S-X [21]	$R_v(\mathbf{X})$	-
D-X	$\sum_{k=1}^{p} R_v^{(k)}(\mathbf{X})$	-
TGSR [85]	$R_v(\mathrm{D}(\mathbf{X}))$	-
GTRSS [38]	$\mathrm{Tr}(\mathrm{D}(\mathbf{X})^\top(\mathbf{L}+\epsilon\mathbf{I})^\beta \mathrm{D}(\mathbf{X}))$	
S-DX-L_1	$R_v(\mathrm{D}(\mathbf{X}))$	$\|\mathrm{D}(\mathbf{X})\|_1$
D-DX-L_1	$\sum_{k=1}^{p-1} R_v^{(k)}(\mathrm{D}(\mathbf{X}))$	$\|\mathrm{D}(\mathbf{X})\|_1$
S-DX-L_2	$R_v(\mathrm{D}(\mathbf{X}))$	$\|\mathrm{D}(\mathbf{X})\|_F^2$
D-DX-L_2	$\sum_{k=1}^{p-1} R_v^{(k)}(\mathrm{D}(\mathbf{X}))$	$\|\mathrm{D}(\mathbf{X})\|_F^2$
S-X-L_1	$R_v(\mathbf{X})$	$\|\mathrm{D}(\mathbf{X})\|_1$
D-X-L_1	$\sum_{k=1}^{p} R_v^{(k)}(\mathbf{X})$	$\|\mathrm{D}(\mathbf{X})\|_1$
S-X-L_2	$R_v(\mathbf{X})$	$\|\mathrm{D}(\mathbf{X})\|_F^2$
D-X-L_2	$\sum_{k=1}^{p} R_v^{(k)}(\mathbf{X})$	$\|\mathrm{D}(\mathbf{X})\|_F^2$

Table 3.3: The Average Recovery Performance on The Various Datasets Measured in RMSE And MAE. The Noise Level Is Set to $\sigma = 0.1, P_s = 0.1, P_p = 0.1$. Sensor Velocity Is Set as $v = 0.25$ for The Synthetic Datasets. Methods That Leverage Static Graphs Are in Blue And Methods That Leverage Dynamic Graphs Are in Red.

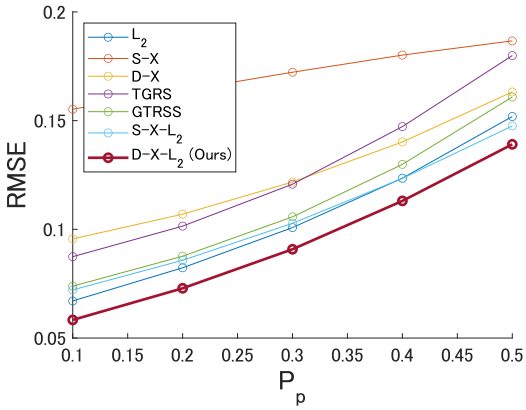
Method	Syn. (smooth)		Syn. (piece-wise flat)		SST Pacific		SST Atlantic	
	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE
L_1	0.0582	0.0446	0.0614	0.0436	0.0643	0.0487	0.0796	0.0635
L_2	0.0507	0.0373	0.0587	0.0445	0.0516	0.0394	0.0668	0.0529
S-TV [10]	0.2791	0.1949	0.2807	0.1947	0.3691	0.2390	0.3486	0.2307
S-X [21]	0.1443	0.1036	0.1539	0.1104	0.1763	0.1249	0.1128	0.0846
D-X	0.0956	0.0704	0.1035	0.0772	0.0826	0.0628	0.0729	0.0564
TGSR [85]	0.0727	0.0532	0.0780	0.0588	0.0761	0.0576	0.0802	0.0618
GTRSS [38]	0.0586	0.0424	0.0652	0.0492	0.0571	0.0440	0.0667	0.0521
S-DX-L_1	0.0530	0.0390	0.0590	0.0431	0.0538	0.0408	0.0680	0.0531
D-DX-L_1	0.0549	0.0404	0.0601	0.0436	0.0527	0.0405	0.0681	0.0533
S-DX-L_2	0.0494	0.0372	0.0579	0.0440	0.0511	0.0389	0.0643	0.0504
D-DX-L_2	0.0497	0.0374	0.0579	0.0440	0.0502	0.0388	0.0650	0.0511
S-X-L_1	0.0544	0.0402	0.0620	0.0456	0.0586	0.0445	0.0718	0.0573
D-X-L_1	0.0492	0.0365	0.0566	**0.0410**	0.0513	0.0394	0.0556	0.0444
S-X-L_2	0.0545	0.0403	0.0631	0.0480	0.0586	0.0445	0.0718	0.0573
D-X-L_2	**0.0453**	**0.0335**	**0.0550**	0.0417	**0.0449**	**0.0347**	**0.0519**	**0.0413**
Observ.	0.2371	0.1482	0.2368	0.1463	0.2383	0.1482	0.2787	0.0341

We compare several combinations of vertex- and temporal-domain regularizations. The methods are labeled based on 1. whether it leverages a static or a dynamic graph (S/D-), 2. the content of $\mathrm{P}(\mathbf{X})$ used for the graph-Laplacian-based regularization (-X/DX-), and 3. whether $\|\cdot\|_1$ or $\|\cdot\|_F^2$ is used for the temporal-difference-based regularization (-L_1/L_2). (refer to the method column of Table 3.3). Note that the Methods D-DX-L_1, D-DX-L_2, D-X-L_1, and D-X-L_2 are the different implementations of the proposed formulation (3.2) where S-DX-L_1, S-DX-L_2, S-X-L_1, and S-X-L_2 are their static counterparts, respectively.

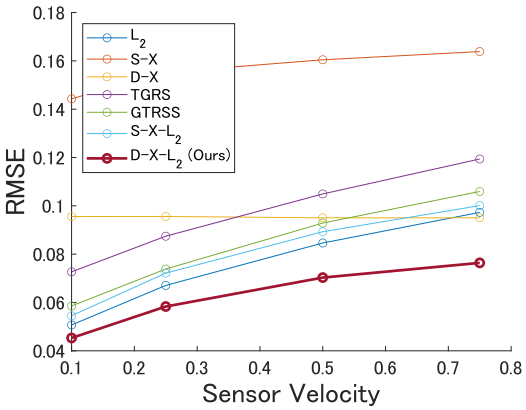


(a) Noise level: σ

(b) Noise level: P_s



(c) Noise level: P_p



(d) Sensor velocity

Figure 3.2: The graphs of RMSE vs various noise levels and sensor velocity on the 128×100 synthetic (smooth) dataset. The fixed noise levels are set as $\sigma = 0.1, P_s = 0.1, P_p = 0.1$ and velocity $v = 0.25$.

Five trials are conducted for every setting and the averaged results are presented. For all the methods that use the regularization parameter λ, we conduct a linear search for every trial. Then we determine a value that performs well for all trials on average. The results of the trials that used the determined λ are used to give the final averaged results. For static methods that use only one graph Laplacian, we input $\mathbf{L}_{p/2}$, the graph constructed at the middle of the time-slots ($t = p/2$).

We compare our methods with TGSR [85] and GTRSS [38], which are state-of-the-art time-varying graph signal recovery methods that leverage signal smoothness in the vertex and temporal domains. For a fair comparison, all methods, including the comparison methods, have been modified to support the same ℓ_1-norm ball constraint as the proposed formulation.

All of the methods share the same constraints $\|\mathbf{Y} - \Phi(\mathbf{X} + \mathbf{S})\|_F \leq \varepsilon, \|\mathbf{S}\|_1 \leq \eta$. This is to avoid parameter tuning in cases where the noise level is known a priori or can be estimated, and in our experiments, ε and η are set to:

$$\varepsilon = 0.9\sigma\sqrt{np(1 - P_s)(1 - P_p)}, \ \eta = \frac{P_s}{2}np. \tag{3.20}$$

All of the methods are implemented by PDS. The stepsizes are set as follows in correspondence to the condition mentioned in Section 2.2.2: $\gamma_1 = \frac{1}{\beta}$, $\gamma_2 = \frac{0.49}{\gamma_1 \lambda_1(\mathbf{A}^\top \mathbf{A})}$.

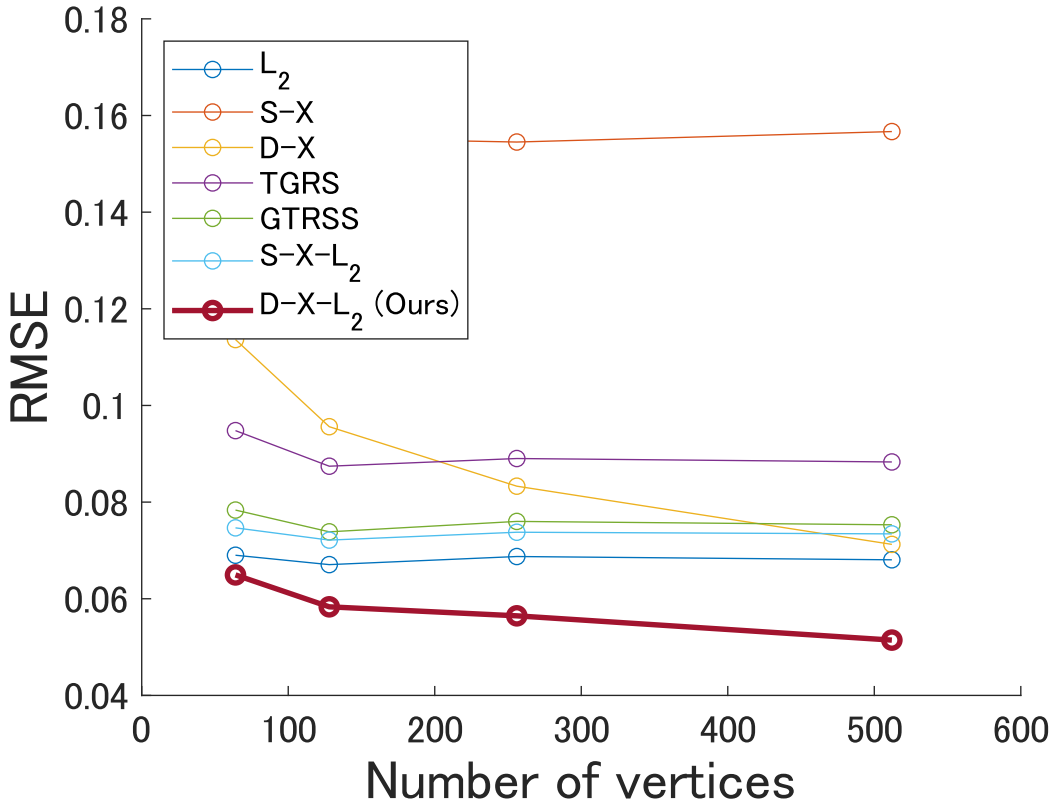


Figure 3.3: The graph of RMSE vs various network sizes on the synthetic (smooth) dataset. The noise levels are set as $\sigma = 0.1, P_s = 0.1, P_p = 0.1$ and the sensor velocity is $v = 0.25$.

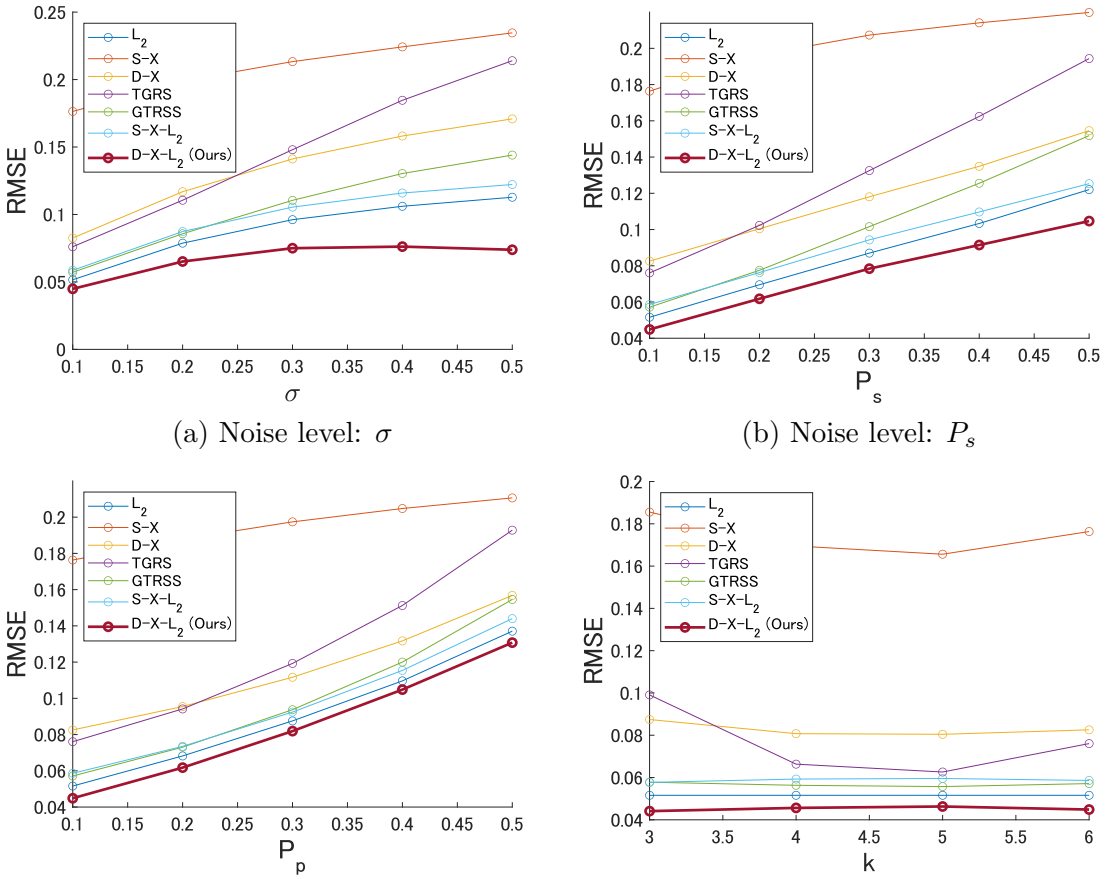
3.3.3 Results and Discussion

Dynamic vs Static Graph Laplacian

Looking at the results in Table 3.3 and Figs. 3.2 and 3.4, the methods that integrate time-varying graph-Laplacian-based regularization (the methods in red, namely, Methods D-X, D-DX-L_1, D-DX-L_2, D-X-L_1, D-X-L_2) generally outperform their static counterparts, which only consider a static graph (methods in blue, namely, Methods S-X, S-DX-L_1, S-DX-L_2, S-X-L_1, S-X-L_2). In particular, when comparing Methods S-X and D-X, which represent a direct comparison between static and dynamic graph-based regularization, the dynamic regularization consistently shows better performance.

Moreover, comparing Methods D-X-L_2 and S-X-L_2 to Methods L_2 (15th and 10th row to the 2nd row in Table 3.3), the static graph-Laplacian-based regularization even seems to be hindering the overall recovery performance. This finding highlights that using a constant $\mathbf{L}$ across all time slots should be avoided when the graph is inherently dynamic.

Regarding the methods that employed the regularization $R_v(\mathrm{D}(\mathbf{X}))$ (Methods D-DX-L_1/L_2: 10th and 11th row in Table 3.3), the performance was not as strong as initially expected. While we hypothesized that the dynamic versions would outperform their static counterparts, the results indicate that the dynamic and static versions perform similarly in most cases. We believe this may be due to the fact that the term $\mathbf{x}_{k+1} - \mathbf{x}_k$ is computed using $\mathbf{L}_k$ in the dynamic version, while $\mathbf{x}_{k+1}$ is actually observed on $\mathbf{L}_{k+1}$. This mismatch could result in suboptimal performance. However, we suspect that this effect is dataset-dependent.



(a) Noise level: σ

(b) Noise level: P_s

(c) Noise level: P_p

(d) Num. of edges: k

Figure 3.4: The graphs of RMSE vs various noise levels and k for graph construction on the sea surface temperature (Pacific) dataset. The fixed noise levels are set as $\sigma = 0.1, P_s = 0.1, P_p = 0.1$, $k = 4$.

By inspecting the results closely, we observed that the dynamic version does indeed outperform the static version in the sea surface temperature dataset. The difference in performance between the synthetic and sea surface temperature datasets seems to stem from the level of smoothness in the vertex domain. The sea surface temperature dataset exhibits significantly higher smoothness in the vertex domain compared to the synthetic dataset, meaning the difference between $\mathbf{L}_k$ and $\mathbf{L}_{k+1}$ is smaller. When the difference between $\mathbf{L}_k$ and $\mathbf{L}_{k+1}$ is smaller, the term $\mathbf{x}_{k+1} - \mathbf{x}_k$ is more likely to exhibit smoothness on $\mathbf{L}_k$, thereby improving the performance of the dynamic version.

Additionally, $R_v(\mathrm{D}(\mathbf{X}))$ $(=(\mathbf{x}_{k+1} - \mathbf{x}_k)^\top \mathbf{L}_k (\mathbf{x}_{k+1} - \mathbf{x}_k))$ was originally designed to address cases where the vertex domain is less smooth than typically assumed, while also capturing temporal characteristics. In our experiments (both on synthetic and sea surface temperature datasets), the vertex domain was found to be very smooth, and we explicitly added regularization terms to handle the temporal domain. We expect that the performance of $R_v(\mathrm{D}(\mathbf{X}))$ would improve in datasets where the vertex domain is less smooth.

Finally, our method achieved superior results compared to the state-of-the-art static graph-based methods TGSR and GTRSS (6th and 7th row in Table 3.3), which further supports the advantage of incorporating dynamic graphs.

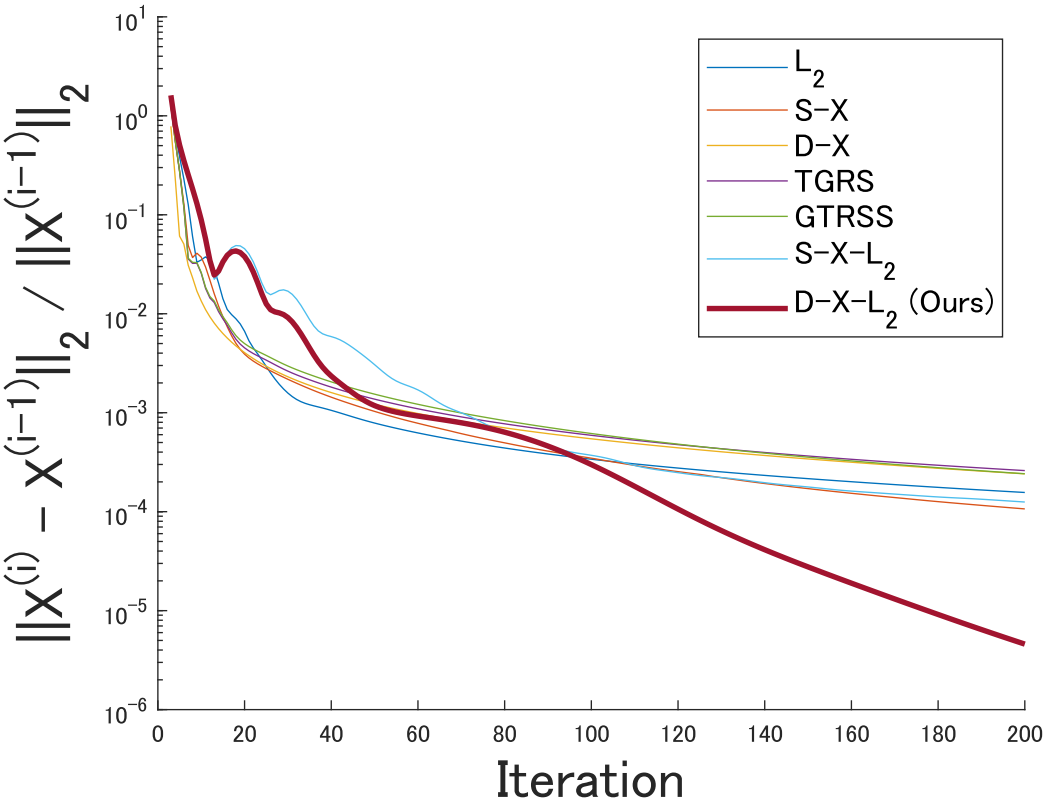


Figure 3.5: The graph of convergence rate on the sea surface temperature dataset (Pacific). The fixed noise levels are set as $\sigma = 0.1, P_s = 0.1, P_p = 0.1$.

