Signal Processing over Dynamic Graphs

Das, B.

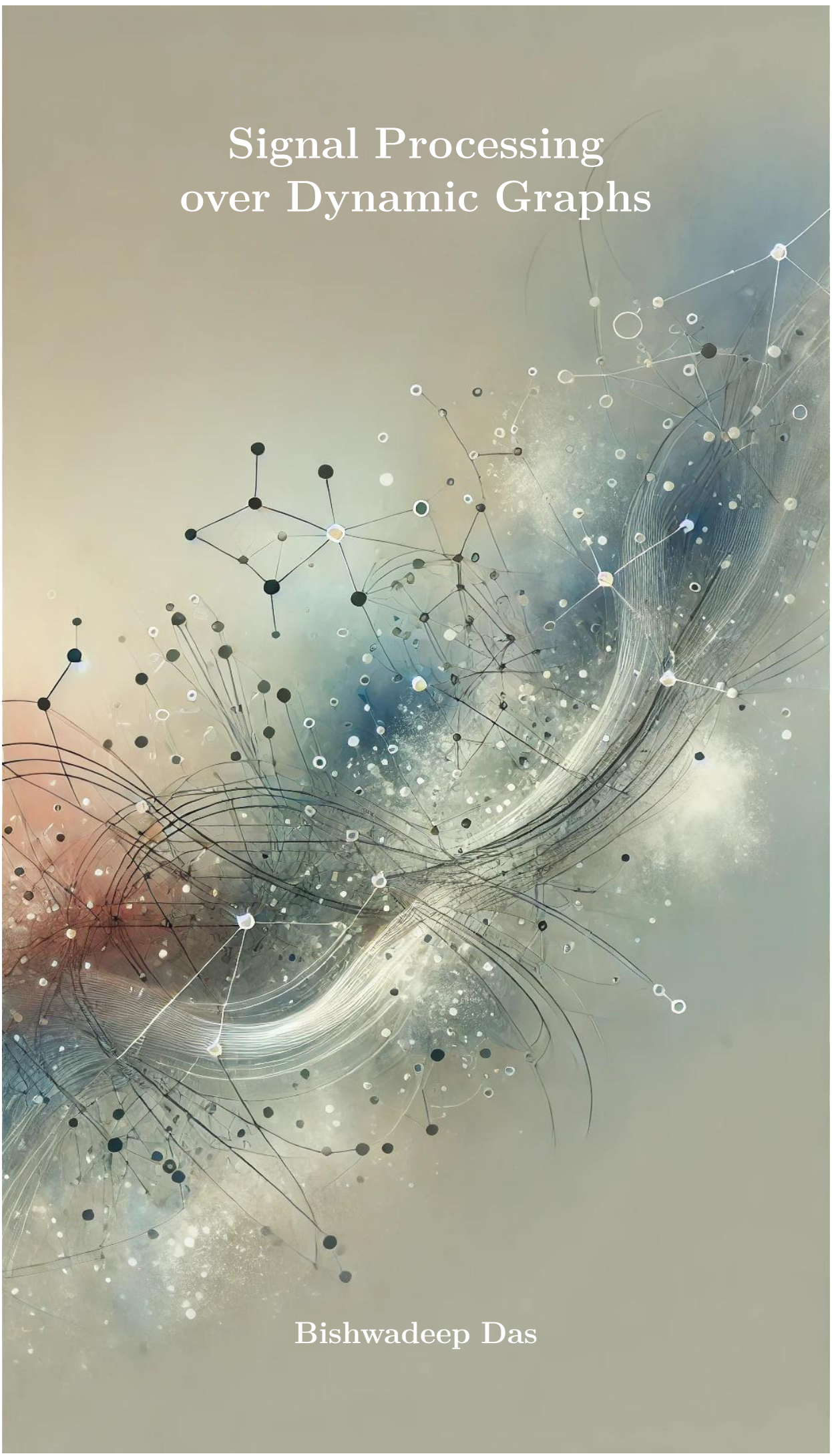
DOI
10.4233/uuid:9285b1c3-5384-4336-b90c-0bef8fec3392

Publication date
2025

Document Version
Final published version

Citation (APA)
Das, B. (2025). *Signal Processing over Dynamic Graphs*. [Dissertation (TU Delft), Delft University of Technology]. https://doi.org/10.4233/uuid:9285b1c3-5384-4336-b90c-0bef8fec3392

Signal Processing over Dynamic Graphs



Bishwadeep Das

SIGNAL PROCESSING OVER DYNAMIC GRAPHS

SIGNAL PROCESSING OVER DYNAMIC GRAPHS

Dissertation

for the purpose of obtaining the degree of doctor
at Delft University of Technology
by the authority of the Rector Magnificus, prof. dr. ir. T.H.J.J. van der Hagen,
chair of the Board for Doctorates
to be defended publicly on
Monday 10 March 2025 at 17:30 o'clock

by

Bishwadeep DAS

This dissertation has been approved by the promotors.

Composition of the doctoral committee:

Rector Magnificus,	chairperson
Prof. dr. A. Hanjalic,	Delft University of Technology, *promotor*
Dr. E. Isufi,	Delft University of Technology, *copromotor*

Independent members:

Prof. dr. A. Garcia Marques	King Juan Carlos University, Spain
Prof. dr. ir. G.J.T. Leus	Delft University of Technology
Prof. dr. G. Mateos,	Rochester University of Technology, United States of America
Prof. dr. C. Richard,	University Côte d'Azur, France
Prof. dr. ir. M.H.G Verhaegen,	Delft University of Technology

TU Delft Delft University of Technology

NWO Nederlandse Organisatie voor Wetenschappelijk Onderzoek

Cover by:

An electronic copy of this dissertation is available at
https://repository.tudelft.nl/.

CONTENTS

SUMMARY

Extending the concepts of classical signal processing to graphs, a wide array of methods have come to the fore, including filtering, reconstruction, classification, and sampling. Existing approaches in graph signal processing consider a known and static topology, i.e., fixed number of nodes and a fixed edge support. Two types of tasks stand out, namely, topology inference, where the edge support along with their weights are estimated from signals; and data processing, where existing data and the known topology are used to perform different tasks. However, such tasks become quite challenging when the network size and support changes over time. Particularly, these challenges involve adapting to the changing topology, data distributions and dealing with unknown topological information. The latter manifests for example, when new nodes are available to attach to the graph but their connectivity is uncertain as is the case in cold start graph-based recommender systems.

The key contribution of this dissertation is proposing methodologies for signal processing over dynamic networks which are aimed at the two aforementioned tasks. For dynamic networks with incoming nodes, we process signals by introducing a parametric stochastic attachment model. In this model, the incoming nodes connect with probability to existing nodes with certain weights. This uncertainty allows us to model input output relations and allows us to cast them in the context of different graph signal processing tasks. We learn the model attachment parameters in a task-aware setting, allowing us to interpret topology identification in task-aware settings. Separately, we also propose filter design strategies for processing signals both at the incoming and existing nodes using stochastic attachment models.

Another contribution of this dissertation is to extend graph signal processing with graph filters to the scenario where the graph keeps growing in size with streaming data. We propose online graph filter design which updates the filter online, based on incoming nodes. We design this both for scenarios where the incoming node connectivity is known and unknown. In the unknown connectivity case, we study the performance difference between knowing and not knowing the topology and how the stochastic attachment influences it. We also show that by adapting the stochastic attachment, we can learn faster from the data stream.

Finally, we consider the task of topology decomposition and identification for dynamic networks with fixed nodes but changing edge support. We build a tensor of partially observed adjacency matrices corresponding to such a dynamic topology and express this in terms of underlying latent graphs and their temporal signatures. Furthermore, we account for the time-varying graph signals as a prior to aid identifying these latent graphs and missing components of the topology. These latent graphs are individually and collectively expressive and provide interpretable decompositions along with outperforming traditional structure agnostic low-rank decompositions.

SAMENVATTING

Door de concepten van klassieke signaalverwerking uit te breiden naar grafen, is een breed scala aan methoden naar voren gekomen, waaronder filteren, reconstructie, classificatie en bemonstering. Bestaande benaderingen in graafsignaalverwerking gaan uit van een bekende en statische topologie, d.w.z. een vast aantal knopen en een vaste ondersteuning. Twee soorten taken springen eruit, namelijk topologie-inferentie, waarbij de ondersteuning samen met hun gewichten worden geschat op basis van grafsignalen; en grafsignaalverwerking, waarbij en de bekende topologie en de signal erop worden gebruikt om verschillende taken uit te voeren. Dergelijke taken worden echter een hele uitdaging als de netwerkgrootte en ondersteuning in de loop van tijd verandert. Deze uitdagingen hebben vooral te maken met het aanpassen aan de veranderende topologie, gegevensdistributies en het omgaan met onbekende topologische informatie. Dit laatste is bijvoorbeeld het geval wanneer er nieuwe knopen beschikbaar zijn aan de graf te koppelen, maar hun connectiviteit onzeker is, zoals het geval is in cold start graafgebaseerde aanbevelingssystemen.