Vertex vs Temporal Domain Priors

In this section, we discuss the impact of vertex and temporal-domain priors on recovery performance. Comparing Methods L_1 and L_2 (which leverage only temporal-domain priors, 1st and 2nd row in Table 3.3) to Methods S-X and D-X (which leverage only vertex-domain priors, 4th and 5th row in Table 3.3), we observe that leveraging temporal-domain priors is much more effective than leveraging vertex-domain priors for the synthetic dataset. This is further supported by the relatively small performance gains between Methods L_1, L_2, and Methods D-X-L_1, D-X-L_2 (which leverage priors from both domains). We speculate that this is because the low velocity and noise levels make the graph signal extremely smooth in the temporal domain, whereas the vertex domain is relatively less smooth even in the noiseless case (Fig. 3.1 (a)). Therefore, the temporal-domain prior is sufficient for signal recovery, and the additional advantage of using the vertex-domain prior is relatively small.

However, the benefits of leveraging the vertex-domain prior become more apparent in the real-world dataset (4th and 5th row in Table 3.3). This is because the sea surface temperature dataset is much smoother in the vertex domain compared to the synthetic dataset. Although the difference in performance between the two regularizations (1st, 2nd row and 4th, 5th row in Table 3.3) depends on the nature of the dataset (specifically, the smoothness of the signals in their respective domains), the proposed methods can balance the effects of the two regularizations through the parameter λ. Considering that our dynamic methods generally performed better, especially in high-noise settings (Figs. 3.2 (a) and 3.4 (a)), our methods demonstrate robustness to the varying nature of datasets.

Supporting this, our method D-X-L_2 (15th row in Table 3.3) performed the best in most settings.

Sensor Velocity

From Fig. 3.2 (d), we can observe that all methods leveraging the temporal domain show a decline in performance as the velocity v of the sensors increases. This decline occurs because, as the velocity increases, the signals become less smooth in the temporal domain. Consequently, the performance of Methods L_1 and L_2 deteriorates, while Method D-X remains largely unaffected. Similarly, the performance gap between Methods D-X-L_2 and L_2 (Fig. 3.2 (d)) widens as the velocity increases, further demonstrating the advantage of incorporating dynamic graph-based regularization.

Methods that rely on static graphs also experience a decrease in performance as the sensor velocity increases. This is because the higher the sensor velocity, the greater the mismatch between the single static graph these methods use and the actual, time-varying graphs on which the signals are observed. Thus, static methods struggle to capture the true dynamics of the signals in high-velocity scenarios.

In practical settings, differences in sensor velocity are often equivalent to variations in sampling frequency. A sensor sampling at a constant frequency while moving at different velocities is essentially equivalent to a sensor moving at a constant velocity but sampling at different frequencies, assuming the underlying signal distribution remains time-invariant. Our method demonstrates the ability to tolerate higher velocities compared to the temporal-domain-only methods L_1 and L_2, meaning it can also tolerate lower sampling frequencies. This indicates that leveraging both vertex and temporal domains can lead to a more cost-efficient recovery method by allowing for lower sampling frequencies without a significant drop in performance.

Noise Levels

We conducted extensive experiments across various noise levels, including σ, P_s, and P_p (see Fig. 3.2 and Fig. 3.4). As expected, the recovery performance degrades as the noise level increases. However, the proposed method consistently outperforms the state-of-the-art time-varying graph signal recovery methods across all noise settings. It is important to note that the comparison methods were modified to support recovery from sparse outliers to ensure a fair comparison. We anticipate an even larger performance gap when compared to the original, unmodified formulations of these methods.

Vertex and Edge Density

As shown in Fig. 3.3, methods using graph-Laplacian-based regularizations generally perform better as the number of vertices (sensors) increases. This is because, in our synthetic setting, a larger number of vertices leads to a spatially denser sensor network. We generate the graph Laplacians using the k-nearest neighbors algorithm ($k = 4$) and Gaussian kernels based on spatial coordinates, under the assumption that vertices in close spatial proximity have similar signal values. While graph Laplacians generated in this way are known to provide smooth and accurate representations of graphs in many situations, the quality of the representation can vary depending on the data. In our case, a denser graph implies that the four vertices each vertex is connected to are spatially closer and, therefore, more likely to have similar signal values. As a result, the setting with more vertices aligns more closely with the assumptions used to generate the data, improving performance.

We also investigated the impact of edge density on the sea surface temperature dataset (Fig. 3.4 (d)). Although performance varies slightly depending on the value of k used

Table 3.4: The Average Running Time for Method D-X Implemented With ADMM And PDS

Dataset size	Time (sec.) / per iteration ($\times 10^{-4}$ sec.)	
	ADMM	PDS
64×100	0.3137 / 4.140	0.1118 / 4.395
64×200	0.6776 / 8.846	0.2688 / 8.889
128×100	1.0780 / 18.40	0.6297 / 18.21
128×200	2.4727 / 42.38	1.4246 / 41.47
256×100	4.3046 / 93.89	3.3033 / 81.89
256×200	8.6370 / 181.1	6.2086 / 158.5
512×100	45.194 / 604.2	30.144 / 584.1
1024×100	147.12 / 2057	127.31 / 2024

in the k-nearest neighbor algorithm to generate the graph Laplacians, the overall trend and performance of the proposed method remain consistent. Thus, we can confirm that the proposed method is robust to the choice of k, and the standard coordinate-to-graph generation scheme is sufficient to support graph-based signal recovery.

Smooth and Piece-wise Flat Signals

For methods that leverage the temporal prior, we initially expected those using Frobenius-norm regularization to perform better on smooth datasets, and methods using the $\|\cdot\|_1$ regularization term to excel on piece-wise flat datasets, given that $\|\cdot\|_1$ regularization typically performs better on data with staircase-like structures. However, contrary to our expectations, methods using $\|\cdot\|_1$ and $\|\cdot\|_F^2$ regularizations performed comparably on the piece-wise flat dataset (1st and 2nd row in Table 3.3).

We speculate that this outcome is due to high noise levels and sensor velocity, which likely disrupted the sparsity of the temporal difference signal. As a result, the smoothing ability of Frobenius-norm regularization may have outweighed the advantages of $\|\cdot\|_1$ regularization in this scenario.

Visual Analysis

Referring to the visual results of the experiments (Figs. 3.6, 3.7, 3.8 and 3.9), we can observe that our methods are able to recover the graph signals fairly well. In these figures, the area of each node is proportional to the magnitude of the error, with larger areas indicating larger errors. It is important to note that, while the color range for the observations is clipped to [0,1] to standardize the color bars across all figures, the magnitude of the error represented by the node sizes is not clipped. As a result, some nodes may have the same color but different sizes, indicating different error magnitudes.

In Fig. 3.6, we see that a small region in the top middle of the graph, where signal values are high, is being overly smoothed. Beyond the challenge posed by the high noise level, the underlying distribution characteristics (Fig. 3.1) also contribute to the difficulty of the problem. The part of the distribution with high values manifests as spike signals in both the vertex and temporal domains, making accurate recovery harder.

For the same reason, lower sensor velocities help preserve signal smoothness in the temporal domain, thereby improving the performance of methods that leverage temporal-domain priors.

Algorithm Efficency

We present the computational time of a PDS-based algorithm in comparison to a commonly used Alternating Direction Method of Multipliers (ADMM)-based algorithm [14]. Method D-X was reimplemented using ADMM, and the running times are compared in Table 3.4 (dataset: synthetic, σ, P_s, $P_p = 0.5$). Unlike ADMM, the PDS algorithm avoids matrix inversions, leading to faster computation times. This difference becomes more pronounced in the time per iteration as the data dimension increases, since matrix inversions for larger data sizes can be computationally complex.

Additionally, we provide a figure showing the convergence of the PDS-based algorithm in Fig. 3.5. It is important to note that the comparison methods were reformulated and reimplemented to handle the same constraints as our method, ensuring a fair comparison.

3.4 Concluding Remarks

In this chapter, we proposed a novel time-varying graph signal recovery problem based on an explicitly dynamic graph model, where the underlying graph is assumed to vary over time, similar to the signals. Our formulation incorporates time-varying graph-Laplacian-based regularization and temporal-difference-based regularization, effectively leveraging priors from both the vertex and temporal domains, while also accounting for the dynamic nature of the graph. By jointly estimating sparsely modeled outliers with the graph signals, our method achieves robustness against various types of noise.

Through experiments on both synthetic and real-world data, we compared our method with existing approaches that apply regularization in the vertex and temporal domains. We discussed how the priors from both domains contribute to recovery performance under different conditions. The proposed problem setting is practical, especially in applications such as remote sensing using mobile sensors, where dynamic graph Laplacians can be easily generated to represent spatial correlations between sensors. Dynamic sensor networks offer a practical and feasible approach to sensing, and we view our research as a foundational step toward addressing this emerging problem.

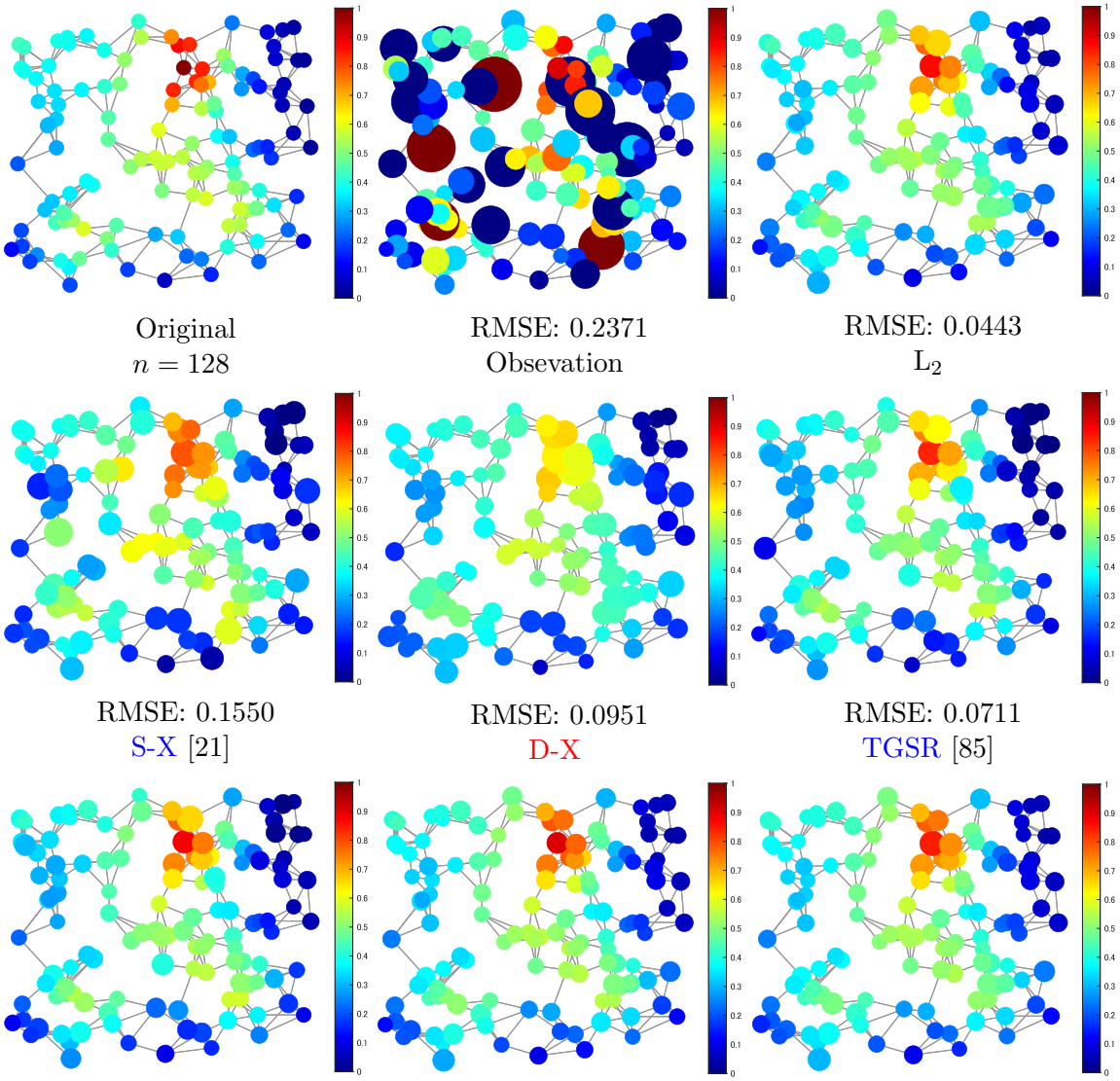


Figure 3.6: Graph signal recovery results of the synthetic (smooth) 128×100 dataset ($\sigma, P_s, P_p = 0.1, v = 0.25$). The area of the nodes is proportional to the magnitude of the error, where larger areas indicate larger errors. Note that although the color range for the observations is clipped to [0,1] to unify the colorbars of all the figures, the magnitude of error represented by the size of the nodes is not clipped, therefore, some nodes may be colored the same but differently sized.

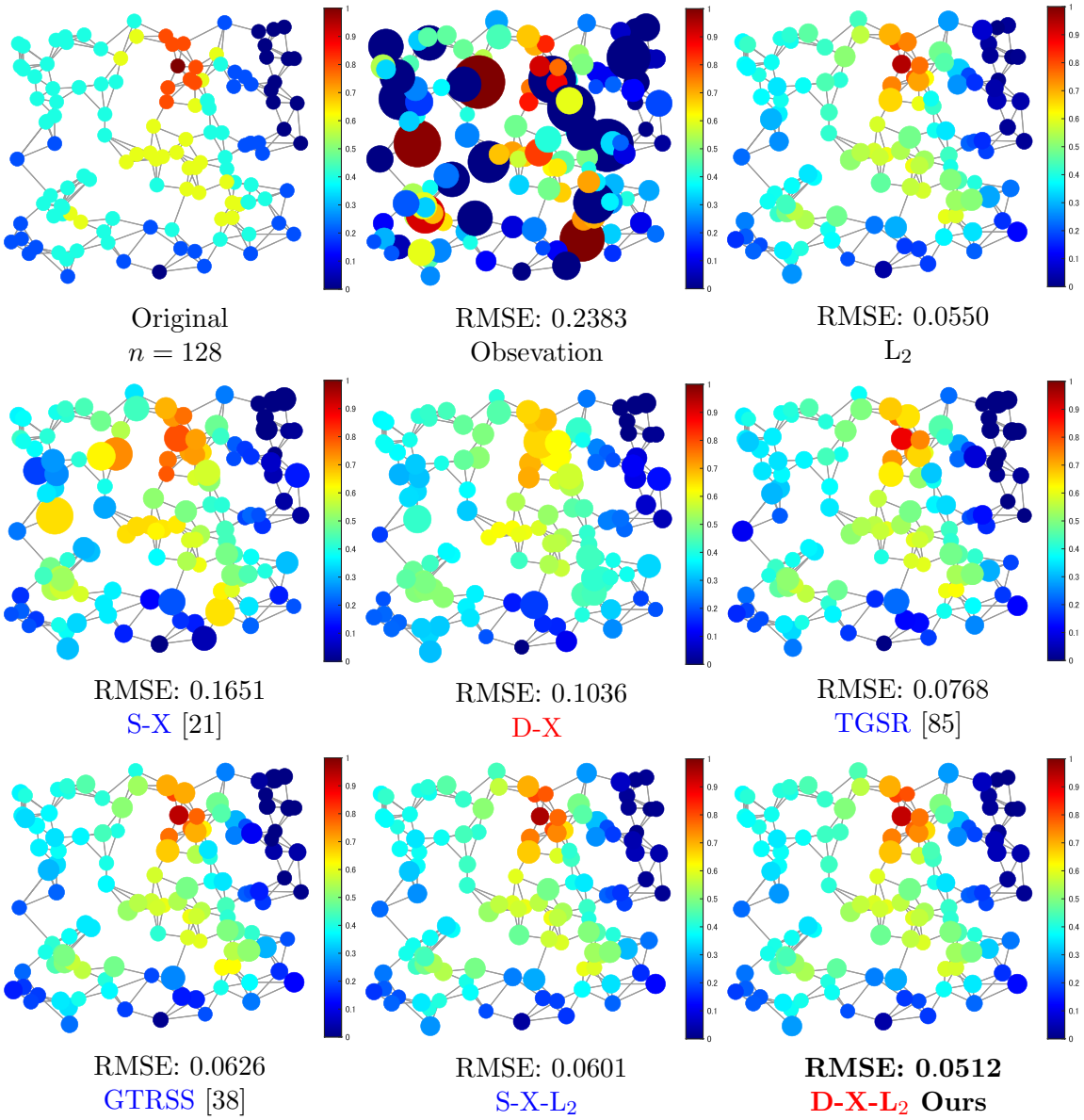


Figure 3.7: Graph signal recovery results of the synthetic (piece-wise smooth) 128×100 dataset ($\sigma, P_s, P_p = 0.1, v = 0.25$). The area of the nodes is proportional to the magnitude of the error, where larger areas indicate larger errors. Note that although the color range for the observations is clipped to [0,1] to unify the colorbars of all the figures, the magnitude of error represented by the size of the nodes is not clipped, therefore, some nodes may be colored the same but differently sized.