De belangrijkste bijdrage van dit proefschrift is het voorstellen van methodologieën voor signaalverwerking over dynamische netwerken die gericht zijn op de twee bovengenoemde taken. Voor dynamische netwerken met inkomende knopen verwerken we signalen door een parametrisch stochastisch verbindingsmodel te introduceren. In dit model verbinden de binnenkomende knopen zich met waarschijnlijkheid met bestaande knopen met bepaalde gewichten. Deze onzekerheid stelt ons in staat om de relaties tussen output en input te modelleren en om deze te gieten in de context van verschillende graafsignaalverwerkingstaken. We leren de parameters van het model in een taakbewuste setting, waardoor we topologie-identificatie kunnen interpreteren in taakbewuste settings. Los daarvan stellen we ook filterontwerpstrategieën voor om signalen te verwerken op zowel de inkomende als de bestaande knopen met behulp van stochastische koppelingsmodellen.

Een andere bijdrage van dit proefschrift is het uitbreiden van graafsignaalverwerking met grafiekfilters naar het scenario waarbij de graaf steeds groter wordt met stromende data. We stellen een online graaffilterontwerp voor dat het filter online bijwerkt op basis van binnenkomende knopen. We ontwerpen dit zowel voor scenario's waarin de connectiviteit van inkomende knopen bekend is als voor scenario's waarin deze onbekend is. In het geval van onbekende connectiviteit bestuderen we het prestatieverschil tussen het wel en niet kennen van de topologie en hoe de stochastische koppeling dit beïnvloedt. We laten ook zien dat we door de stochastische koppeling aan te passen sneller kunnen leren van de datastroom.

Ten slotte bekijken we de taak van topologiedecompositie en -identificatie voor dynamische netwerken met vaste knopen maar veranderende ondersteuning. We bouwen een tensor van gedeeltelijk waargenomen adjacency matrices die correspondeert met

een dergelijke dynamische topologie en drukken deze uit in termen van onderliggende latente grafen en hun temporele signaturen. Bovendien houden we rekening met de tijdsvariërende graafsignalen als een prior om deze latente grafen en ontbrekende componenten van de topologie te helpen identificeren. Deze latente grafen zijn individueel en collectief expressief en leveren interpreteerbare decomposities die beter presteren dan traditionele structuur agnostische low-rank decomposities.

1

INTRODUCTION

1.1. PROCESSING DATA OVER GRAPHS

Networks and network-structured data are universally observed. Common examples of networks include biological [1], social [2], transportation [3], and citation networks [4] and the respective data they generate such as gene expressions signals, social opinions, traffic, and emerging research topics. To represent the structure of these networks, we rely on graphs which are composed of a set of nodes and edges that form pair-wise connections between some of the nodes. In social networks, users typically represent the nodes and edges denote similarities between users in the form of virtual friendship. In biological networks, protein molecules are the nodes and edges exist between molecules which share similar functions. [5]. Network data is intrinsically linked to its structure, which involves the nodes. An example is the quantified opinion of users in a social network, which can represent anything from taste in music to opinions on social issues. Another example is the functionality of a protein molecule in a protein network, and the rate of traffic at a crossing in a road transportation network. Graph data are intrinsically coupled with the underlying topology, which is an irregular domain as opposed to structured domains concerning audio or image data. To extract meaningful information, or solve graph-related tasks with networks and network data, it is needed to develop tools for processing data over graphs.

Such tools have been developed and used extensively to process data over graphs by utilising the structure, over a variety of applications [6–8]. To represent the network data, we can define a function mapping from the node set to the set of real number, which is also known as a graph signal. To illustrate, consider the task of predicting the rating a user would give to a specific item in a movie recommender system such as Netflix or IMDB. Graph signal processing-based approaches have achieved great success in collaborative filtering either in linear form or via nonlinear methods via graph neural networks [9]. There exists a user-user graph, which considers users as nodes and connects similar users via edges. This user-user graph is built on the similarities estimated from the existing ratings or from a social network among users [10]. The graph signal here is the rating a user provides to a particular movie item. The signal processing task is using the existing data and the

topology of the graph to infer the rating of users that have not yet consumed a particular item. This and the similar graph signal processing problem have received a lot of attention and breakthroughs when applied on static graphs.

However, graphs are rarely static but dynamic as they change over time. An important observed dynamic is one comprising changing edges over time [11]. In the context of recommender systems, the preferences of users change, leading to changes in pair-wise similarities between users, and, in turn, leading to a graph with changing connections [12]. The task now becomes to predict ratings each user would provide to items over time by accounting also for the changes in the topology.

The second significant dynamic involves changing nodes, particularly the growth of the graph through an addition of nodes. This is studied extensively for citation, grid, and social networks [13–15] but they all concern topological modeling aspects from a network science perspective and overlook the processing tasks associated to these networked-data.. In the earlier recommender system example, this happens when a new user enters the system [16–18]. In contrast to static graphs, processing data over expanding graphs has to take into account the evolving topology induced by the addition or deletion of nodes and edges. Thus, we want to predict how a new user would rate a movie without having access to any of the user's previous ratings or social connections [16–18].

1.2. CHALLENGES OF DYNAMIC GRAPH SIGNAL PROCESSING

Despite a multitude of works collectively addressing problems in static and dynamic graph signal processing, there are still two main challenges that remain unsolved.

The first challenge concerns the certainty surrounding the dynamic graph. In some cases, it is known how the graph will grow. This happens in a recommender system when a new user has provided access to ratings and relevant side-information which can be used to estimate similarities with the existing users. However, in many other instances, there is uncertainty in this regard [19]. This happens when a new user in the recommender system has no associated information, or when the similarities between users are estimated poorly due to delays in getting the desired number of ratings. It is therefore difficult to estimate how this user would connect to the existing network. This poses a greater challenge where new users are always arriving and items need to be recommended to these users without waiting for their ratings to be available.

The second challenge concerns the statistical nature of the data over the incoming nodes. The data is not available all-at-once, and it may not follow a fixed distribution, making it difficult to rely on conventional batch-based approaches. In line with the recommender system example, the new users who keep joining the network need not have the same taste in movies as the existing users or even the users who joined before. Moreover, the data over the existing networks may also change [20].

The third challenge concerns dynamic graphs with fixed nodes and varying edge support. We may observe a dynamic graph but its evolution is often governed by a few underlying factors that are difficult to estimate. Typically, these factors are estimated from spatiotemporal graph signals [21–24]. However, given a dynamic

network, its analysis is limited to representing them as tensors followed by low-rank tensor decompositions, without accounting for the dynamic graph signals. The focus is mostly on downstream tasks. Low-rank decompositions are limited in their ability to represent topologies int eh form of adjacency or Laplacian matrices, let alone obtain the driving underlying factors.

For graphs with fixed nodes and changing edges, there exist some works discussing processing signals over them [25, 26]. For growing graphs, stochastic attachment models have been studied [13, 14, 27] as a way to model them. These models are often data and/or task-agnostic and depend on the structural characteristics of the existing graph. For example, one of these models [14] states that the new user is likely to connect to, i.e., have similarities in taste with the existing user who has rated more items. This may not always be true for predicting ratings for the new user.

Second, the existing works which can handle incoming nodes assume known attachment information, i.e., full knowledge of how they attach to the existing graph. [28–30]. They also infer the attachment in the presence of features [31] or generate embeddings from features. This allows these approaches to handle incoming nodes.

Dynamic graphs with fixed nodes and changing support have been studied [32–40]. The focus, however, has been on downstream tasks. For this purpose, the dynamic topology is represented as a tensor and a low-rank tensor decomposition is utilized for these tasks. The decompositions typically do not incorporate graphs or graph signals and are thus not always interpretable. While the low-rank representations make sense for tasks such as community detection, where similar nodes are expected to have closely-spaced embeddings, they do not throw light on the nature of the structural evolution over time. Such approaches also consider the signals or the tensor representation to be fully available, i.e., no missing observations.

1.3. THESIS CONTRIBUTION

Given the three challenges described in the previous section, the following are the aims of this dissertation:

> *Develop a theoretical framework for processing signals over dynamic graphs by accounting for changes both in the topology and in the signal associated to it.*

The specific aims of this dissertation are

1. **Changing nodes** (***Aim-1***)**.** We introduce a general framework that can process data over expanding graphs. Establishing the framework involves the following:
    i. Inferring the connectivity for incoming nodes that is both task-aware and data-driven.
    ii. Study signal processing techniques for inference over the existing nodes as well as single-shot incoming nodes with replacement.
    iii. Study online processing techniques for inference over a stream of incoming nodes.

2. **Changing connections** (***Aim-2***)**.** We develop a graph-aware representation of the network dynamics from a partially observed topology and signals. More precisely, we aim to uncover latent graphs driving the structural evolution using both topological and signal information as an alternative to existing low-rank decompositions.

Targeting these aims, this dissertation answers the following three challenging research questions.