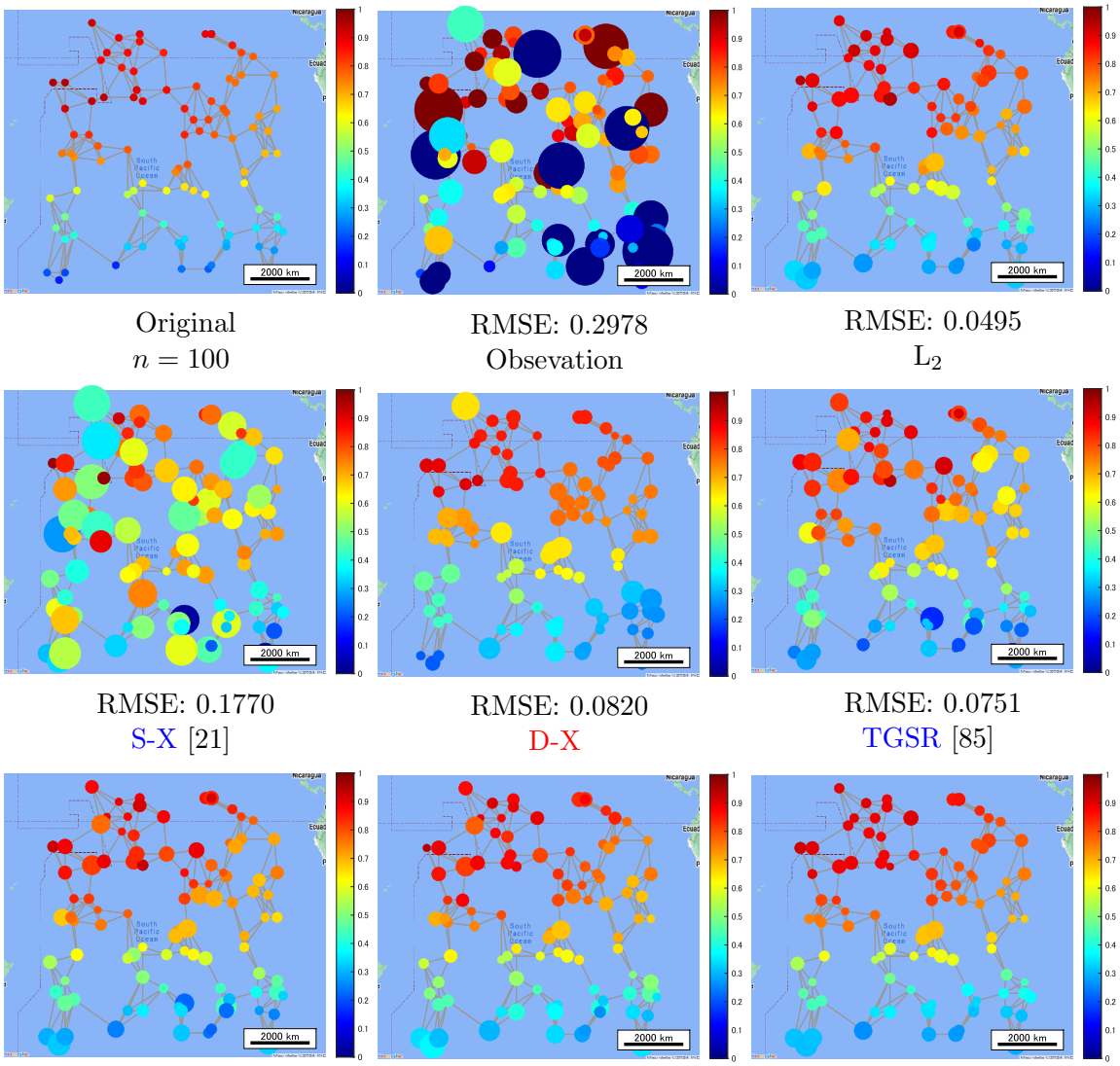


Figure 3.8: Graph signal recovery results of the sea surface temperature data (Pacific Ocean) $(\sigma, P_s, P_p = 0.1)$. The area of the nodes is proportional to the magnitude of the error, where larger areas indicate larger errors. Note that although the color range for the observations is clipped to [0,1] to unify the colorbars of all the figures, the magnitude of error represented by the size of the nodes is not clipped, therefore, some nodes may be colored the same but differently sized.

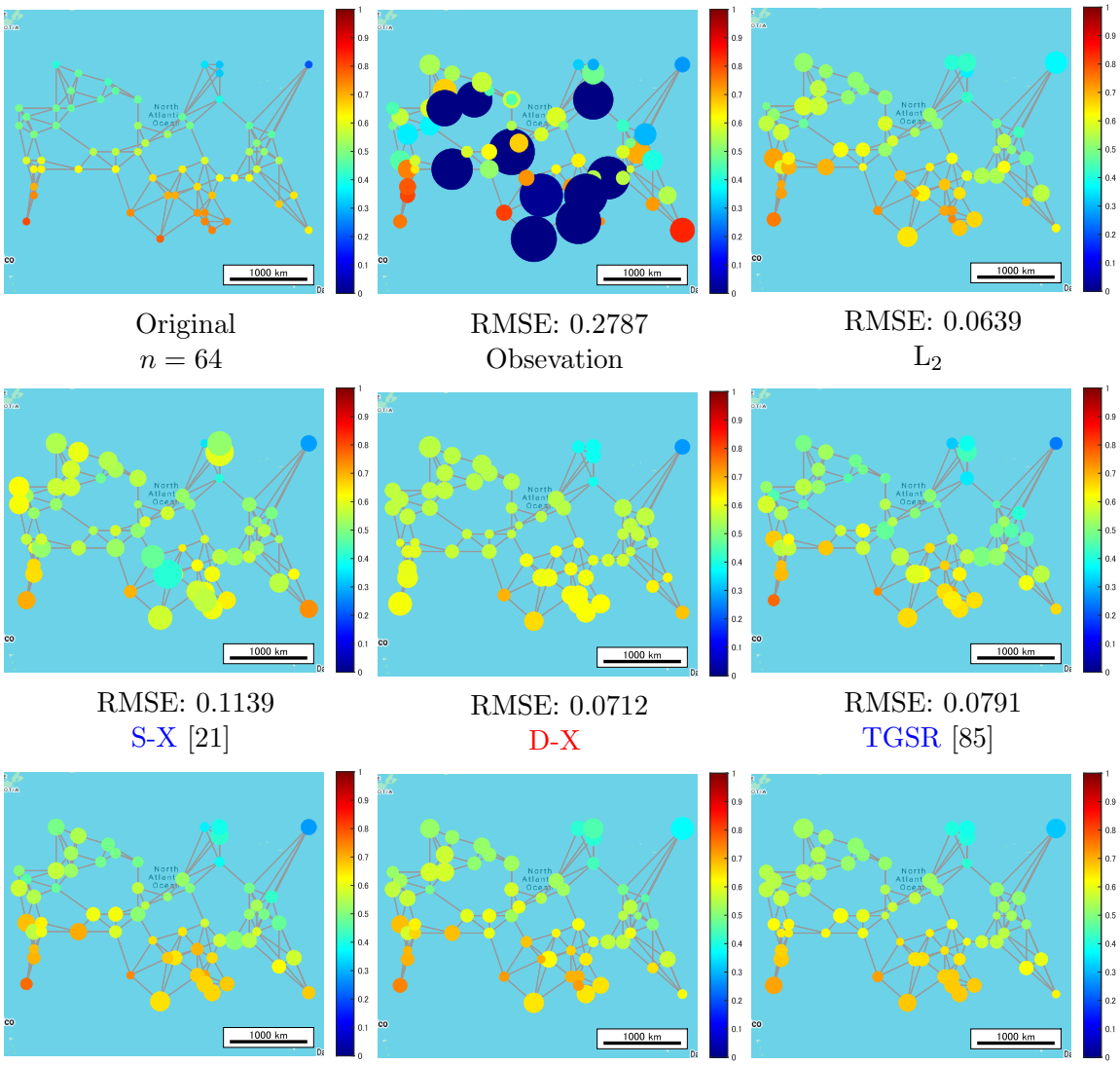


Figure 3.9: Graph signal recovery results of the sea surface temperature data (Atlantic Ocean) ($\sigma, P_s, P_p = 0.1$). The area of the nodes is proportional to the magnitude of the error, where larger areas indicate larger errors. Note that although the color range for the observations is clipped to [0,1] to unify the colorbars of all the figures, the magnitude of error represented by the size of the nodes is not clipped, therefore, some nodes may be colored the same but differently sized.

Chapter 4

Market Graph Integrated Sparse Index Tracking

4.1 Introduction

Sparse index tracking [8,9,37,53,109] is a prominent passive investment strategy [61] that aims to replicate the performance of a financial index. Since markets are empirically known to experience long-term growth [5], reasonable returns can be expected by tracking an index representative of the market. In order to track and invest in an index, an investor must distribute their capital across the numerous assets that constitute the index (e.g., 500 assets in the case of the S&P500). To accomplish this, the investor must construct a portfolio, a vector of weights that indicates how the total capital is allocated.

Sparse index tracking is reduced to a regression problem where the goal is to construct a portfolio based on sparsity and other criteria. One such criterion is sector neutrality [8,18], proposed to reduce the risk of biased portfolios that heavily invest in specific industries. Assets typically belong to one of the sectors defined by the Global Industry Classification Standard (GICS). For instance, the S&P500 comprises 11 sectors: consumer discretionary, consumer staples, energy, financials, health care, industrials, information technology, materials, telecommunications, utilities, and real estate. A portfolio constructed without consideration of the sector information can be biased towards specific industries. This bias could pose a significant risk, as an industry-wide event can impact all assets within the corresponding sector, potentially causing catastrophic damage to the capital.

Although sector neutrality does succeed in reducing the investment risk, it has some drawbacks, namely, the potential bias in the number of assets included in the sectors. In the case of the S&P500 index, there is a more than eight times difference between the sectors with the most and least number of assets (this difference varies on the time period). This raises the question: *Could we construct a new sparse index tracking framework that can diversify our investments more effectively based on entirely new, less biased asset groups?*

4.1.1 Contributions and Chapter Organization

In this study, we propose a risk-managed sparse index tracking framework that can strictly diversify investments based on asset groups constructed from actual data. To this end, we first introduce a new sparse index tracking formulation that enhances the usablity of the conventioal tracking algorithms. Conventional tracking methods involved using either an ℓ_p-norm ($0 < p < 1$) regularization or an ℓ_0-norm constraint based on an algorithm that separates asset selection and capital allocation into two steps. We propose a new ℓ_0-norm-constraint-based formulation and an algorithm that can simultaneously handle

asset selection and capital allocation. This allows us to directly control the number of assets in the portfolio, which is a significant advantage in actual asset management.

Building upon this formulaiton, we then introduce a new risk-managed sparse index tracking method that leverages market-graph neutrality. First, we perform graph clustering [4, 30] on a market graph [46, 94, 95] to divide the assets into groups that we can use instead of sectors, which we refer to as market-graph clusters.

Then, the clusters are used to impose market-graph neutrality on the sparse index tracking problem. We hypothesize that clusters that are less biased and derived from actual data can better reflect the connections between assets and better reduce portfolio risks compared to using heuristically pre-defined sectors. Furthermore, in the context of graph representation, the conventional sectors are seldomly updated, therefore can be interpreted as a static graph representation of the market. In contrast, the market graph is updated at every rebalancing to reflecting the latest correlations between the assets, which is based on the idea of a dynamic graph representation. However, using clusters that may change at every rebalancing could potentially increase the number of assets traded, increasing the transaction costs. To solve this, we also impose turnover sparsity to our market-graph neutral sparse index tracking problem to minimize the transaction costs. We formulate the tracking problem so that the diversity of the portfolio is ensured by hard constraints. A primal-dual splitting (PDS) [24, 100] based algorithm is developed to handle the formulated problem. The main contributions of this study are as follows.

- We develop a PDS-based algorithm that can directly handle an ℓ_0-norm constraint while performing asset selection and capital allocation simultaneously.
- We propose a risk-managed sparse index tracking method, which constructs portfolios based on market-graph neutrality and turnover sparsity.

The chapter is organized as follows. In Section 4.2, we introduce some basic concepts related to sparse index tracking and graph clustering. In Section 4.3, we present the proposed method, along with its benefits and optimization algorithm. Finally, in Section 4.4, we evaluate our method on real-world datasets.

4.2 Related Work

4.2.1 Sparse Index Tracking

The objective of sparse index tracking [8, 9, 37, 53, 109] is to construct a sparse portfolio $\mathbf{w} \in \mathbb{R}^N$, matching the performance of the benchmark market index, with $K(<< N)$ nonzero weights out of N total assets.

The conventional approach to this was to divide the problem into two phases, namely, asset selection and capital allocation. Numerous methods for asset selection have been proposed. A typical approach was to select K largest assets in terms of market capitalization [71]. Other methods include selecting assets that exhibit similar performance to the target index [11, 33] or selection based on the cointegration between log-prices of the K assets and the index value [1]. However, the impact of the two-step approach on tracking performance remained uncertain, prompting the development of one-step methods [8,9] as alternatives. The LAIT (Linear Approximation for Index Tracking) algorithm proposed in [9] confirmed the positive effects of simultaneous asset selection and capital allocation by solving the following regression problem:

$$\min_{\mathbf{w}} \frac{1}{T}\|\mathbf{r^b} - \mathbf{X}\mathbf{w}\|_2^2 + \lambda \mathrm{R}(\mathbf{w}) \text{ s.t. } \begin{cases} \mathbf{w} \geq \mathbf{0}, \\ \mathbf{1}^\top \mathbf{w} = 1. \end{cases} \tag{4.1}$$

where $\mathrm{R}(\mathbf{w})$ is set to an ℓ_p-norm regularization $(0 < p < 1)$[1] to enforce portfolio sparsity[2]. The benchmark index returns across T days are denoted by $\mathbf{r^b} = [r_1^{\mathbf{b}}, ..., r_T^{\mathbf{b}}]^\top \in \mathbb{R}^T$ and the asset-wise returns across T days are denoted by $\mathbf{X} = [\mathbf{r}_1, ..., \mathbf{r}_T]^\top \in \mathbb{R}^{T\times N}$. The nonnegative constraint prohibits short-selling (negative weights), and the sum-to-one constraint is enforced to allocate weights based on a constant capital.

However, in the formulation of (4.1), there is no explicit relationship between the hyperparameter λ and the number of assets in the portfolio, making it difficult to search for a λ that would maintain a certain number of assets. This means that the investor cannot directly set the desired number of assets, which hinders actual asset management.

To solve this, an algorithm named NNOMP-PGD [53] that combines nonnegative orthogonal matching pursuit [102] and projected gradient descent [70] has been proposed. The method separates the sparse index tracking problem into two stages:

$$\min_{\mathbf{w}} \frac{1}{T}\|\mathbf{r^b} - \mathbf{X}\mathbf{w}\|_2^2 \text{ s.t. } \begin{cases} \|\mathbf{w}\|_0 \leq K_1, \\ \mathbf{w} \geq \mathbf{0}, \end{cases} \tag{4.2}$$

and

$$\min_{\mathbf{w}} \frac{1}{T}\|\mathbf{r^b} - \mathbf{X}\mathbf{w}\|_2^2 \text{ s.t. } \begin{cases} \mathbf{w} \geq \mathbf{0}, \\ \mathbf{1}^\top\mathbf{w} = 1. \end{cases} \tag{4.3}$$

The first stage solves the asset selection problem with an ℓ_0-norm [41, 52, 54, 55, 58, 73] constraint (i.e., the number of nonzero entries in $\mathbf{w}$ is less than or equal to a user-set parameter) by NNOMP, and the second stage solves the capital allocation problem by PGD. By solving a formulation enforced with an ℓ_0-norm constraint, investors can easily control the number of assets that compose the portfolio by the parameter K_1.

Although NNOMP-PGD does succeed in directly controlling the sparsity of the portfolio through the ℓ_0-norm constraint, we believe that the tracking performance can be improved by conducting the asset selection and capital allocation simultaneously. This is due to the nonnegative orthogonal matching pursuit algorithm, wherein the value of the nonzero element added to $\mathbf{w}$ per iteration directly influences the selection of the nonzero element in $\mathbf{w}$ for the following iterations. The nonzero element values do not necessarily satisfy the sum-to-one constraint or even the upper limit.

Sector Neutrality

Let $\mathbf{M} \in \mathbb{R}^{S\times N}$ (S is the number of sectors) be a binary mask, where $m_{ij} = 1$ if the j-th asset belongs to the i-th sector, and $m_{ij} = 0$ otherwise (a normalized version $\widetilde{\mathbf{M}} = \mathrm{Diag}(\mathbf{M1})^{-1}\mathbf{M}$, is used in the actual formulation). A sector neutral sparse index tracking problem can be formulated as follows [8]:

$$\min_{\mathbf{w}} \frac{1}{T}\|\mathbf{r^b} - \mathbf{X}\mathbf{w}\|_2^2 + \theta\|\widetilde{\mathbf{M}}\mathcal{I}_{\mathbf{w}}\|_2^2 \text{ s.t. } \begin{cases} \mathbf{w} \geq \mathbf{0}, \\ \mathbf{1}^\top\mathbf{w} = 1, \end{cases} \tag{4.4}$$

where $\mathcal{I}_{\mathbf{w}} \in \mathbb{R}^N$ indicates the nonzeros elements of $\mathbf{w}$, i.e. $\mathcal{I}_{\mathbf{w}j} = 1$ if $\mathbf{w}_j > 0$, 0 otherwise. Matrix $\mathrm{Diag}(\cdot)$ denotes a diagonal matrix with $\cdot$ as its principal diagonal.

By leveraging sector neutrality, we can avoid biased portfolios, where the weights are only distributed to some of the sectors. It should be noted, however, that since sector neutrality is imposed as a regularization term in 4.4, it only promotes sector neutrality and does not guarantee strict diversity across all sectors.

[1] Although $\|\cdot\|_p$ is strictly speaking not a norm when $0 \leq p < 1$, it is conventionally referred to as the ℓ_p-norm.

[2] To avoid nonconvex optimization, the ℓ_1-norm is commonly used as a surrogate function of the ℓ_0-norm. However, due to the short-selling and sum-to-one constraints, the ℓ_1-norm of the portfolio is always a constant ($\|\mathbf{w}\|_1 = 1$). Therefore, the ℓ_1-norm regularization cannot be used in this formulation.

4.2.2 Tracking Error Measure

Several measures of tracking error (TE) have been proposed, the most common measure being the empirical tracking error (ETE) [7, 9, 43, 62]:

$$\mathrm{ETE}(\mathbf{w}) := \frac{1}{T}\|\mathbf{r^b} - \mathbf{X}\mathbf{w}\|_2^2. \tag{4.5}$$

The empirical tracking error is convex and differentiable as

$$\nabla\mathrm{ETE}(\mathbf{w}) = -\frac{2}{T}\mathbf{X}^\top(\mathbf{r^b} - \mathbf{X}\mathbf{w}). \tag{4.6}$$

The gradient $\nabla\mathrm{ETE}(\mathbf{w})$ is β-Lipschitz continuous, where $\beta = 2/T\lambda_1(\mathbf{X}^\top\mathbf{X})$ ($\lambda_1(\cdot)$ denotes the maximum eigenvalue of $\cdot$).