RQ1: How do we process signals over expanding graphs when the connectivity is both known and unknown? (***Aim-1***)

To process signals over expanding graphs, the first step is to choose a tool fit for the purpose. In this dissertation, we opt for graph filters which are parameterized, distributed, and interpretable operators that shift the signals over the topology and combine them. Another reason for using graph filters is due to their widespread adoption for a variety of graph signal processing applications. However, using graph filters in the expanding setting is not straightforward. When the connectivity of incoming nodes is unknown, this poses a natural challenge to existing graph signal processing approaches. For instance, not knowing the topology fully due to absence of incoming node information can severely affect the output of graph filter-based operations. We address this challenge via stochastic attachment models, which are random in nature but are parameterized, thus enabling us to learn them. Naturally, how such approaches deviate from the case where the topology is fully known becomes important to investigate, theoretically and experimentally. Introduction of stochastic attachment models will affect the graph filter operation, notably its output. This will involve analyzing the filter operation, namely its output in expectation w.r.t. the parameters of the stochastic model. Since the stochastic attachment model is parameterized, a natural extension is to learn them from data in the context of signal processing over the expanding graph. From an experimental perspective, it is important to answer how does this task-aware, filter driven attachment compare to heuristic or data-driven ones. This involves comparing the performance for the class of proposed filters to heuristic and data-driven approaches which are either task, or topology-aware.

RQ2: How do we design dynamic algorithms for signal processing on continually expanding graphs? (***Aim-1***)

The growth of graphs is a gradual process. Existing approaches suited to fixed size graphs can process batches of data and are thus computationally expensive. Repeating this approach every time the graph grows is not desirable. Moreover, the data often arrives in a stream, thus batch solutions cannot be deployed. Thus, we would like quick efficient, adaptive model that updates with the streaming data. We do this by proposing an online filter design framework and quantify the worst-case performance gap w.r.t. the batch-based solution. More specifically, we conduct a static regret analysis and investigate its upper bound as a proxy for the worst case performance difference. Another relevant aspect worth investigating is quantifying

the effect of the stochastic attachment on the online filter performance. We address this by adapting the stochastic attachment and studying the corresponding regret. Finally, we compare the online graph filter with batch-based solutions and other deterministic graph signal processing tools in real-case studies. The comparison with bath-based solutions informs us how fast the filter adapts to the data and topology updates whereas the comparison with other graph-based tools can highlight the effect of graph filters in general.

RQ3: Given a partially observed evolving topology and an evolving process over it, are there a collection of latent graphs that drive both the evolution of this topology and of the process? (***Aim-2***)

We often consider for granted that the network we observe governs the dynamics in its topology and respective process. However, as is often the case with hidden Markov models or other latent representations of dynamics or observed phenomena, we question if there are meaningful graphs governing the topological evolution. Existing approaches for analyzing dynamic networks rely on forming a tensor by stacking the adjacency matrices and performing a low-rank decomposition. While this may be suited for downstream tasks like community detection, the low rank assumption renders it not so useful for capturing the actual structure, i.e., the graph topology, as low rank matrices need not capture all possible types of topologies. Such decompositions also do not exploit the nature of dynamic graph signals, which have shown to be closely linked to the topology or multiple static graphs. The posed research question targets representing dynamic networks in a new way. The primary challenge here is to decompose the structural evolution in an interpretable way, taking into account missing observations and dynamic graph signals. We do this by assuming that the evolving topology comprises a set of underlying, hidden latent graphs, each contributing to the observed topology. Different from low-rank decompositions, the focus here is on uncovering the topology, along with their relative importance when it comes to explaining the partial observations. Associated challenges involve integrating the dynamic graph signals to recover the underlying graphs using tools from topology identification and a methodology to solve for these components.

1.4. DISSERTATION OUTLINE

The dissertation contains seven chapters structured as follows:

Chapter 2 : Background. This chapter provides the necessary technical background required to follow the subsequent chapters. We introduce graph signals, the graph spectrum, and graph filters. We discuss dynamic graphs from the lens of this dissertation.

Chapter 3 : Learning task-aware expanding graphs. This chapter focuses on modelling the attachment of incoming nodes when their connectivity is unknown for different graph signal processing tasks such as interpolation using graph filters and topology identification using graph signal smoothness. We use a parametric stochastic attachment model to account for the unknown connections, in a way that

is possible to learn them. We study the effects of such an attachment model on graph filters, specifically the filter order, their ability to process data over undirected graphs, and the effect of this perturbation over the filter outputs. Since the attachment model is stochastic, we solve the tasks in expectation w.r.t the model parameters. The learnt model parameters correspond to a new way of interpreting how incoming nodes interact with the existing ones.

Chapter 4 : Graph filter for incoming nodes. This chapter focuses on designing graph filters for signal processing tasks on both the existing as well as incoming nodes under unknown connectivity. Differently from Chapter 3 where we focus only on the incoming node and connectivity, we focus on the incoming and existing nodes and designing the filter, given a task and no connectivity information for the incoming node. To do this, we train a two-filter bank over a pair of directed graphs, which capture the influence of the existing nodes on the incoming node and vice-versa.

Chapter 5 : Online filtering over expanding graphs. This chapter focuses on graph filter design over a graph which keeps growing over time. We take inspiration from online learning to update the graph filter every time a new node arrives, the task being inferring signals at these nodes. Motivated by the stochastic approaches in Chapters 3 and 4, we also perform online filter updates in the stochastic case where the connectivity of incoming nodes is unknown. To investigate the behaviour of this approach, we derive bounds on the static regret of the online approach for both the deterministic and stochastic online learners. We also introduce an adaptive stochastic online learner in an attempt to reduce the dependency on a fixed stochastic learner and study its regret.

Chapter 6 : Dynamic Graph Decomposition. This chapter focuses on representing the topology of a dynamic network with changing edges from partially observed edges and accompanying spatiotemporal graph signals. To make our representation topology aware, we assume the structural evolution to be composed of latent graphs, thus accounting for the structure. This draws a stark contrast with the more popular low-rank tensor-based decomposition, as matrix representations of graphs are typically not low rank. To account for graph signals, we impose a topology-data prior in the form of graph signal smoothness, the hope being it can aid in recovering the latent graphs, particularly when we do not observe many edges. We analyze our approach in terms of how expressive the recovered latent graphs are and contrast with the more typical low-rank decomposition w.r.t. their ability to predict the presence or absence of missing edges.

Chapter 7 : Conclusion. This chapter presents the conclusions obtained from this dissertation, notably from Chapters 3 to 6 and addresses future directions that extend naturally from this dissertation.

1.5. HOW TO READ THIS DISSERTATION

The main scientific contribution of this dissertation is presented in the technical Chapters 3, 4, 5, and 6. The background material can be found in Chapter 2. Each

of these chapters corresponds to a publication, which is referenced at the beginning of the chapter. In this book, we retain the original form of the publications, with only minor modifications where necessary. Chapters 3, 4, and 5 can be read sequentially, as they contain overlapping concepts. Chapter 6, however, can be read independently.

1.6. LIST OF PUBLICATIONS

Journal Papers

1. **B.Das**, A.Hanjalic, E. Isufi, "Task-aware connectivity learning for incoming nodes over growing graphs", *IEEE Transactions on Signal and Information Processing over Networks*, 8, pages 894-906, 2022.

2. **B.Das**, E.Isufi, "Online Graph Filtering Over Expanding Graphs", *IEEE Transactions on Signal Processing*, Feb. 2024

3. **B.Das**, A.B.Vlas, A.G.Marques, E.Isufi, "Dynamic Graph Topology Decomposition", *to be submitted to IEEE Transactions on Signal Processing*

Conference Papers

1. **B.Das**, E. Isufi, "Tensor graph decomposition for temporal networks", *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Seoul, South Korea, April 2024.

2. **B.Das**, E.Isufi, "Online filtering over expanding graphs", *Asilomar Conference on Signals, Systems, and Computers*, Virtual, Oct. 2022 **(finalist best paper award)**.

3. **B.Das**, E. Isufi, "Graph filtering over expanding graphs", *2022 IEEE Data Science and Learning Workshop (DSLW)*, May 2022.(**best paper award**)

4. **B.Das**, E. Isufi, "Learning expanding graphs for signal interpolation", *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2022.

Other contributions

1. S.Rey, **B.Das**, E.Isufi, "Online Learning Of Expanding Graphs", *IEEE Open Journal of Signal Processing*

2. **B.Das**, M.Navarro, S.Segarra, E.Isufi, "Bayesian Filtering on Graphs", *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Hyderabad, India, Apr. 2024

3. M.Yang, **B.Das**, E.Isufi, "Online Edge Flow Prediction Over Expanding Simplicial Complexes", *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Rhodes, Greece, Jun. 2023

4. **B.Das**, E. Isufi, "Online Vector Autoregressive Models Over Expanding Graphs", *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Jun. 2023.

2

Background

2.1. ELEMENTS OF GRAPH SIGNAL PROCESSING

We start this chapter by defining the basic concepts and terminologies surrounding graphs, their representations, and graph signals.[41]

2.1.1. GRAPHS

A graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ comprises a vertex set $\mathcal{V} = \{v_1, \ldots, v_N\}$ of N nodes and an edge set $\mathcal{E} = \{(v_i, v_j)\}$ for all pairs (v_i, v_j) that share an edge between them. The nodes can be humans [42], sensors [43], or research articles [44], whereas the corresponding interactions can be communication between a pair of humans, sensors exchanging information with each other, or one research article citing another. The number of edges is low compared to the total number of possible edges and this makes graphs sparse[1]. The neighbourhood $\mathcal{N}_i$ of node v_i comprises all nodes v_j with $j \neq i$ such that $(v_i, v_j) \in \mathcal{E}$ for undirected graphs[2]. We consider two types of edges:

Undirected edges. If $(v_i, v_j) \in \mathcal{E}$, this implies $(v_j, v_i) \in \mathcal{E}$. Graphs with only undirected edges are called undirected graphs. An example is a social network where users are nodes and edges exist between users that are friends.