Considering the original purpose of index tracking, which is to make a profit by investing, ETE that penalizes even when the portfolio beats the benchmark index is not desirable. Therefore, the measure downside risk (DR) [9, 37] has been proposed to penalize only when the returns are behind that of the benchmark index:

$$\mathrm{DR}(\mathbf{w}) := \frac{1}{T}\|(\mathbf{r^b} - \mathbf{X}\mathbf{w})^+\|_2^2, \tag{4.7}$$

where $[(\mathbf{r^b} - \mathbf{X}\mathbf{w})^+]_i = \max\{[\mathbf{r^b} - \mathbf{X}\mathbf{w}]_i, 0\}$. The downside risk is also convex and differentiable. Consider $g_1(\mathbf{w}') = \|(\mathbf{w}')^+\|_2^2/T$ and $g_2(\mathbf{w}) = \mathbf{r^b} - \mathbf{X}\mathbf{w}$ so that $\mathrm{DR} = g_1(g_2(\mathbf{w}))$. The derivatives are

$$\begin{aligned}[\nabla g_1(\mathbf{w}')]_i &= \begin{cases} 2\mathbf{w}'_i/T, & \text{if } \mathbf{w}'_i \geq 0, \\ 0, & \text{otherwise,} \end{cases} \\ \therefore \nabla g_1(\mathbf{w}') &= 2(\mathbf{w}')^+/T,\end{aligned} \tag{4.8}$$

and

$$\nabla g_2(\mathbf{w}) = -\mathbf{X}^\top, \tag{4.9}$$

respectively. Therefore:

$$\begin{aligned}\nabla\,\mathrm{DR}(\mathbf{w}) &= \nabla(g_1 \circ g_2)(\mathbf{w}) \\ &= \nabla g_2(\mathbf{w})\nabla g_1(g_2(\mathbf{w})) \\ &= -\frac{2}{T}\mathbf{X}^\top(\mathbf{r}^b - \mathbf{X}\mathbf{w})^+.\end{aligned} \tag{4.10}$$

Furthermore, when comparing ETE and DR, the gradient of DR clearly does not exceed that of ETE. Therefore, DR can be considered β-Lipschitz continuous for at least the same value of β as ETE.

4.2.3 Transaction Costs

The typical transaction costs model applied in the U.S. markets is $\$0.005 \times v$ with a minimum cost of \$1, where v is the trading volume. When the total capital is sufficiently large (making the volume-wise cost predominant), the total transaction cost is not influenced by the number of assets traded. Conversely, with smaller capital amounts, where the minimum cost prevails, the number of traded assets directly impacts transaction costs. Thus, a sparse portfolio is desired.

In the same context, a turnover constraint [8,9] has been introduced to cap the number of assets traded during each rebalancing period, defined as $\|\mathbf{w}_0 - \mathbf{w}\|_0 \leq K_2$, with $\mathbf{w}_0$ representing the portfolio prior to rebalancing. By imposing this constraint, only a limited number of assets are updated, limiting the number of assets traded, therefore reducing transaction costs.

4.2.4 Portfolio Cut Framework

The portfolio cut framework [4, 30] performs portfolio cut (a graph clustering method) on a market graph.

Market Graph

We can use the N assets that compose an index to construct a market graph [46, 94, 95], where the weight adjacency matrix [69, 87, 92] is defined based on absolute correlation coefficients:

$$\mathbf{A} = \begin{bmatrix} 0 & \frac{|\sigma_{12}|}{\sqrt{\sigma_{11}\sigma_{22}}} & \cdots & \frac{|\sigma_{1N}|}{\sqrt{\sigma_{11}\sigma_{NN}}} \\ \frac{|\sigma_{21}|}{\sqrt{\sigma_{11}\sigma_{22}}} & 0 & \cdots & \frac{|\sigma_{2N}|}{\sqrt{\sigma_{22}\sigma_{NN}}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{|\sigma_{N1}|}{\sqrt{\sigma_{11}\sigma_{22}}} & \frac{|\sigma_{N2}|}{\sqrt{\sigma_{NN}\sigma_{22}}} & \cdots & 0 \end{bmatrix}, \tag{4.11}$$

so that the weight of the connection between assets n and m is a_{nm}. The covariance between the returns of asset n and asset m is denoted by σ_{nm}. Therefore, statistically dependent assets are strongly connected on the market graph, and independent assets are connected by negligible weights or not connected at all (0 weight).

Vertex Clustering

The goal of vertex clustering is to divide the vertices of a graph into an arbitrary number of clusters, where the vertices of the same group are strongly connected and weakly connected to the vertices of other clusters. This can be achieved by minimizing *RatioCut* [40, 96], which has been proposed to calculate the strength of the connections between clusters. Although minimizing RatioCut is combinatorial and NP-hard, it can be relaxed to the following problem:

$$\min_{\mathbf{c}\in\mathbb{R}^N} \mathbf{c}^\top \mathbf{L}\mathbf{c} \text{ s.t.} \begin{cases} \mathbf{c}^\top \mathbf{1} = 0, \\ \mathbf{c}^\top \mathbf{c} = 1, \end{cases} \tag{4.12}$$

where $\mathbf{L} \in \mathbb{R}^{N\times N}$ denote the graph Laplacian. The solution to the above problem is given by the *Fiedler vector*, the eigenvector corresponding to the second smallest eigenvalue of $\mathbf{L}$, and in the case where the task is to divide the graph into two clusters, the membership of a vertex to each cluster can be determined by the polarity of the Fiedler vector. This can be extended to $K > 2$ clusters, in which case the solution is given as the K eigenvectors of $\mathbf{L}$ corresponding to the smallest eigenvalues. Then, the set of eigenvectors is clustered by K-means clustering.

4.3 Proposed Method

4.3.1 Problem Formulation

An ℓ_0-norm-based sparse index tracking

We formulate a new ℓ_0-norm-based index tracking problem that allows the selection of portfolio and turnover sparsity constraints.

The formulation is as follows:

$$\min_{\mathbf{w}} \mathrm{TE}(\mathbf{w}) \text{ s.t.} \begin{cases} \mathbf{w} \in S_s, \\ \mathbf{w} \in S_{l,u}, \\ \mathbf{1}^\top \mathbf{w} = 1. \end{cases} \tag{4.13}$$

Here, $\mathbf{w}$ represents the portfolio weights, S_s embodies the sparsity constraints, and $S_{l,u}$ represents the box constraints. The tracking error, denoted as TE is either ETE or DR introduced in Section 4.2.2. The sparsity constraint is denoted by $S_s = S_0$ or $S_s = S_{w_0}$ where

$$\begin{aligned} S_0 &:= \{\mathbf{w} \in \mathbb{R}^N \mid \|\mathbf{w}\|_0 \leq K_1\} \\ S_{\mathbf{w}_0} &:= \{\mathbf{w} \in \mathbb{R}^N \mid \|\mathbf{w} - \mathbf{w}_0\|_0 \leq K_2\}. \end{aligned} \tag{4.14}$$

When $S_s = S_0$, the formulation imposes sparsity on the portfolio in order to output a portfolio of K_1 sparseness. On the other hand, when $S_s = S_{\mathbf{w}_0}$, the sparsity applies to the turnover, thus producing a portfolio that only requires K_2 trades from the previous portfolio ($\mathbf{w}_0$). Note that although S_0 is a special case of $S_{\mathbf{w}_0}$ when $\mathbf{w}_0 = \mathbf{0}$, we distinguish the two for clarity and to differentiate K_1 and K_2.

The set $S_{l,u}$ is a box constraint with a lower bound of l and an upper bound of u. The lower bound is set to $l = 0$ to prohibit short selling, and an upper bound is set to avoid extreme capital allocations, which is often risky in investment.

Martket graph neutrality-imposed sparse index tracking

Based on the ℓ_0-norm-based sparse index tracking, proposed in the previous section, we introduce a risk-managed sparse index tracking method that leverages market-graph neutrality instead of the conventional sector neutrality. To replace the sector neutrality, we first perform vertex clustering on the market graph in order to divide the assets into K clusters denoted as $\mathcal{C}_1, ..., \mathcal{C}_K$. Furthermore, in the final part of the RatioCut minimization introduced in Section 4.2.4, we perform a constrained K-means clustering [15] instead of the typical K-means clustering. Constrained K-means clustering allows us to strictly constrain each cluster to contain at least N_{min} assets. We then use the clusters to construct a market-graph neutral sparse portfolio based on the following formulation:

$$\begin{aligned} &\min_{\mathbf{w}} \frac{1}{T}\|\mathbf{r}^{\mathbf{b}} - \mathbf{X}\mathbf{w}\|_2^2 + \lambda\|\mathbf{w} - \mathbf{w}_{pre}\|_1, \\ &\text{s.t.} \begin{cases} \|\mathbf{w}_{\mathcal{C}_i}\|_0 \leq k_i, & \forall i \in \{1, ..., K\}, \\ \mathbf{1}^\top \mathbf{w}_{\mathcal{C}} = b_i, & \forall i \in \{1, ..., K\}, \\ \mathbf{w} \in S_{l,u}, \end{cases} \end{aligned} \tag{4.15}$$

where $\mathbf{w}_{\mathcal{C}_i} \in \mathbb{R}^{N_i}$ is the vector of weights corresponding to the assets composing cluster $\mathcal{C}_i$ (N_i is the number of assets composing cluster $\mathcal{C}_i$). The first constraint enforces sparsity within each of the clusters, where the number of nonzero weights is constrained to less than k_i; therefore, the investment is diversified across all sectors and is sparsifed within all sectors. The second constraint ensures that the weights are distributed under the given total capitals that are allocated per cluster (b_i). The third constraint is a box constraint introduced in the previous section.

To set k_i and b_i, we first set the total number of nonzero weights in the portfolio $\mathbf{w}$, denoted by k. Then k_i is calculated by $k_i = \lfloor k\frac{N_i}{N} \rfloor$ ($\lfloor \cdot \rfloor$ indicates the floor function). The remaining $k - \sum_i k_i$ is added to the cluster with the most number of assets. Therefore, $\sum_i k_i = k$. Next, b_i is given by $b_i = k_i/k$ ($\sum_i b_i = 1$).

Furthermore, we impose a turnover sparsity constraint ($\|\mathbf{w} - \mathbf{w}_{pre}\|_1$) to the formulation to reduce transaction costs [104]. Unlike the pre-defined sectors, the calculated clusters may change their components every rebalancing. Rebalancing the portfolio to adapt to such clusters could potentially lead to an increase in the number of assets that need to be traded, thus resulting in inflated transaction costs.

Algorithm 2 PDS-based algorithm for solving (4.13)

Input: $\mathbf{r}^{\mathrm{b}}$, $\mathbf{X}$
Output: Output signal $\mathbf{w}$

Initialize $\mathbf{w} = \mathbf{0}$
while *A stopping criterion is not satisfied* **do**
  $\mathbf{w}^{(k+1)} \leftarrow P_{S_s}(\mathbf{w}^{(k)} - \gamma_1(\nabla \mathrm{TE}(\mathbf{w}^{(k)}) + \mathbf{v}_1^{(k)} + \mathbf{v}_2^{(k)}))$
  $\mathbf{v_1}^{(k)} \leftarrow \mathbf{v_1}^{(k)} + \gamma_2(2\mathbf{w}^{(k+1)} - \mathbf{w}^{(k)})$
  $\mathbf{v_2}^{(k)} \leftarrow \mathbf{v_2}^{(k)} + \gamma_2(2\mathbf{w}^{(k+1)} - \mathbf{w}^{(k)})$
  $\mathbf{v}_1^{(k+1)} \leftarrow \mathbf{v}_1^{(k)} - \gamma_2 P_{S_{l,u}}(\frac{1}{\gamma_2}\mathbf{v}_1^{(k)})$
  $\mathbf{v}_2^{(k+1)} \leftarrow \mathbf{v}_2^{(k)} - \gamma_2 P_{S_1}(\frac{1}{\gamma_2}\mathbf{v}_2^{(k)})$
  $k \leftarrow k + 1$
end whilereturn $\mathbf{w}^{(k)}$

4.3.2 Algorithm: ℓ_0-norm-based sparse index tracking

We use the primal-dual splitting method (PDS) to solve (4.13). We can use indicator functions ι_{S_s}, $\iota_{S_{l,u}}$ and ι_{S_1}, where

$$S_1 := \{\mathbf{w} \in \mathbb{R}^N \mid \mathbf{1}^\top \mathbf{w} = 1\}, \tag{4.16}$$

to reformulate (4.13) as

$$\min_{\mathbf{w}} \mathrm{TE}(\mathbf{w}) + \iota_{S_s}(\mathbf{w}) + \iota_{S_{l,u}}(\mathbf{w}) + \iota_{S_1}(\mathbf{w}). \tag{4.17}$$

By defining $\mathbf{v} := [\mathbf{v}_1^\top \ \mathbf{v}_2^\top]^\top$ $(\mathbf{v}_1, \mathbf{v}_2 \in \mathbb{R}^N)$, and $f_1, f_2, f_3, \mathbf{A}$ as

$$\begin{aligned} f_1(\mathbf{w}) &:= \mathrm{TE}(\mathbf{w}), \\ f_2(\mathbf{w}) &:= \iota_{S_s}(\mathbf{w}), \\ f_3(\mathbf{v}) &:= \iota_{S_{l,u}}(\mathbf{v}_1) + \iota_{S_1}(\mathbf{v}_2), \\ \mathbf{A} &:= \begin{bmatrix} \mathbf{I} & \mathbf{I} \end{bmatrix}^\top, \end{aligned} \tag{4.18}$$

where $\mathbf{I} \in \mathbb{R}^{N \times N}$ is an identity matrix, the problem in (4.17) is reduced to (2.12). Note that all of the tracking error measures introduced are β-Lipschitz continuous, differentiable convex functions, therefore satisfy the conditions on f_1 mentioned in Section 2.2.2. Although the PDS algorithm is guaranteed to converge when (4.17) is convex, this is not the case for our formulation because of the ℓ_0-norm constraint. Although empirically the PDS algorithm converges most of the time, we gradually restrict the stepsizes γ_1 and γ_2 to stabilize the algorithm for nonconvex optimization. This is supported by studies of ADMM and PDS algorithms for nonconvex cases, where stepsizes are diminished every iteration [41, 52, 73] or lowered in case of nonconvergence [27].

When $S_s = S_0$, the proximity operator of f_2 : $\mathrm{prox}_{\gamma\iota_{S_0}}$ is a projection[3] onto the set S_0, which is as follows:

$$\begin{aligned} [\mathrm{prox}_{\gamma\iota_{S_0}}(\mathbf{z})]_i &= [P_{S_0}(\mathbf{z})]_i \\ &= \begin{cases} z_i, \text{ if } i \in \{(1), ...(K_1)\}, \\ 0, \text{ if } i \in \{(K_1+1)...(N)\}, \end{cases} \end{aligned} \tag{4.19}$$

where we denote the elements of $\mathbf{z}$ sorted in descending order in terms of their absolute values by $z_{(1)}, ..., z_{(N)}$, i.e., $|z_{(1)}| \geq |z_{(2)}| \geq ... \geq |z_{(N)}|$. In short, a projection onto the

[3]Strictly speaking, since S_0 is a nonconvex set, the projection onto this set cannot be defined as a one-to-one mapping, but fortunately, one of the projected points can be computed analytically as in (4.25).

set S_0 can be calculated by leaving the K_1 largest absolute values and projecting the remaining elements of the vector to 0.

When $S_s = S_{\mathbf{w}_0}$, the proximity operator (Algorithm 2 line 2) is given by

$$\text{prox}_{\gamma\iota_{S_{\mathbf{w}_0}}}(\mathbf{z}) = P_{S_{\mathbf{w}_0}}(\mathbf{z}) = \mathbf{w}_0 + P_{S_0}(\mathbf{z} - \mathbf{w}_0). \tag{4.20}$$

For f_3, the proximity operator of $\iota_{S_{l,u}}(\mathbf{v}_1)$ (line 5) is given by

$$\text{prox}_{\gamma\iota_{S_{l,u}}}(\mathbf{z}) = P_{S_{l,u}}(\mathbf{z}) = [\max\{l, \min\{z_i, u\}\}]_{1\leq i\leq N}, \tag{4.21}$$

and the proximity operator of $\iota_{S_1}(\mathbf{v}_2)$ (line 6) is given by

$$\text{prox}_{\gamma\iota_{S_1}}(\mathbf{z}) = P_{S_1}(\mathbf{z}) = \mathbf{z} + \frac{1 - \mathbf{1}^\top\mathbf{z}}{\|\mathbf{1}\|_2^2}\mathbf{1}. \tag{4.22}$$

Simultaneous Asset Selection and Capital Allocation

When ETE is chosen as the tracking error measure and $S_s = S_0$, our method essentially solves the same problem as NNOMP-PGD (given that u is large enough), and the difference is reduced to the optimization algorithm. We argue that the inherent ability of our algorithm to perform asset selection and capital allocation simultaneously endows it with superior index tracking performance—a claim substantiated in the experimental section (Section 4.4.2). NNOMP-PGD, in contrast, separates capital allocation into two steps: asset selection and capital allocation. While the impact of such a process on tracking accuracy is unclear, we suspect that even if this two-step procedure could, in theory, generate an optimal portfolio, it might not apply to NNOMP-PGD.

The NNOMP phase (responsible for asset selection) of the algorithm does not impose the sum-to-one constraint. Therefore, when a nonzero value is assigned to one of the elements of the portfolio, the value of the element is not restricted in any way. The assigned value directly affects the asset selection in the following iteration, since the updated portfolio is incorporated into the residual update of the current iteration. The updated residual is used in the next iteration to select the asset.

Hence, asset selection may proceed under less-than-ideal conditions, lacking necessary constraints on values assigned in each iteration. We hypothesize that this could impair the tracking performance and that simultaneous asset selection and capital allocation might prove beneficial.

4.3.3 Algorithm: Martket graph neutrality-imposed sparse index tracking

We use the primal-dual splitting method (PDS) [24,100], an optimization method that can handle nondifferentiable terms by employing proximity operators [65], to solve (4.15). We can use indicator functions ι_{S_k}, ι_{S_b} and $\iota_{S_{l,u}}$, where $S_k := \{\mathbf{z} \in \mathbb{R}^N \mid \forall i \in \{1, 2, ..., K\}, \|\mathbf{z}_i\|_0 \leq k_i\}$ and $S_b := \{\mathbf{z} \in \mathbb{R}^N \mid \forall i \in \{1, 2, ..., K\}, \mathbf{1}^\top\mathbf{z}_i = b_i\}$, to reformulate (4.15) as

$$\begin{aligned}\min_{\mathbf{w}} \frac{1}{T}\|\mathbf{r^b} - \mathbf{X}\mathbf{w}\|_2^2 + \lambda\|\mathbf{w} - \mathbf{w}_{pre}\|_1 \\ +\iota_{S_k}(\mathbf{w}) + \iota_{S_b}(\mathbf{w}) + \iota_{S_{l,u}}(\mathbf{w}),\end{aligned} \tag{4.23}$$

a formulation that can be handled by the PDS-based algorithm (Algorithm (3)).

Algorithm 3 PDS-based algorithm for solving (4.15)

Input: $\mathbf{r^b}$, $\mathbf{X}$
Output: Output signal $\mathbf{w}$

Initialization: $\mathbf{w} = \mathbf{0}$
while *A stopping criterion is not satisfied* **do**
  $\mathbf{w}^{(n+1)} = \text{prox}_{\iota_{S_k}}(\mathbf{w}^{(n)} - \gamma_1(-\frac{2}{T}\mathbf{X}^\top(\mathbf{r^b} - \mathbf{X}\mathbf{w}^{(n)})) + \mathbf{v}_1^{(n)} + \mathbf{v}_2^{(n)} + \mathbf{v}_3^{(n)})$
  $\mathbf{v_1}^{(n)} \leftarrow \mathbf{v_1}^{(n)} + \gamma_2(2\mathbf{w}^{(n+1)} - \mathbf{w}^{(n)})$
  $\mathbf{v_2}^{(n)} \leftarrow \mathbf{v_2}^{(n)} + \gamma_2(2\mathbf{w}^{(n+1)} - \mathbf{w}^{(n)})$
  $\mathbf{v_3}^{(n)} \leftarrow \mathbf{v_3}^{(n)} + \gamma_2(2\mathbf{w}^{(n+1)} - \mathbf{w}^{(n)})$
  $\mathbf{v}_1^{(n+1)} = \mathbf{v}_1^{(n)} - \gamma_2 \text{prox}_{\frac{\lambda}{\gamma_2}\|\cdot - \mathbf{w}_{pre}\|_1}(\frac{1}{\gamma_2}\mathbf{v}_1^{(n)})$
  $\mathbf{v}_2^{(n+1)} = \mathbf{v}_2^{(n)} - \gamma_2 P_{S_b}(\frac{1}{\gamma_2}\mathbf{v}_2^{(n)})$
  $\mathbf{v}_3^{(n+1)} = \mathbf{v}_3^{(n)} - \gamma_2 P_{S_{l,u}}(\frac{1}{\gamma_2}\mathbf{v}_3^{(n)})$
  $n \leftarrow n + 1$
end whilereturn $\mathbf{w}^{(n)}$

By defining $\mathbf{v} := [\mathbf{v}_1^\top\ \mathbf{v}_2^\top\ \mathbf{v}_3^\top]^\top$ $(\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3 \in \mathbb{R}^N)$, and $f_1, f_2, f_3, \mathbf{A}$ as

$$\begin{aligned} f_1(\mathbf{w}) &:= \frac{1}{T}\|\mathbf{r^b} - \mathbf{X}\mathbf{w}\|_2^2, \\ f_2(\mathbf{w}) &:= \iota_{S_k}(\mathbf{w}), \\ f_3(\mathbf{v}) &:= \lambda\|\mathbf{v}_1 - \mathbf{w}_{pre}\|_1 + \iota_{S_b}(\mathbf{v}_2) + \iota_{S_{l,u}}(\mathbf{v}_3), \\ \mathbf{A} &:= \begin{bmatrix}\mathbf{I} & \mathbf{I} & \mathbf{I}\end{bmatrix}^\top, \end{aligned} \tag{4.24}$$

where $\mathbf{I} \in \mathbb{R}^{N\times N}$ is an identity matrix, the problem in (4.23) is reduced to (2.12).

For f_2, proximity operator $\text{prox}_{\gamma\iota_{S_k}}$ is a series of projections of $\mathbf{w}_{\mathcal{C}_i}$ onto sets $S_{k_i} := \{\mathbf{z} \in \mathbb{R}^{N_i}, |\ \|\mathbf{z}\|_0 \le k_i\}$, which is as follows:

$$[P_{S_{k_i}}(\mathbf{z})]_j = \begin{cases} z_j, & \text{if } j \in \{(1), ...(k_i)\}, \\ 0, & \text{if } j \in \{(k_i+1)...(N_i)\}, \end{cases} \tag{4.25}$$

where the suffix $[\cdot]_j$ denotes the j-th element of the given vector. We denote the elements of $\mathbf{z}$ sorted in descending order in terms of their absolute values by $z_{(1)}, ..., z_{(N_i)}$, i.e., $|z_{(1)}| \ge |z_{(2)}| \ge, ..., \ge |z_{(N_i)}|$. In short, a projection onto the set S_{k_i} can be calculated by leaving the k_i largest absolute values and projecting the remaining elements of the vector $\mathbf{w}_{\mathcal{C}_i}$ to 0. Similarly for the proximity operator of $\iota_{S_b}(\mathbf{v}_2)$ in f_3, it is a series of projections of $\mathbf{w}_{\mathcal{C}_i}$ onto sets $S_{b_i} := \{\mathbf{z} \in \mathbb{R}^{N_i}, |\ \mathbf{1}^\top\mathbf{z} = b_i\}$, which is given by

$$P_{S_{b_i}}(\mathbf{z}) = \mathbf{z} + \frac{b_i - \mathbf{1}^\top\mathbf{z}}{\|\mathbf{1}\|_2^2}\mathbf{1}. \tag{4.26}$$

The proximity operator of $\lambda\|\mathbf{v}_1 - \mathbf{w}_{pre}\|_1$ is given by

$$\text{prox}_{\lambda\|\cdot - \mathbf{w}_{pre}\|_1}(\mathbf{z}) = \mathbf{w}_{pre} + \text{prox}_{\lambda\|\cdot\|_1}(\mathbf{z} - \mathbf{w}_{pre}), \tag{4.27}$$

where $\text{prox}_{\lambda\|\cdot\|_1}(\mathbf{z})$ is equivalent to the soft-thresholding operation:

$$[\text{prox}_{\lambda\|\cdot\|_1}(\mathbf{z})]_j := \text{sgn}(z_j)\max\{0, |z_j| - \lambda\}. \tag{4.28}$$

The proximity operator of $\iota_{S_{l,u}}(\mathbf{v}_3)$ is given in (4.21).

Table 4.1: The Tracking Performance Measured in MDTE[bps] and Normalized Accumulated Returns (Ret.). Dataset: S&P500, Initial Capital: $40000. Note that LAIT is Not Included in Tables 4.1 and 4.2 due to the Difficulties Encountered in Adjusting the Sparsity to an Exact Value Using LAIT.

Method	K_2	2012 - 2017						2017 - 2022					
		$K_1 = 40$		$K_1 = 60$		$K_1 = 80$		$K_1 = 40$		$K_1 = 60$		$K_1 = 80$	
		MDTE	Ret.	MDTE	Ret.	MDTE	Ret.	MDTE	Ret.	MDTE	Ret.	MDTE	Ret.
ℓ_0-ADMM [91]	–	1.73	1.68	1.77	1.59	1.82	1.53	3.72	1.28	3.76	1.27	3.74	1.22
NNOMP-PGD [53]	–	0.85	1.71	0.79	1.64	0.75	1.60	1.43	1.38	1.21	1.49	1.06	1.50
Proposed[P, ETE]	–	0.74	1.75	0.64	1.71	0.48	1.73	1.13	1.43	0.79	1.46	0.68	1.47
Proposed[T, ETE]	K_1	**0.39**	1.87	**0.33**	1.77	**0.24**	1.72	**0.52**	1.49	**0.42**	1.39	**0.35**	1.34
	$K_1/2$	0.47	1.93	0.37	1.81	0.31	1.81	0.61	1.62	0.55	1.48	0.41	1.41
	$K_1/3$	0.51	1.89	0.41	1.81	0.32	1.80	0.66	**1.69**	0.59	**1.61**	0.46	1.47
Proposed[P, DR]	–	0.81	1.83	0.67	**1.91**	0.59	**1.95**	1.19	1.58	0.95	1.58	0.83	1.55
Proposed[T, DR]	K_1	0.42	1.92	0.35	1.76	0.31	1.76	0.64	1.49	0.46	1.36	0.40	1.36
	$K_1/2$	0.50	1.94	0.41	1.86	0.37	1.77	0.72	1.57	0.57	1.59	0.48	1.48
	$K_1/3$	0.52	**1.99**	0.44	1.89	0.39	1.86	0.84	1.67	0.59	1.50	0.56	**1.57**
Benchmark	–	–	1.38	–	1.38	–	1.38	–	1.02	–	1.02	–	1.02

4.3.4 Computational Complexity

In this section, we discuss the computational complexity of the proposed algorithm.The operations in our PDS-based algorithm for the ℓ_0-norm-based sparse index tracking (Algorithm 2) that potentially requires complex computations are the three proximity operations (projections), namely, P_{S_s}, $P_{S_{l,u}}$ and P_{S_1}, and $\nabla \mathrm{TE}$ involved in the line 2 of the algorithm. The projection P_{S_s} is composed of two steps. The sorting process can be computed in the order of $O(N \log N)$ (we use the sort() function implemented in MATLAB, which uses quicksort), and the projection process in the order of $O(N)$. For the remaining $P_{S_{l,u}}$ and P_{S_1}, both can be computed in the order of $O(N)$. Gradient $\nabla \mathrm{TE}$ can be computed in the order of $O(NT)$. Therefore, the overall computational complexity of the proposed algorithm per iteration can be simplified to $O(NT)$, since T is generally large enough such that other operations can be disregarded.

For the computational complexity of the market graph neutrality-imposed tracking algorithm (Algorithm 3), the main difference between the basic sparse index tracking complexity-wise is the addition of the turnover regularization ($\mathrm{prox}_{\lambda\|\cdot-\mathbf{w}_{pre}\|_1}(\mathbf{z})$) to the objective function. Since its computational complexity is $O(N)$, the overall computational complexity of the proposed algorithm per iteration can be simplified to $O(NT)$, as in the basic sparse index tracking algorithm. We should take into consideration the computational cost of performing the market graph clustering, which is performed prior to the optimization process. The bottleneck of the clustering process lies in the eigenvalue decomposition of the market graph Laplacian matrix, which has a computational complexity of $O(N^3)$, where N is the number of assets. However, the clustering process is required only once unlike the iterative optimization process. In practice, the required number of iterations for the optimization process is generally much larger than the number of assets, and through our experiments, we have confirmed that the computational complexity of $O(N^3)$ is comparable to the computational complexity of the full optimization process. Therefore, the overall computational complexity is not bottlenecked by the clustering process.

4.4 Experiments

4.4.1 Dataset and Settings

To test the performance of the index tracking methods, we adopt the rolling window scheme [9,53]. The first T_{train} time-frames are used to design the first portfolio, which will

Table 4.2: The Tracking Performance Measured in MDTE[bps] and Normalized Accumulated Returns (Ret.). Dataset: Russell3000, Initial Capital: \$40000. Note that LAIT is Not Included in Tables 4.1 and 4.2 due to the Difficulties Encountered in Adjusting the Sparsity to an Exact Value Using LAIT.

Method	K_2	2010 - 2014						2015 - 2019					
		$K_1 = 100$		$K_1 = 150$		$K_1 = 200$		$K_1 = 100$		$K_1 = 150$		$K_1 = 200$	
		MDTE	Ret.	MDTE	Ret.	MDTE	Ret.	MDTE	Ret.	MDTE	Ret.	MDTE	Ret.
ℓ_0-ADMM [91]	–	3.21	2.11	3.21	2.01	3.20	1.97	2.44	1.64	2.48	**1.70**	2.49	1.60
NNOMP-PGD [53]	–	0.78	1.78	0.75	1.65	1.89	1.69	1.07	1.59	1.03	1.52	1.62	1.48
Proposed[P, ETE]	–	0.83	**2.28**	0.65	**2.20**	0.54	1.92	0.88	1.53	0.68	1.47	0.58	1.57
Proposed[T, ETE]	K_1	**0.51**	1.97	**0.42**	1.85	**0.33**	1.81	**0.55**	**1.70**	**0.40**	1.56	**0.35**	1.60
	$K_1/2$	0.56	1.92	0.47	1.80	0.37	1.79	0.61	1.68	0.46	1.64	0.40	1.60
	$K_1/3$	0.60	1.96	0.51	1.78	0.40	1.76	0.63	1.68	0.48	1.63	0.43	**1.67**
Proposed[P, DR]	–	1.20	2.25	0.99	2.07	0.86	1.96	1.11	1.64	0.84	1.57	0.71	1.43
Proposed[T, DR]	K_1	0.58	1.99	0.44	2.00	0.36	1.90	0.62	1.55	0.49	1.55	0.41	1.46
	$K_1/2$	0.64	2.04	0.50	2.07	0.41	1.94	0.64	**1.70**	0.55	1.63	0.42	1.48
	$K_1/3$	0.71	2.02	0.51	2.00	0.44	**1.99**	0.69	1.60	0.57	1.57	0.44	1.48
Benchmark	–	–	1.37	–	1.37	–	1.37	–	1.18	–	1.18	–	1.18

Table 4.3: The Comparison Between Different Initialization Methods, Tracking Performance Measured in MDTE[bps] and Normalized Accumulated Returns (Ret.). Dataset: S&P500, Initial Capital: \$10000, $K_1 = 40$.

Method	K_2	S&P500 (2012 - 2017)						Russell3000 (2010 - 2014)					
		Init. A		Init. B		Init. C		Init. A		Init. B		Init. C	
		MDTE	Ret.	MDTE	Ret.	MDTE	Ret.	MDTE	Ret.	MDTE	Ret.	MDTE	Ret.
Proposed[P, ETE]	–	0.74	1.75	0.58	2.03	0.61	1.93	1.13	1.43	0.76	1.65	0.78	1.78
Proposed[T, ETE]	K_1	0.39	1.87	0.37	1.85	0.42	1.83	0.52	1.49	0.46	1.40	0.51	1.63
	$K_1/2$	0.47	1.93	0.43	1.93	0.52	1.89	0.61	1.62	0.57	1.46	0.67	1.63
	$K_1/3$	0.51	1.89	0.46	1.96	0.58	1.77	0.66	1.69	0.61	1.53	0.73	1.71
Proposed[P, DR]	–	0.81	1.83	0.63	2.08	0.64	2.03	1.19	1.58	0.90	1.68	0.86	1.55
Proposed[T, DR]	K_1	0.42	1.92	0.35	1.86	0.44	1.88	0.64	1.49	0.50	1.49	0.63	1.56
	$K_1/2$	0.50	1.94	0.41	1.98	0.50	1.90	0.72	1.57	0.61	1.60	0.78	1.67
	$K_1/3$	0.52	1.99	0.46	1.95	0.54	1.99	0.84	1.67	0.62	1.65	0.79	1.60
Benchmark	–	–	1.38	–	1.38	–	1.38	–	1.02	–	1.02	–	1.02

be used for out-of-sample testing in the next T_{test} time-frames. At the end of the testing period, we use the last T_{train} time-frames to design the next portfolio. We continue the training and testing cycle n times. Therefore, a total of $T_{\text{train}} + nT_{\text{test}}$ time-frames are used for each experiment. The parameters are set to $n = 10$, $T_{\text{train}} = 200$, and $T_{\text{test}} = 100$.

We use the S&P500 (September 2012 - August 2022) and Russell3000 (January 2010 - December 2019) index datasets, commonly used datasets for index tracking [9, 53] for the experiments. Assets not covering the entire period are excluded, resulting in final datasets comprising 463 and 1624 assets for the S&P500 and Russell3000, respectively. The adjusted closing prices of the assets are used, and the return of an asset j at time-frame t is given by

$$\mathbf{X}_{t,j} = \frac{\text{price}_{t,j} - \text{price}_{t-1,j}}{\text{price}_{t-1,j}}. \tag{4.29}$$

We split both S&P500 and Russell3000 index datasets into two, September 2012 - October 2017 and November 2017 - August 2022 (S&P500), and January 2010 - October 2014 and March 2015 - December 2019 (Russell3000), respectively. Note that each time-frame represents a trading day, and because the stock exchange is closed on weekends, the dataset is not sampled regularly in the time direction.

We measure how well the portfolio replicates the benchmark index by computing the magnitude of the daily tracking error (MDTE) defined as

$$\text{MDTE} = \frac{1}{nT_{\text{test}}} \| \operatorname{diag}(\mathbf{X}\mathbf{W}) - \mathbf{r}^{\mathbf{b}} \|_2, \tag{4.30}$$

Table 4.4: The Tracking Performance Measured in MDTE[bps] and Sharpe Ratio. Dataset: S&P500, Initial Capital: $10000.

Time period	Metric	Turnover	Sector neutrality		Market-graph neutrality					
			LAIT [8]	Ours	EW [30] $K = 11$	Ours $K = 7$	Ours $K = 9$	Ours $K = 11$	Ours $K = 13$	Ours $K = 15$
2012-2017	MDTE	w/o	0.60	1.07	**0.33**	1.23	0.95	1.08	1.07	1.23
		w/	-	0.86	-	0.77	0.75	0.75	0.72	0.70
	Sharpe ratio	w/o	1.21	0.73	0.60	0.65	0.95	0.87	0.78	0.88
		w/	-	0.99	-	**1.30**	1.16	1.20	1.18	0.98
2017-2022	MDTE	w/o	0.78	1.68	**0.68**	1.74	1.55	1.53	1.78	1.80
		w/	-	1.78	-	1.77	1.55	1.37	1.75	1.60
	Sharpe ratio	w/o	0.28	0.31	0.14	**0.37**	0.29	0.28	0.29	0.31
		w/	-	0.30	-	0.30	0.35	0.34	0.34	**0.37**

where $\mathrm{diag}(\cdot)$ indicates a vector consisting of the diagonal elements of a given matrix. $\mathbf{W} \in \mathbb{R}^{N \times nT_{\text{test}}}$ is a matrix where the portfolio designed at the end of a training period is stacked for the following test period, so that column t contains the portfolio used in time-frame t. Note that the MDTE value is presented in basis points ($1\text{bps} = 10^{-4}$). Portfolio risk is evaluated by the *Sharpe ratio* (i.e., the ratio of the mean to the standard deviation of portfolio returns, the higher the better). Since we could not obtain the asset-wise weights from the index sponsors, we use a uniform portfolio $\mathbf{b}$ as a benchmark index, where all of the capital is allocated evenly across all N assets. The benchmark index return is given by $\mathbf{r}^{\mathbf{b}} = \mathbf{X}\mathbf{b}$.

Furthermore, we conduct a simulation of actual investments made based on the computed portfolios. We simulate rebalancing based on a new portfolio at the start of every test period. The acquired returns are reinvested. We apply a transaction costs model common in the U.S. markets: the cost per transaction is $\$0.005 \times v$ with a minimum cost of $1. The Ret. columns of Tables 4.1 and 4.2 show the normalized accumulated return at the end of the entire testing period.

As for the comparison methods, in addition to the aforementioned NNOMP-PGD [53] and LAIT [9], the state-of-the-art sparse index tracking methods, we also compare our method with ℓ_0-ADMM (Alternating Direction Method of Multipliers) [91]. Despite how ℓ_0-ADMM focuses on optimizing between return and risk, making it distinct from typical index tracking approaches, its implementation of an ℓ_0-norm constraint mirrors that of NNOMP-PGD and our proposed method, warranting its inclusion in our comparison. Regarding our methods, portfolio-sparse methods (Proposed[P, TE]) allocate the capital from scratch every rebalancing, in other words, $S_s = S_0$. For turnover-sparse methods (Proposed[T, TE]), the turnovers are sparsified instead of the portfolio ($S_s = S_{\mathbf{w}_0}$) with K_2 set to various values. Note that turnover-sparse methods adopt $S_s = S_0$ for the first training period since there is no previous portfolio to refer to. We also evaluate the performance of different tracking error measures (ETE and DR).