Directed edges. These edges start from one node and are directed at another. For such edges, if $(v_i, v_j) \in \mathcal{E}$, this does not imply $(v_j, v_i) \in \mathcal{E}$. Graphs with directed edges are called directed graphs. A example is a citation network where articles are nodes and edges exist when a new article cites an already existing one. The nature of the citation process makes the edge directed.

2.1.2. GRAPH SHIFT OPERATOR

A graph shift operator (GSO) of a graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ is a matrix $\mathbf{S} \in \mathbb{R}^{N \times N}$ with $S_{i,j} \neq 0$ iff $(v_i, v_j) \in \mathcal{E}$. For undirected graphs, we have $S_{i,j} = S_{j,i}$. The GSO embeds the graph topology in the form of a matrix. Examples of the GSO include the adjacency and the Laplacian matrices, as we detail next.

Adjacency matrix. The adjacency matrix of $\mathcal{G}$ is denoted as $\mathbf{A}$ with $A_{i,j} > 0$ iff $(v_i, v_j) \in \mathcal{E}$, and $A_{i,j} = 0$ otherwise.

Degree matrix. The degree matrix $\mathbf{D}$ of $\mathcal{G}$ is a diagonal matrix defined as

$$\mathbf{D} = \text{diag}(\mathbf{A1}) \tag{2.1}$$

where diag($\mathbf{x}$) is a diagonal matrix with $\mathbf{x}$ as its diagonal elements and $\mathbf{1}$ being the vector of all ones. We have $D_{i,i} = \sum_{j=1}^{N} A_{i,j}$ is the degree of node v_i. For a binary $\mathbf{A}$, the degree of a node is an integer, whereas for $\mathbf{A}$ with arbitrary weights, it is a scalar.

For directed graphs, there exist two types of degree matrices, namely the in-degree matrix $\mathbf{D}_i$ and out-degree matrix $\mathbf{D}_o$ defined respectively as

$$\mathbf{D}_i = \sum_{i=1}^{N} A_{i,j} \qquad \mathbf{D}_o = \sum_{j=1}^{N} A_{i,j}. \tag{2.2}$$

[1] In this thesis, we will consider graphs without self-loops, i.e., $(v_i, v_i) \notin \mathcal{G}$.

[2] We can define in and out-neighborhoods for directed graphs accordingly.

Laplacian matrix. The graph Laplacian [45] of an undirected $\mathcal{G}$ is defined as

$$\mathbf{L} = \mathbf{D} - \mathbf{A}. \tag{2.3}$$

Any off-diagonal element $L_{i,j}$ of $\mathbf{L}$ equals zero if $(v_i, v_j) \notin \mathcal{E}$ and equals $-A_{i,j}$ if $(v_i, v_j) \in \mathcal{E}$. The ith diagonal element $L_{i,i}$ is the degree of node i. By definition the graph Laplacian is symmetric.

All the shift operators considered in this dissertation have bounded elements, i.e., $S_{i,j}$ is always bounded. Other examples of GSOs include the normalized adjacency [46], normalized Laplacian [47, 48], and random-walk Laplacian [49, 50].

2.1.3. GRAPH SIGNAL

A graph signal is a mapping from the vertex to the space of real numbers. It is defined as

$$\mathsf{f} : \mathcal{V} \to \mathbb{R}. \tag{2.4}$$

We denote the graph signal of an N node graph as $\mathbf{x} = [x_1, \ldots, x_N]^\top$ where x_i is the signal associated to node v_i. Compared to typical signals in signal processing which are defined over time, graph signals are defined over the vertices of a graph.[3] The signals are inextricably linked to the topology of the graph itself and thus observing a graph signal can tell us something about the underlying topology [51].

2.1.4. GRAPH SIGNAL VARIABILITY

Variability of graph signals measures how fast a graph signal $\mathbf{x}$ varies over $\mathcal{G}$. Unlike temporal signals, the variability here depends on the graph structure. Depending on the GSO, we have two popular variability measures: the quadratic variation and the total variation.

Quadratic variation with L. For the graph Laplacian, the quadratic variation (QV) of a signal $\mathbf{x}$ is defined as

$$\mathsf{QV}(\mathbf{x}) = \mathbf{x}^\top \mathbf{L}\mathbf{x} = \sum_{i,j} A_{i,j}(x_i - x_j)^2 \tag{2.5}$$

which is the sum of the squared differences of graph signals across all edges multiplied by the edge weight. The quadratic variation is high if the graph signals at nodes which share an edge differ greatly.