Regarding the other settings of the experiment, the stopping criterion is set to

$$\frac{\|\mathbf{w}^{(k)} - \mathbf{w}^{(k-1)}\|_2}{\|\mathbf{w}^{(k-1)}\|_2} \leq 1.0 \times 10^{-5}. \tag{4.31}$$

For the box constraint in (4.13) and (4.15), the lower and upper bounds are set to $l = 0, u = 4/K_1$. The stepsizes γ_1 and γ_2 of the proposed algorithm are first set to sufficiently meet the conditions mentioned in Section 2.2.2, and then multiplied by 0.999 every iteration.

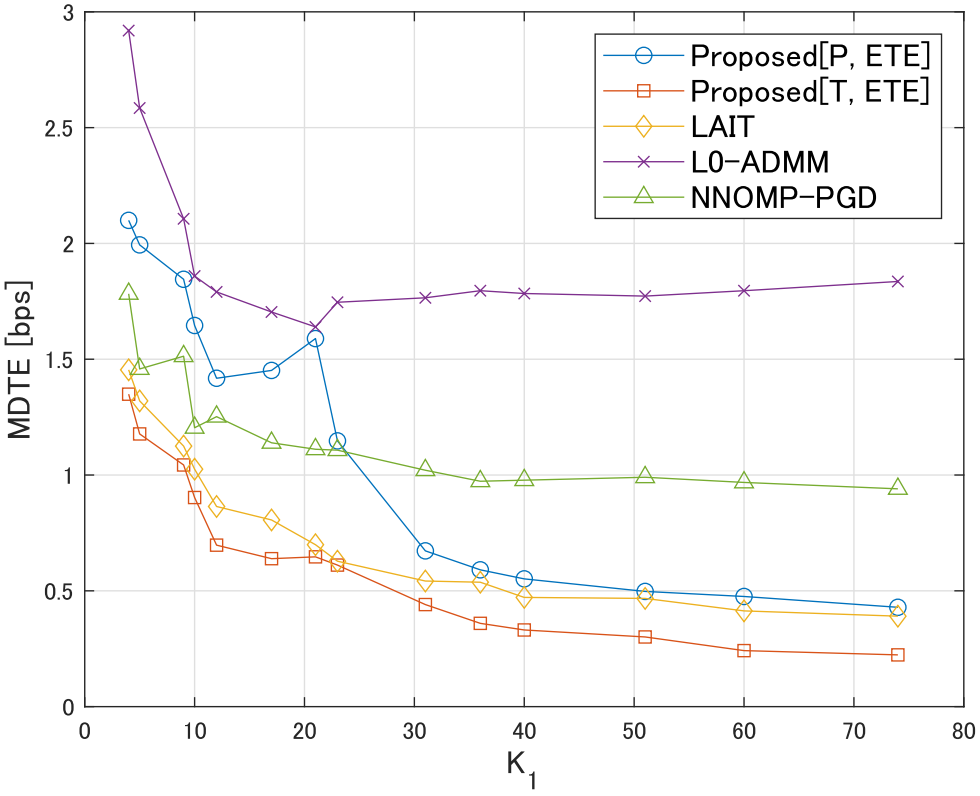


Figure 4.1: The graph of MDTE[bps] across different sparsity on S&P500, 2012 - 2017. The vertical axis indicates MDTE[bps] and the horizontal axis indicates the sparsity. To avoid parameter tuning on λ (which controls sparsity in LAIT), we fixed the value of λ per data point. Therefore, the exact sparsity of per training period differs. The sparsity of Proposed[P, ETE] (K_1), NNOMP-PGD and ℓ_0-ADMM is adjusted to be the same as that of LAIT. As for Proposed[T, ETE], $K_2 = K_1$. Parameter K_1 in the graph indicates the sparsity of the portfolio at $n = 10$.

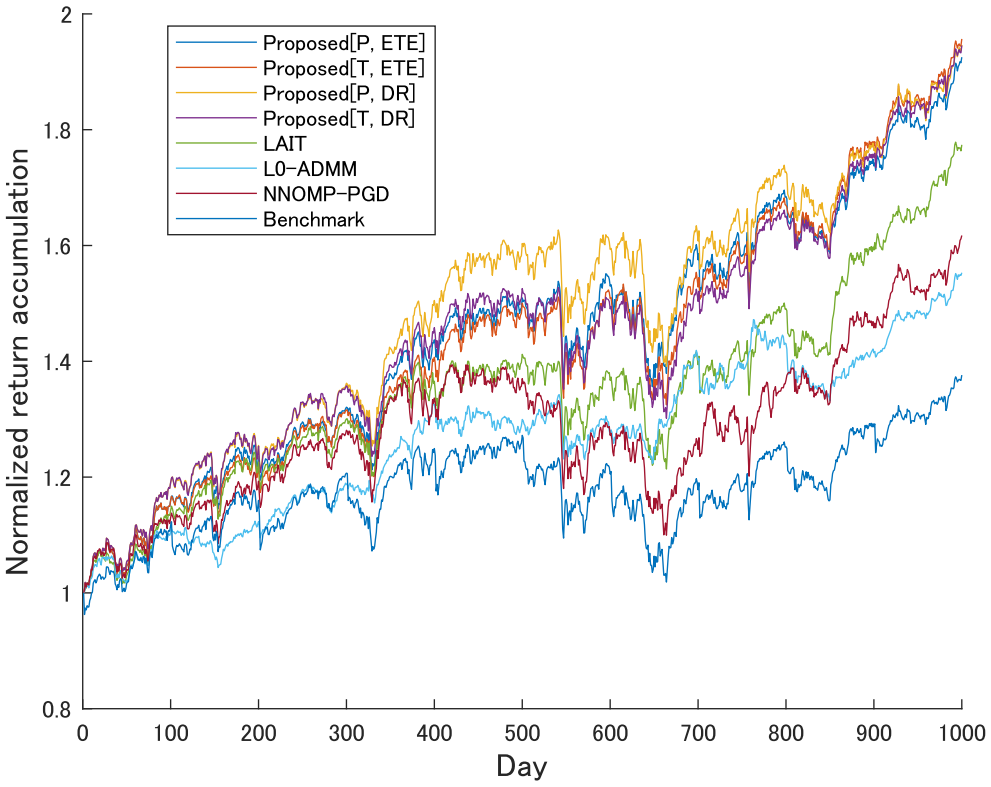
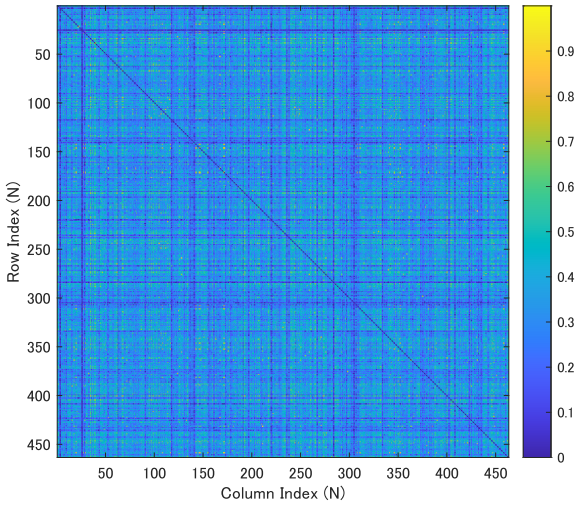
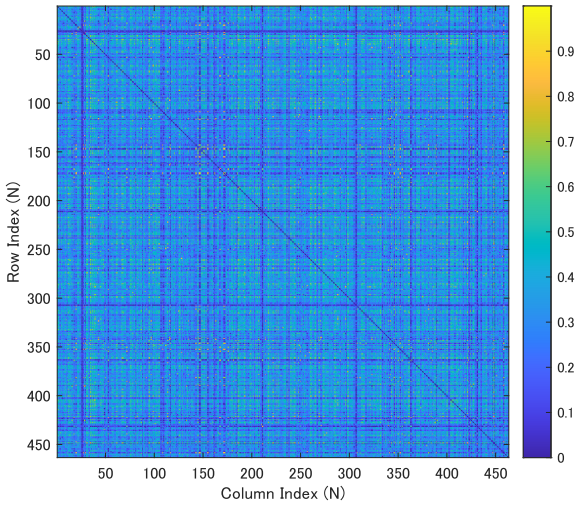


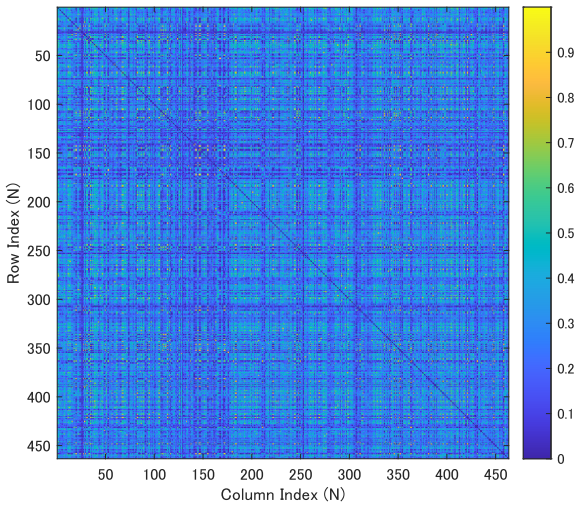
Figure 4.2: The graph of the investment simulation on S&P500, 2012 - 2017. The vertical axis indicates the normalized accumulated return and the horizontal axis indicates the time period. The sparsity of Proposed[P, ETE] (K_1), NNOMP-PGD and ℓ_0-ADMM is adjusted to be the same as that of LAIT. As for Proposed[T, ETE], $K_2 = K_1/3$. The graph indicates an investment simulation based on one of the data points of Fig. 4.1. The initial capital is \$10000.



$n = 1$



$n = 5$



$n = 10$

Figure 4.3: The visual presentation of the dynamics of the correlation matrix used for market graph clustering via heatmaps.

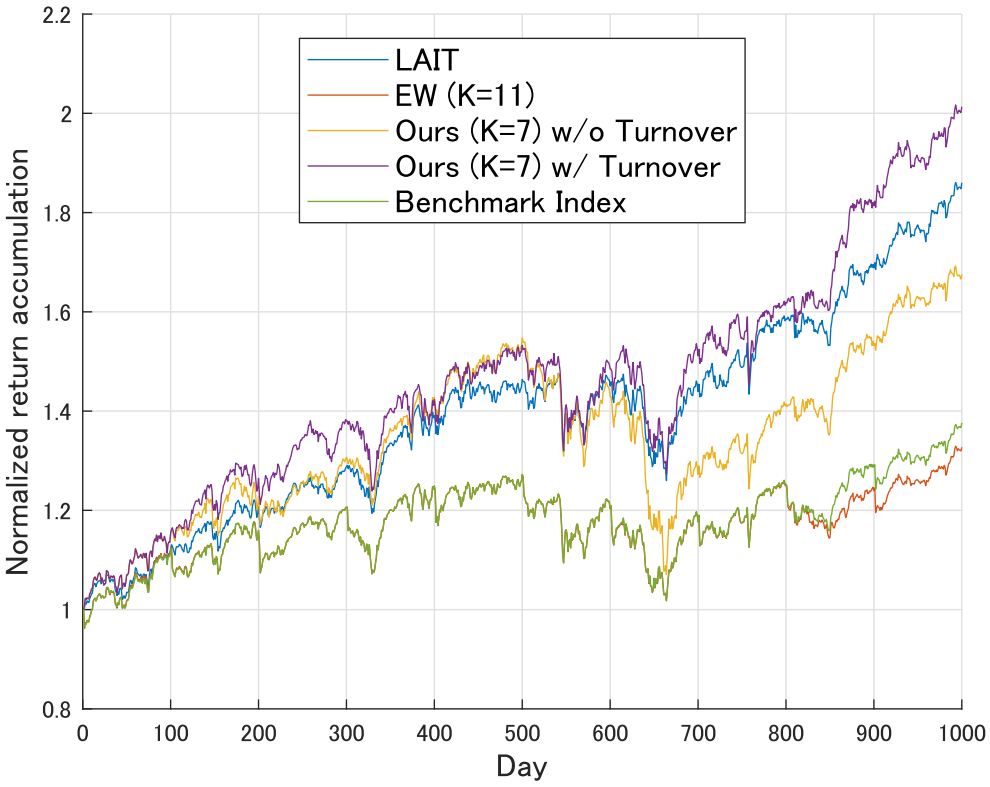


Figure 4.4: The graph of the investment simulation on S&P500, 2012 - 2017. The vertical axis indicates the normalized accumulated return and the horizontal axis indicates the time period. The initial capital is \$10000, and the applied cost per transaction is \$0.005 $\times$ tradevolume with a minimum cost of \$1.

4.4.2 Results

Tracking Performance

We first evaluate the tracking performance of the proposed ℓ_0-norm-based sparse index tracking method compared to the conventional tracking methods.

Across all evaluated settings, our proposed method surpassed both ℓ_0-ADMM and NNOMP-PGD in performance on the S&P500 and Russell3000 datasets, as detailed in Tables 4.1 and 4.2 (MDTE). This supports our hypothesis that concurrent asset selection and capital allocation can be advantageous. Both Proposed[P, ETE] and NNOMP-PGD strategies utilize similar ℓ_0-norm constraints and tracking error measures. The primary difference between them lies in the algorithms used for solving these constraints. Note that LAIT is not included in Tables 4.1 and 4.2 due to the difficulties encountered in adjusting the sparsity to an exact value using LAIT. Regarding LAIT (Fig. 4.1), while the turnover-sparse (Proposed[T, ETE]) method achieved better tracking performance, the portfolio-sparse (Proposed[P, ETE]) method slightly underperformed LAIT. The superior performance of LAIT likely stems from its guaranteed convergence to an optimal solution. The proposed portfolio-sparse method's slight underperformance should be considered as a trade-off between its ability to directly control the sparsity of the portfolio.

Furthermore, comparing the tracking performance of portfolio-sparse (Proposed[P, ETE]) methods and turnover-sparse (Proposed[T, ETE]) methods, we can see that turnover-sparse methods always performed better. We believe this is because portfolios constructed using turnover-sparse methods are composed of more nonzero weights compared to those of portfolio-sparse methods. Because turnover-sparse methods only enforce the sparseness of the turnover, the sparsity of the portfolio itself is not considered. Therefore, as the simulation progresses, the portfolio gradually becomes fuller. A fuller portfolio can replicate the target index with more nonzero weights, which should affect the tracking performance positively. Turnover-sparse methods with $K_2 = K_1$ tracked the index more effectively than other K_2 values did for the same reason: larger K_2 values densify the portfolio faster than smaller K_2 values. Similarly, a larger K_1 performed better for portfolio-sparse methods since portfolios constructed from larger K_1 values have more nonzero weights.

Next, we evaluate the tracking performance of the market graph neutrality-imposed sparse index tracking (Table 4.4). It should first be noted that the tracking performance (MDTE) was generally worse compared to the conventional methods (LAIT [8] and EW [30]). In comparison with LAIT, the inferior tracking performance is due to our method imposing much more harsh constraints than LAIT. Since the imposed constraints deprive the method's ability to freely choose the assets, the ability to neutralize the portfolio and the ability to track the index are in relation to a trade-off. In fact, how little the tracking performance has degraded with and without sector neutrality for LAIT (4.1 and 4.4) emphasizes how poorly sector neutrality is imposed in LAIT. As for the fact that our method was outperformed by EW, this is due to the benchmark index being constructed from a full and uniform portfolio (EW is also full).

Return Accumulation

Our proposed methods outperformed the benchmark index and other comparative methods in terms of return accumulation across most settings, as detailed in Tables 4.1 and 4.2 (Ret.). It is also clear that sparse portfolios are much more efficient compared to benchmark (full) portfolios (Fig. 4.2). However, regarding comparisons between portfolio-sparse and turnover-sparse methods and the various tracking measures, we acquired varying results depending on the dataset and period.

First, when we applied the empirical tracking error (ETE) as the tracking measure, turnover-sparse methods (Proposed[T, ETE]) generally performed better than portfolio-

sparse methods (Proposed[P, ETE]). This seems intuitive since turnover-sparse methods sparsify the turnover, which directly affects the transaction costs. In the worst-case rebalancing scenario, portfolios constructed by portfolio-sparse methods sell all assets in possession and newly buy completely different assets. In this case, the transaction cost would be $\$2K_1$ (assuming the minimum cost is dominant). In comparison, the worst-case scenario for portfolios constructed by turnover-sparse methods is always K_2.

However, results became more varied and inconsistent when applying the downside risk (DR) as the tracking measure. At present, we lack a definitive explanation for these inconsistencies, though we speculate they may stem from the challenges associated with the nonconvex nature of the formulation, a topic we further discuss in Section 4.4.2. In addition to the above behavior, DR-applied methods performed moderately in comparison to ETE-applied methods. We expected DR to accumulate more returns than ETE, but there are some results where ETE outperformed the DR counterpart. While DR is designed to trade tracking accuracy for potentially higher returns, our results indicate that this trade-off does not consistently yield the expected benefits.

Portfolio Neutrality

Here, we discuss the method's ability to reduce investment risks by imposing sector and market graph neutrality constraints.

- Market-graph neutrality achieved a higher Sharpe ratio compared to sector neutrality (Ours with sector neutrality vs Ours with market-graph neutrality (Table 4.4)). Both methods construct portfolios based on the proposed formulation; therefore, the comparison is a genuine comparison between sector and market-graph neutrality.
- The proposed framework (market-graph neutrality with turnover sparsity) achieved a higher Sharpe ratio compared to benchmark methods (LAIT vs market-graph neutrality with turnover sparsity (Table 4.4)).

We should also address the fact that our methods without turnover sparsity did not outperform LAIT in the 2012-2017 time period, which at first glance may imply the inferiority of our formulation compared to LAIT. It should be noted that LAIT constructs portfolios based on much looser conditions (refer (4.4)) compared to our formulation. The main differences are: 1) portfolio sparsity and sector neutrality are imposed by a regularization, and 2) the constraint on capital allocation (the sum-to-one constraint) is imposed on the entire portfolio and not on each of the sectors. Therefore, investments made by portfolios constructed by LAIT are not strictly diversified across all sectors (e.g., there were cases where some sectors were not invested in at all). Even in the case where the nonzero elements are evenly distributed across the sectors, since there are no sector-wise constraints on capital allocation, there is no way to prevent biased investment. Consequently, we speculate that the high Sharpe ratio was achieved based on higher returns gained by concentrated investment in high-performing assets. However, such portfolios are naturally more risky, which can also be asserted by the fact that Ours without turnover was able to outperform LAIT in the 2017-2022 dataset, which was a time period more difficult to accumulate returns by index tracking (lower Sharpe ratio compared to 2012-2017). This also supports the advantages of the proposed market-graph neutrality & turnover-sparsity method since it can accumulate more returns without giving up the trade-off between portfolio performance and diversity.

Furthermore, it should be noted that our method allows the choice of K, namely, the number of clusters. The parameter K defines how the investment is diversified, and it completely disregards portfolio neutrality when $K = 1$. Where the use of conventional sectors only allows the choice between whether to diversify the investment or not, our

framework allows the investor to control the trade-off between pursuing higher returns or reducing the risk by diversifying the investment.

By referring to Fig. 4.3, we can visually confirm the dynamics of the correlation matrix used for market graph clustering. The heatmaps show the correlation matrix at different time periods, and we can observe how the correlation between the assets changes over time. The clustering of the assets based on the correlation matrix is expected to change over time, and by diversifying our investment across these clusters, we can effectively leverage the dynamics of the market.

Initialization

Given the nonconvex nature of our formulation, convergence cannot be guaranteed by our proposed algorithm. Therefore, the method of initializing the parameter $\mathbf{w}$ may potentially impact the results. To understand how initialization impacts our algorithm's performance, we evaluated three distinct initialization strategies for the ℓ_0-norm-based sparse index tracking algorithm: Init. A: $\mathbf{w} = \mathbf{0}$, the original initialization adopted in our algorithm (Algorithm 2). Init. B: initialize all elements of $\mathbf{w}$ to $1/N$. Init. C: initialize $\mathbf{w}$ as $\mathbf{w} = \mathbf{w}_0$ ($\mathbf{w} = \mathbf{0}$ for the first training period).

The varied outcomes, as detailed in Table 4.3, underscore the effect of the chosen initialization method on the algorithm's performance. Given the inconsistency of results across different datasets, identifying a universally superior initialization method proved challenging. It is noteworthy, however, that our method consistently outperformed NNOMP-PGD across the majority of settings, irrespective of the initialization technique employed.

Furthermore, we present the convergence behavior of our algorithm in Fig. 4.5. Our algorithm converged in a similar manner across all experimental settings. Despite lacking a formal convergence guarantee, the observed convergence behavior and compelling tracking performance allow us to conclude that our method is capable of generating competitive portfolios, which is on par with methods like LAIT that guarantee convergence.

Summary of the Experimental Results

Through extensive numerical experiments on S&P500 and Russell3000 datasets, we have:

- Confirmed our hypothesis that simultaneous asset selection and capital allocation can be beneficial in terms of tracking accuracy.
- Presented the merits of turnover sparsity in both index tracking and wealth accumulation.
- Confirmed the advantages of market-graph neutrality over sector neutrality.
- Analyzed the convergence behavior and how various initialization methods influence the effectiveness of the proposed algorithm.

4.5 Concluding Remarks

In this chapter, we proposed a sparse index tracking method that addressed both asset selection and capital allocation simultaneously, based on an ℓ_0-norm constraint, enhancing tracking performance compared to the conventional method that handled these two aspects separately. Building on this, we introduced a novel sparse index tracking framework that used market-graph neutrality instead of conventional sector neutrality. Through extensive experiments, we demonstrated the effectiveness of our ℓ_0-norm-constraint-based formulation and the benefits of market-graph neutrality.

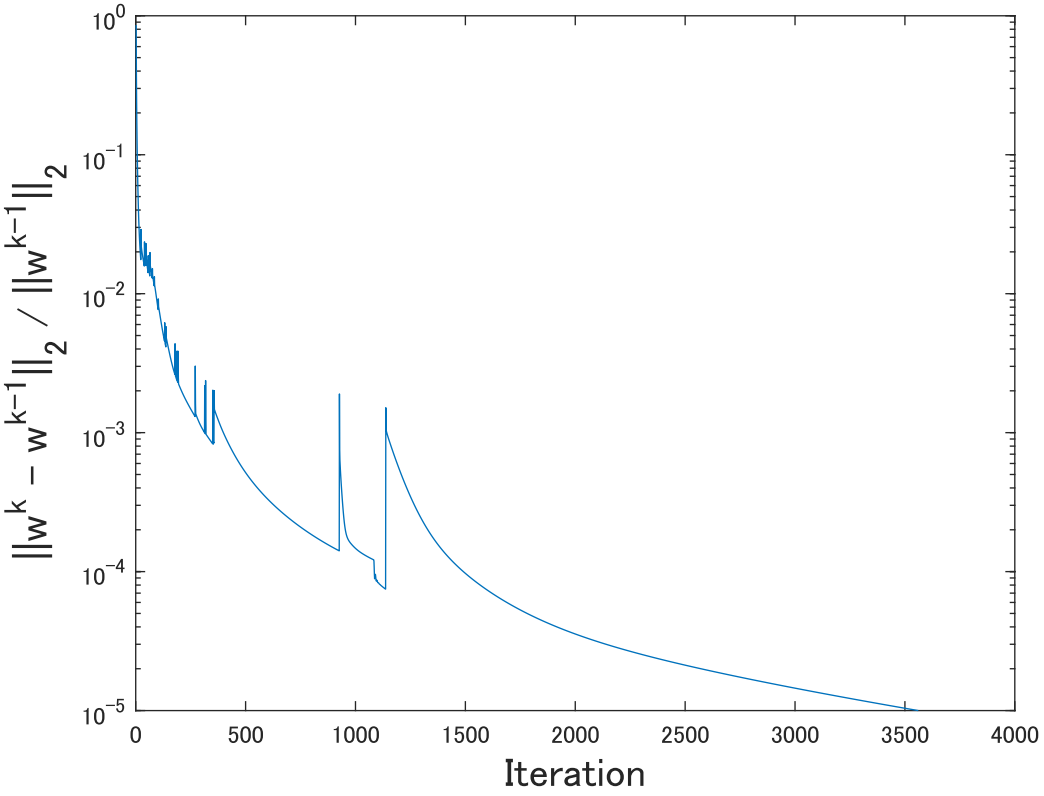


Figure 4.5: The graph of the proposed algorithm's ([P,ETE]) convergence behavior on the S&P500 (2012 - 2017) dataset ($K_1 = 40$, Init. A). The vertical axis indicates $\frac{\|\mathbf{w}^{(k)} - \mathbf{w}^{(k-1)}\|_2}{\|\mathbf{w}^{(k-1)}\|_2}$ and the horizontal axis indicates the number of iterations.

Chapter 5

General Conclusion

This dissertation has explored the methodology of leveraging the dynamics of the graph topology in time-varying graph signal estimation. By examining how to expand the conventional static graph assumptions to dynamic graph assumptions in various domains, we have addressed the following research question: *How can we exploit the dynamics of the graph topology to enhance time-varying graph signal estimation in situations where the underlying graph topologies are explicitly dynamic?* Specifically, we have introduced approaches to exploit the dynamic nature of the graph by constructing time-varying graphs and incorporating them into the estimation process. Through experiments in two different domains (environmental monitoring and finance), we have demonstrated the effectiveness of dynamic graph assumptions over static ones.

In Chapter 3, we extended the conventional problem setting, in which the sensor network was assumed to be static, to a scenario where the sensor network is explicitly dynamic. We achieved this by formulating an optimization problem that can handle a dynamic graph. To evaluate our method, we constructed a synthetic dataset that accurately reflects the characteristics of a dynamic sensor network. Through extensive experiments on both the synthetic dataset and real-world datasets, we confirmed the advantages of dynamic graph representations over static graph representations in time-varying graph signal recovery.

In Chapter 4, we proposed a method for reducing investment risks in sparse index tracking by introducing a notion called *market graph neutrality*. By frequently redefining new groups through graph clustering on the dynamic market graph, we achieved a grouping of assets that is updated in real-time to reflect the market's dynamics. This is an extension of the conventional sector-based grouping, which can be interpreted as static. Furthermore, to enable effective sparse index tracking, we introduced an ℓ_0-norm-constrained tracking method that allows for easy control of the number of non-zero weights by adjusting a single parameter. This served as the foundation for formulating the market-graph-neutrality-based sparse index tracking method. Through experiments on real-world financial data, we demonstrated the effectiveness of our method in reducing investment risks and achieving higher returns while maintaining high tracking accuracy.

5.1 Limitations

As described in the previous section, this dissertation explores leveraging the dynamics of the graph topology in time-varying graph signal estimation. However, there are several limitations to this study that must be addressed in future research. In this section, we discuss these limitations and propose directions for future work.

5.1.1 Graph Construction for Inexplicit Cases

In both Chapter 3 and Chapter 4, we assumed that the graph topology is either explicitly known or can be easily constructed based on available information, such as physical coordinates or correlations between price movements. This is a reasonable assumption because physical coordinates are often available in physical sensing scenarios, and non-physical settings often allow us to construct graphs from uncorrupted data. However, there are two cases where graph construction is not straightforward.

First, in some situations, the physical coordinates themselves constitute the observations (e.g., 3D point cloud data). In tasks such as 3D point cloud signal denoising or recovery, the observed signals (the sensor coordinates) may be corrupted by various types of noise. In these cases, graph construction from coordinates or uncorrupted data is not possible, demanding more sophisticated methods. Possible solutions include applying graph learning techniques that can handle corrupted data or initially denoising the data without spatial priors and then constructing the graph from the partially denoised data for subsequent estimation.

Next, because we avoid overly sophisticated graph construction methods (e.g., graph learning) in our study, the edge weights must be defined heuristically. If we leverage a smoothness prior to physical sensing scenarios, a natural choice would be a Gaussian kernel or the inverse of the Euclidean distance. However, this is not always the case, and defining such weights can be challenging when no obvious definition exists. This challenge is closely related to one of the limitations of the optimization-based approach—namely, the need to handcraft a formulation that sufficiently captures the meaningful priors in the data. To effectively leverage spatial priors for signal estimation, constructing a graph that reflects the data's characteristics is essential.

5.1.2 Variations in Graph Dynamics

Our studies extend static graph representations to dynamic ones to capture changes in the graph topology more effectively. However, our assumption regarding graph dynamics focuses on edges and weights. For example, in Chapter 3, the problem setting involved moving sensors that change position over time, making the graph dynamic. This assumption excludes other potential changes to the network, such as adding or removing sensors. Similarly, in Chapter 4, the turnover prior was introduced under the assumption that the same assets remain in the index from one-time instance to the next. Our proposed methods cannot handle such alterations in graph attributes and would require splitting the data before and after each change, applying our methods separately.

Another type of graph dynamic involves irregular structures in the temporal dimension of the time-varying graph. In our studies, we assumed a constant update frequency, i.e., the graph is updated at fixed intervals. Although this assumption is reasonable in many cases, there are scenarios where the graph is updated irregularly. For instance, in a network of drone-based sensors, data collection might pause for recharging or refueling at irregular intervals. Ideally, we want a method that can handle data observed before and after such intervals seamlessly; however, our methods are not designed to manage such temporal irregularities.

5.2 Future Works

5.2.1 Introducing Prior Knowledge of Graph Dynamics

As noted in Section 5.1.2, our work is limited in its ability to handle a wide range of graph dynamics. Although we have extended static graph representations to dynamic

ones in both Chapter 3 and Chapter 4, we have not fully explored the possibility of incorporating priors related to graph dynamics, such as changes in the set of sensors, evolving relationships among them, or irregularities in update intervals. By incorporating such priors, we can develop more sophisticated methods that accommodate a broader range of graph dynamics.

5.2.2 Consideration of Irregular Structures in the Temporal Domain

Our work applies graph signal processing to time-varying data while leveraging graph dynamics. This approach effectively captures the irregular structure of the data in the spatial domain. To widen the scope of our methods, we could extend these concepts to address irregular structures in the temporal domain as well.

In this regard, multiplex and multilayer graph signal processing [107, 108] hold great potential. They model the temporal dimension as additional layers of connectivity, effectively creating a 3D graph. Unlike conventional time-varying graph signal processing, which treats time as a series of discrete, uniformly sampled snapshots (2D graphs), multiplex and multilayer approaches represent temporal relationships as weighted edges in another dimension. If those temporal edges are constant, we revert to conventional time-varying graph processing. By leveraging multiplex and multilayer frameworks, we can manage irregular temporal structures more effectively.

5.2.3 Range of Applications

In Chapter 3 and Chapter 4, we demonstrated our methods in both physical (environmental monitoring) and non-physical (finance) scenarios. Although the philosophy behind leveraging graph topology dynamics is universal, our specific methods are limited to time-varying graph signal recovery and sparse index tracking. We have yet to establish a generalized framework applicable to a wider range of tasks. As future work, we envision expanding these methods to additional domains—such as traffic management, smart cities, or bandwidth optimization—where dynamic graph modeling can provide substantial benefits. By doing so, we can move toward a more versatile framework that applies to a broader range of applications.

5.3 Closing Remarks

We have outlined the limitations of our current study and highlighted directions for future research to advance time-varying graph signal estimation over dynamic graphs. We conclude with a brief reflection on the implications of our research.

Our work explored the methodology of leveraging graph topology dynamics in time-varying graph signal estimation. Dynamic graph scenarios have often been overlooked in time-varying graph signal processing; thus, our contributions significantly advance this field by illuminating the advantages of dynamic over static graph representations. With the rapid development of sensor technology and data-collection methods, vast datasets are now representable as time-varying graph signals, and demand for effective methods to handle these data will continue to grow. Our research establishes a foundation for developing such methods and opens new avenues for future work in this domain.

Acknowledgment

This dissertation is the final product of my time as a Ph.D. student. I would like to take this opportunity to express my deepest gratitude to the people who have supported me throughout my doctoral life.

First and foremost, I would like to express my sincere gratitude to my advisor, Associate Professor Shunsuke Ono. Without his guidance, I would not have been able to aspire toward obtaining a Ph.D. degree. His teachings compose the foundations of my research skills, mindset, and values.

Next, I would like to express my heartfelt gratitude to my colleague Kazuki Naganuma. His passion and dedication to research have always been a great source of inspiration to me, and our fondest of friendship has and always will be a great support to me.

I am also grateful to Shino Ogata for her invaluable help with all the administrative tasks at Science Tokyo.

I also thank all the members of my laboratory for their warm friendship.

I would like to express my deep gratitude to Professor Isao Ono, Professor Misako Takayasu, Professor Tsuyoshi Murata, and Professor Rio Yokota, Science Tokyo, Japan, for serving as members of the examining committee of this dissertation.

Without the support of my family and friends, I would not have been able to complete this dissertation. I am deeply grateful for their unwavering support.

Finally, I would like to express my greatest gratitude to Sara Hoshikawa, whose streams provided the central source of motivation and encouragement during the most challenging times of my doctoral life.

This work was supported by Grant-in-Aid for the Japan Society for the Promotion of Science (JSPS) Fellows (DC2).

References

[1] Carol A. Optimal hedging using cointegration. *Philos. Trans. R. Soc. A Math. Phys. Eng. Sci.*, 357:2039–2058, 08 1999.

[2] M. Afonso, J. Bioucas-Dias, and M. Figueiredo. An augmented Lagrangian approach to the constrained optimization formulation of imaging inverse problems. *IEEE Trans. Image Process.*, 20(3):681–695, 2011.

[3] H. Akaike. Fitting autoregressive models for prediction. *Ann. Inst. Stat. Math.*, 21:243–247, 1969.

[4] A. Arroyo, B. Scalzo, L. Stanković, and D. P Mandic. Dynamic portfolio cuts: A spectral approach to graph-theoretic diversification. In *Proc. 2022 IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, pages 5468–5472. IEEE, 2022.

[5] B. M Barber and T. Odean. Trading is hazardous to your wealth: The common stock investment performance of individual investors. *J. Finance*, 55(2):773–806, 2000.

[6] S. A. Barghouthi and A. Ehsan. Market efficiency analysis of amman stock exchange through moving average method. *Int. J. Bus. Soc.*, 18:531, 2017.

[7] J.E. Beasley, N. Meade, and T.-J. Chang. An evolutionary heuristic for the index tracking problem. *Eur. J. Oper. Res.*, 148(3):621–643, 2003.

[8] K. Benidis, Y. Feng, and D. P. Palomar. Optimization methods for financial index tracking: From theory to practice. *Found. Trends Optim.*, 3(3):171–279, 2018.

[9] K. Benidis, Y. Feng, and D. P. Palomar. Sparse portfolios for high-dimensional financial index tracking. *IEEE Trans. Signal Process.*, 66(1):155–170, 2018.

[10] P. Berger, G. Hannak, and G. Matz. Graph signal recovery via primal-dual algorithms for total variation minimization. *IEEE J. Sel. Top. Signal Process.*, 11(6):842–855, 2017.

[11] D. Bianchi and A. Gargano. High-dimensional index tracking with cointegrated assets using an hybrid genetic algorithm. *Capital Markets: Asset Pricing & Valuation eJournal*, 03 2011.

[12] I. Bisio, C. Garibotto, H. Haleem, F. Lavagetto, and A. Sciarrone. A systematic review of drone based road traffic monitoring system. *IEEE Access*, 10:101537–101555, 2022.

[13] J. Boulanger, N. Pustelnik, L. Condat, L. Sengmanivong, and T. Piolot. Nonsmooth convex optimization for structured illumination microscopy image reconstruction. *Inverse Probl.*, 34(9):095004, 2018.

[14] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Found. Trends Mach. Learn*, 3:1–122, 2011.

[15] P. S. Bradley, K. P. Bennett, and A. Demiriz. Constrained k-means clustering. *Microsoft Research, Redmond*, 20(0):0, 2000.

[16] J. A. Castro-Correa, J. H. Giraldo, M. Badiey, and F. D. Malliaros. Gegenbauer graph neural networks for time-varying signal reconstruction. *IEEE Trans. Neural Netw. Learn. Syst.*, 2024.

[17] J. A. Castro-Correa, J. H. Giraldo, A. Mondal, M. Badiey, T. Bouwmans, and F. D. Malliaros. Time-varying signals recovery via graph neural networks. In *Proc. 2023 ICASSP IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, pages 1–5, 2023.

[18] Y. Che, S. Chen, and X. Liu. Sparse index tracking portfolio with sector neutrality. *Mathematics*, 10(15):2645, 2022.

[19] S. Chen and Y. C. Eldar. Graph signal denoising via unrolling networks. In *Proc. ICASSP 2021 - 2021 IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, pages 5290–5294, 2021.

[20] S. Chen, A. Sandryhaila, J. M. F. Moura, and J. Kovacevic. Signal denoising on graphs via graph filtering. In *2014 IEEE Glob. Conf. Signal Inf. Process. (GlobalSIP).*, pages 872–876, 2014.

[21] S. Chen, A. Sandryhaila, J.M.F. Moura, and J. Kovacevic. Signal recovery on graphs: Variation minimization. *IEEE Trans. Signal Process.*, 63(17):4609–4624, 2015.

[22] S. Chen and C. Shao. Efficient online tracking-by-detection with kalman filter. *IEEE Access*, PP:1–1, 2021.

[23] G. Chierchia, N. Pustelnik, J.-C. Pesquet, and B. Pesquet-Popescu. Epigraphical projection and proximal tools for solving constrained convex optimization problems. *Signal Image Video Process.*, 9(8):1737–1749, 2015.

[24] L. Condat. A primal–dual splitting method for convex optimization involving Lipschitzian, proximable and linear composite terms. *J. Optim. Theory Appl.*, 158, 08 2013.

[25] L. Condat. A generic proximal algorithm for convex optimization—application to total variation minimization. *IEEE Signal Process. Lett.*, 21(8):985–989, 2014.

[26] L. Condat. Fast projection onto the simplex and the ℓ_1 ball. *Math. Program.*, 158:575–585, 2016.

[27] L. Condat and A. Hirabayashi. Cadzow denoising upgraded: A new projection method for the recovery of dirac pulses from noisy linear measurements. *Sampling Theory Signal Image Process.*, 14(1):17–47, 2015.

[28] L. Condat, D. Kitahara, A. Contreras, and A. Hirabayashi. Proximal splitting algorithms for convex optimization: A tour of recent advances, with new twists. *SIAM Rev.*, 65(2):375–435, 2023.

[29] C. Couprie, L. Grady, L. Najman, J.-C. Pesquet, and H. Talbot. Dual constrained TV-based regularization on graphs. *SIAM J. Imag. Sci.*, 6(3):1246–1273, 2013.

[30] B. S. Dees, L. Stanković, A. G Constantinides, and Da. P Mandic. Portfolio cuts: A graph-theoretic framework to diversification. In *Proc. 2020 IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, pages 8454–8458. IEEE, 2020.

[31] C. Dinesh, G. Cheung, and I. V. Bajić. 3d point cloud color denoising using convex graph-signal smoothness priors. In *2019 IEEE 21st Int. Workshop Multimed. Signal Process. (MMSP)*, pages 1–6, 2019.

[32] X. Dong, D. Thanou, P. Frossard, and P. Vandergheynst. Learning laplacian matrix in smooth graph signal representations. *IEEE Trans. Signal Process.*, 64(23):6160–6173, 2016.

[33] C. Dose and Silvano C. Clustering of financial time series with application to index and enhanced index tracking portfolio. *Phys. A: Stat. Mech. Appl.*, 355(1):145–151, 2005. Market Dynamics and Quantitative Economics.

[34] R. Du, C. Chen, B. Yang, N. Lu, X. Guan, and X. Shen. Effective urban traffic monitoring by vehicular sensor networks. *IEEE Trans. Veh. Technol.*, 64:273–286, 2015.

[35] H. E. Egilmez, E. Pavez, and A. Ortega. Graph learning from data under Laplacian and structural constraints. *IEEE J. Sel. Top. Signal Process.*, 11(6):825–841, 2017.

[36] V. R. M. Elias, V. C. Gogineni, W. A. Martins, and S. Werner. Kernel regression on graphs in random fourier features space. In *Proc. 2021 ICASSP IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, pages 5235–5239, 2021.

[37] A. A. Gaivoronski, S. Krylov, and N. van der Wijst. Optimal portfolio selection and dynamic benchmark tracking. *Eur. J. Oper. Res.*, 163(1):115–131, 2005.

[38] J. H. Giraldo, A. Mahmood, B. Garcia-Garcia, D. Thanou, and T. Bouwmans. Reconstruction of time-varying graph signals via sobolev smoothness. *IEEE Trans. Signal Inf. Process. Netw.*, 8:201–214, 2022.

[39] F. Grassi, A. Loukas, N. Perraudin, and B. Ricaud. A time-vertex signal processing framework: Scalable processing and meaningful representations for time-series on graphs. *IEEE Trans. Signal Process.*, 66(3):817–829, 2018.

[40] L. Hagen and A. B. Kahng. New spectral methods for ratio cut partitioning and clustering. *IEEE transactions on computer-aided design of integrated circuits and systems*, 11(9):1074–1085, 1992.

[41] M. Hong, Z. Luo, and M. Razaviyayn. Convergence analysis of alternating direction method of multipliers for a family of nonconvex problems. In *Proc. 2015 IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, pages 3836–3840, 2015.

[42] E. Isufi, F. Gama, and A. Ribeiro. Edgenets: Edge varying graph neural networks. *IEEE Trans. Pattern Anal. Mach. Intell.*, 44(11):7457–7473, 2022.

[43] R. Jansen and R. Dijk. Optimal benchmark tracking with small portfolios. *J. Portfolio Manage.*, 28:33–39, 12 2002.

[44] Junzheng Jiang, David B. Tay, Qiyu Sun, and Shan Ouyang. Recovery of time-varying graph signals via distributed algorithms on regularized problems. *IEEE Trans. Signal Inf. Process. Netw.*, 6:540–555, 2020.

[45] J. Jońca, M. Pawnuk, Y. Bezyk, A. Arsen, and I. Sówka. Drone-assisted monitoring of atmospheric pollution—a comprehensive review. *Sustainability*, 14(18):11516, 2022.

[46] V. Kalyagin, A. Koldanov, P. Koldanov, and V. Zamaraev. Market graph and markowitz model. *Optimization in Science and Engineering: In Honor of the 60th Birthday of Panos M. Pardalos*, pages 293–306, 2014.

[47] Z. Kang, H. Pan, S. C. H. Hoi, and Z. Xu. Robust graph learning from noisy data. *IEEE Trans. Cybern.*, 50(5):1833–1843, 2020.

[48] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2017.

[49] H. Kojima, H. Noguchi, K. Yamada, and Y. Tanaka. Restoration of time-varying graph signals using deep algorithm unrolling. In *Proc. 2023 ICASSP IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, pages 1–5, 2023.

[50] N. Komodakis and J.-C. Pesquet. Playing with duality: An overview of recent primal-dual approaches for solving large-scale optimization problems. *IEEE Signal Process. Mag.*, 32(6):31–54, 2015.

[51] S. Kyochi, S. Ono, and I. W. Selesnick. Epigraphical relaxation for minimizing layered mixed norms. *IEEE Trans. Signal Process.*, 69:2923–2938, 2021.

[52] G. Li and T. Pong. Global convergence of splitting methods for nonconvex composite optimization. *SIAM J. Optim.*, 25:2434–2460, 09 2015.

[53] X. P. Li, Z. Shi, C. Leung, and H. C. So. Sparse index tracking with k-sparsity or ε-deviation constraint via l_0-norm minimization. *IEEE Trans. Neural Netw. Learn. Syst.*, pages 1–14, 2022.

[54] X. P. Li, Z.-L. Shi, Q. Liu, and H. C. So. Fast robust matrix completion via entry-wise ℓ_0-norm minimization. *IEEE Trans. Cybern.*, 53(11):7199–7212, 2023.

[55] X. P. Li, Y. Yan, E. E. Kuruoglu, H. C. So, and Y. Chen. Robust recovery for graph signal via ℓ_0-norm regularization. *IEEE Signal Process. Lett.*, 30:1322–1326, 2023.

[56] C. Y. Lin, L. Wu, Z. Wen, H. Tong, V. Griffiths-Fisher, L. Shi, and D. Lubensky. Social network analysis in enterprise. *Proc. IEEE*, 100:2759–2776, 2012.

[57] J. Liu, J. Lin, H. Qiu, J. Wang, and L. Nong. Time-varying signal recovery based on low rank and graph-time smoothness. *Digit. Signal Process.*, 133:103821, 2023.

[58] Z. Liu, X. P. Li, and H. C. So. ℓ_0-norm minimization-based robust matrix completion approach for MIMO radar target localization. *IEEE Trans. Aerosp. Electron. Syst.*, 59(5):6759–6770, 2023.

[59] L. Lombardo, S. Corbellini, M. Parvis, A. Elsayed, E. Angelini, and S. Grassini. Wireless sensor network for distributed environmental monitoring. *IEEE Trans. Instrum. Meas.*, 67:1214–1222, 2018.

[60] A. Loukas and D. Foucard. Frequency analysis of time-varying graph signals. In *2016 IEEE Glob. Conf. Signal Inf. Process. (GlobalSIP).*, pages 346–350, 2016.

[61] B. Malkiel. Passive investment strategies and efficient markets. *Eur. Finance Manage.*, 9:1–10, 03 2003.

[62] D. Maringer and O. Oyewumi. Index tracking with constrained portfolios. *Intell. Syst. Accounting, Finance Manage.*, 15:57–71, 06 2007.

[63] G. Mois, T. Sanislav, and S. Folea. A cyber-physical system for environmental monitoring. *IEEE Trans. Instrum. Meas.*, 65:1463–1471, 2016.

[64] A. Mondal, M. Das, A. Chatterjee, and P. Venkateswaran. Recovery of missing sensor data by reconstructing time-varying graph signals. In *Proc. 2022 30th Eur. Signal Process. Conf. (EUSIPCO).*, pages 2181–2185, 2022.

[65] J.-J. Moreau. Dual convex functions and proximal points in a Hilbert space. *CRAS, Paris*, 255:2897–2899, 1962.

[66] M. Nagahama, K. Yamada, Y. Tanaka, S. H. Chan, and Y. C. Eldar. Graph signal denoising using nested-structured deep algorithm unrolling. In *Proc. ICASSP 2021 - 2021 IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, pages 5280–5284, 2021.

[67] K. Naganuma and S. Ono. A general destriping framework for remote sensing images using flatness constraint. *IEEE Trans. Geosci. Remote Sens.*, 60:1–16, 2022.

[68] S. Namani and B. Gonen. Smart agriculture based on iot and cloud computing. In *2020 3rd Int. Conf. Inf. Comput. Technol. (ICICT)*, pages 553–556, 2020.

[69] S. K. Narang and A. Ortega. Compact support biorthogonal wavelet filterbanks for arbitrary undirected graphs. *IEEE Trans. Signal Process.*, 61(19):4673–4685, 2013.

[70] T. T. Nguyen, J. Idier, C. Soussen, and E. Djermoune. Non-negative orthogonal greedy algorithms. *IEEE Trans. Signal Process.*, 67(21):5643–5658, 2019.

[71] K. J. Oh, T. Y. Kim, and S. Min. Using genetic algorithm to support portfolio optimization for index fund management. *Expert Syst. Appl.*, 28(2):371–379, 2005.

[72] Luís ML Oliveira and Joel JPC Rodrigues. Wireless sensor networks: A survey on environmental monitoring. *J. Commun.*, 6(2):143–151, 2011.

[73] S. Ono. L_0 gradient projection. *IEEE Trans. Image Process.*, 26(4):1554–1564, April 2017.

[74] S. Ono. Primal-dual plug-and-play image restoration. *IEEE Signal Process. Lett.*, 24(8):1108–1112, 2017.

[75] S. Ono and I. Yamada. Hierarchical convex optimization with primal-dual splitting. *IEEE Trans. Signal Process.*, 63(2):373–388, 2014.

[76] S. Ono and I. Yamada. Signal recovery with certain involved convex data-fidelity constraints. *IEEE Trans. Signal Process.*, 63(22):6149–6163, 2015.

[77] S. Ono, I. Yamada, and I. Kumazawa. Total generalized variation for graph signals. In *Proc. 2015 IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, pages 5456–5460, 2015.

[78] 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.*, 2(2):137–148, 2016.

[79] A. Pareja, G. Domeniconi, J. Chen, T. Ma, T. Suzumura, H. Kanezashi, T. Kaler, S. Baruah, and C. E. Leiserson. EvolveGCN: Evolving graph convolutional networks for dynamic graphs. In *Proc. AAAI Conf. Artif. Intell.*, volume 34, pages 5363–5370, 2020.

[80] J. Pearson and E. Stear. Kalman filter applications in airborne radar tracking. *IEEE Trans. Aerosp. Electron. Syst.*, AES-10:319–329, 1974.

[81] N. Perraudin, A. Loukas, F. Grassi, and P. Vandergheynst. Towards stationary time-vertex signal processing. In *Proc. 2017 IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, pages 3914–3918, 2017.

[82] N. Perraudin, J. Paratte, D. Shuman, L. Martin, V. Kalofolias, P. Vandergheynst, and D. K. Hammond. GSPBOX: A toolbox for signal processing on graphs. *ArXiv e-prints*, 2014.

[83] J. Picone. Signal modeling techniques in speech recognition. *Proc. IEEE*, 81:1215–1247, 1993.

[84] X. Pu, S. Lun Chau, X. Dong, and D. Sejdinovic. Kernel-based graph learning from smooth signals: A functional viewpoint. *IEEE Trans. Signal Inf. Process. Netw.*, PP:1–1, 02 2021.

[85] K. Qiu, X. Mao, X. Shen, X. Wang, T. Li, and Y. Gu. Time-varying graph signal reconstruction. *IEEE J. Sel. Top. Signal Process.*, 11(6):870–883, 2017.

[86] D. Romero, N. I. Vassilis, and G. B. Giannakis. Kernel-based reconstruction of space-time functions on dynamic graphs. *IEEE J. Sel. Top. Signal Process.*, 11(6):856–869, 2017.

[87] A. Sandryhaila and J. M. F. Moura. Discrete signal processing on graphs. *IEEE Trans. Signal Process.*, 61(7):1644–1656, 2013.

[88] A. Sandryhaila and J.M.F. Moura. Big data analysis with signal processing on graphs: Representation and processing of massive data sets with irregular structure. *IEEE Signal Process. Mag.*, 31(5):80–90, 2014.

[89] S. Sardellitti, S. Barbarossa, and P. Di Lorenzo. Online learning of time-varying signals and graphs. In *Proc. ICASSP 2021 - 2021 IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, pages 5230–5234, 2021.

[90] A. Seelenfreund, George C. C. Parker, J. C. Pennypacker, and E. James. Monthly moving averages—an effective investment tool? *J. Financ. Quant. Anal.*, 3:315–326, 1968.

[91] Z.-L. Shi, X. P. Li, C.-S. Leung, and H. C. So. Cardinality constrained portfolio optimization via alternating direction method of multipliers. *IEEE Trans. Neural Netw. Learn. Syst.*, 35(2):2901–2909, 2024.

[92] 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.*, 30(3):83–98, 2013.

[93] S. Takemoto, K. Naganuma, and S. Ono. Graph spatio-spectral total variation model for hyperspectral image denoising. *IEEE Geosci. Remote Sens. Lett.*, 19:1–5, 2022.

[94] P. M. Pardalos V. Boginski, S. Butenko. *On Structural Propertiesof the Market Graph.* in Innovations in Financial and Economic Networks, A. Nagurney, Ed. Edward Elgar Publishers, 2003.

[95] P. M. Pardalos V. Boginski, S. Butenko. Statistical analysis of financial networks. *Computational Statistics & Data Analysis*, 48(2):431–443, 2005.

[96] Ulrike V. L. A tutorial on spectral clustering. *Statistics and computing*, 17:395–416, 2007.

[97] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio. Graph attention networks. *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2018.

[98] A. Venkitaraman, S. Chatterjee, and P. Handel. Predicting graph signals using kernel regression where the input signal is agnostic to a graph. *IEEE Trans. Signal Inf. Process. Netw.*, PP:1–1, 08 2019.

[99] A. Venkitaraman, P. Frossard, and S. Chatterjee. Kernel regression for graph signal prediction in presence of sparse noise. In *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5426–5430, 2019.

[100] B. C. Vu. A splitting algorithm for dual monotone inclusions involving cocoercive operators. *Adv. Comput. Math.*, 38(3):667–681, 2013.

[101] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip. A comprehensive survey on graph neural networks. *IEEE Trans. Neural Netw. Learn. Syst.*, 32(1):4–24, 2021.

[102] M. Yaghoobi, D. Wu, and M. E. Davies. Fast non-negative orthogonal matching pursuit. *IEEE Signal Process. Lett.*, 22(9):1229–1233, 2015.

[103] Koki Yamada, Yuichi Tanaka, and Antonio Ortega. Time-varying graph learning with constraints on graph temporal variation. *arXiv preprint arXiv:2001.03346*, 2020.

[104] E. Yamagata and S. Ono. Sparse index tracking: Simultaneous asset selection and capital allocation via ℓ_0-constrained portfolio. *arXiv preprint arXiv:2309.10152*, 2023.

[105] E. Yamagata and S. Ono. Robust time-varying graph signal recovery for dynamic physical sensor network data, arXiv:2202.06432, 2023.

[106] B. Yu, H. Yin, and Z. Zhu. Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. *Proc. Int. Jt. Conf. Artif. Intell. (IJCAI)*, pages 3634–3640, 2018.

[107] C. Zhang and H. Wai. Learning multiplex graph with inter-layer coupling. In *ICASSP 2024 IEEE Int. Conf. Acoust, Speech Signal Process (ICASSP)*, pages 12916–12920, 2024.

[108] S. Zhang, Q. Deng, and Z. Ding. Signal processing over multilayer graphs: Theoretical foundations and practical applications. *IEEE Internet Things J.*, 11(2):2453–2471, 2024.

[109] Y. Zheng, T. M. Hospedales, and Y. Yang. Diversity and sparsity: A new perspective on index tracking. In *Proc. 2020 IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, pages 1768–1772, 2020.

[110] J. Zhou, D. Gao, and D. Zhang. Moving vehicle detection for automatic traffic monitoring. *IEEE Trans. Veh. Technol.*, 56:51–59, 2007.

Publications Related to This Dissertation

Articles in Journal Papers

[J1] E. Yamagata, K. Naganuma and S. Ono, "Robust Time-Varying Graph Signal Recovery for Dynamic Physical Sensor Network Data," *IEEE Transactions on Signal and Information Processing over Networks*, 2024.

[J2] E. Yamagata and S. Ono, "Sparse Index Tracking: Simultaneous Asset Selection and Capital Allocation via ℓ_0-Constrained Portfolio," *IEEE Open Journal of Signal Processing*, vol. 5, pp. 810-819, 2024.

International Conference

[C1] E. Yamagata and S. Ono, "Recovery of Time Series of Graph Signals Over Dynamic Topology," in *Proceedings of Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, Tokyo, Japan, pp. 330-336, 2021, (invited paper).

[C2] E. Yamagata and S. Ono, "Risk-Managed Sparse Index Tracking Via Market Graph Clustering," in *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Seoul, Korea, Republic of, pp. 9796-9800, 2024.

Other Publications

Articles in Journal Papers

[J3] H. Kataoka, K. Okayasu, A. Matsumoto, E. Yamagata, R. Yamada, N. Inoue, A. Nakamura, and Y. Satoh,, "Pre-training without Natural Images," *International Journal of Computer Vision*, vol. 130, pp. 990-1007, 2022.

International Conference

[C3] R. Nakamura, R. Tadokoro, E. Yamagata, Y. Kondo, K. Hara, H. Kataoka, and N. Inoue, "Pseudo-Outlier Synthesis Using Q-Gaussian Distributions for Out-of-Distribution Detection," in *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Seoul, Korea, Republic of, pp. 3120–3124, 2024.

[C4] H. Kataoka, K. Hara, R. Hayashi, E. Yamagata, and N. Inoue, "Spatiotemporal Initialization for 3D CNNs With Generated Motion Patterns," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pp. 1279–1288, 2022.

[C5] H. Kataoka, A. Matsumoto, R. Yamada, Y. Satoh, E. Yamagata, and N. Inoue, "Formula-Driven Supervised Learning With Recursive Tiling Patterns," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*, pp. 4098–4105, 2021.

[C6] N. Inoue†, E. Yamagata†, and H. Kataoka, "Initialization Using Perlin Noise for Training Networks with a Limited Amount of Data," in *Proceedings of the 25th International Conference on Pattern Recognition (ICPR)*, Milan, Italy, pp. 1023–1028, 2021, (†: equal contribution).

[C7] H. Kataoka, K. Okayasu, A. Matsumoto, E. Yamagata, R. Yamada, N. Inoue, A. Nakamura, and Y. Satoh, "Pre-training without Natural Images," in *Proceedings of the Asian Conference on Computer Vision (ACCV)*, 2020.