Total variation with A. For the graph adjacency matrix, the total variation (TV) of a signal $\mathbf{x}$ is defined as

$$\mathsf{TV}(\mathbf{x}) = ||\mathbf{x} - \mathbf{A}_n\mathbf{x}||_2^2 \tag{2.6}$$

[3] A temporal signal can also be seen as a graph signal defined over a cyclic or acyclic time graph [46].

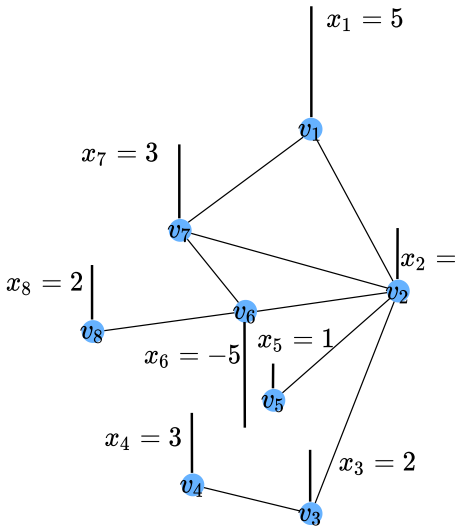


Figure 2.1.: A graph $\mathcal{G}$ with 8 nodes and graph signal $\mathbf{x} = [5,2,2,3,1,-5,3,2]^\top$. Each node v_i is associated with a scalar x_i.

where $\mathbf{A}_n = \frac{1}{|\lambda_{\max}|}\mathbf{A}$ where $\lambda_{\max}$ is the eigenvalue of $\mathbf{A}$ with the largest magnitude. Here $\mathsf{TV}(\mathbf{x})$ is the squared norm of the difference between the signal $\mathbf{x}$ and the output when the normalized GSO $\mathbf{A}_n$ acts on it. Higher variability implies that the signal at most nodes changes more after being operated upon by $\mathbf{A}_n$.

2.1.5. GRAPH SPECTRUM

In addition to the vertex domain, graphs can also be analyzed in the spectral domain [45]. The spectrum of a graph is the set of eigenvalues of its GSO obtained from the eigendecomposition

$$\mathbf{S} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^{-1} \tag{2.7}$$

where $\mathbf{V}$ is an $N \times N$ matrix and $\mathbf{\Lambda} = \text{diag}(\lambda_1, \ldots, \lambda_N)$ contains the eigenvalues on the main diagonal. The ordering of the eigenvalues of the GSO plays a key role in interpreting the spectrum of the graph.

Graph Laplacian ordering. Since the eigenvalues of the graph Laplacian are real (by virtue of $\mathbf{L}$ being symmetric), a natural question is how to arrange these eigenvalues so as to give a them a frequency interpretation to analyze graph signals similar to that in classical signal processing. To do so, we rely on the quadratic variation w.r.t the graph Laplacian in (2.5). It is of interest that the solution to the optimization

problem

$$\min_{\mathbf{x}} \mathbf{x}^\top \mathbf{L}\mathbf{x} \tag{2.8}$$

$$\text{subject to } ||\mathbf{x}||_2 = 1 \tag{2.9}$$

are the eigenvectors of $\mathbf{L}$ and the corresponding minimum values of the total variation are the eigenvalues. For example, for $\mathbf{x} = \frac{1}{\sqrt{N}}\mathbf{1}$, we have $\mathsf{QV}(\mathbf{x}) = 0$, ie., the eigenvector corresponding to a constant graph signal has zero variability, and thus zero denotes the lowest frequency. Thus, if $\lambda_i > \lambda_j$, then λ_i is associated with more frequency and variability.

Graph adjacency ordering. The eigenvalues of $\mathbf{A}$ can be both real and complex, and thus its eigenvalues need a different ordering scheme. For this, we consider the variability measure [46]

$$\mathsf{TV}(\mathbf{x}) = ||\mathbf{x} - \mathbf{A}_n\mathbf{x}||_1 \tag{2.10}$$

with $\mathbf{A}_n = \frac{1}{|\lambda_{\max}|}\mathbf{A}$. We first consider real eigenvalues. For an eigen pair $(\lambda_n, \mathbf{v}_n)$, the variation is

$$\mathsf{TV}(\mathbf{v}_n) = \left|1 - \frac{\lambda_n}{|\lambda_{\max}|}\right| ||\mathbf{v}_n||_1. \tag{2.11}$$

For two real eigenvalues λ_i and λ_i, if $\lambda_i > \lambda_j$, then $\mathsf{TV}(\mathbf{v}_i) < \mathsf{TV}(\mathbf{v}_j)$. Thus, smaller eigenvalues are associated with greater variability and therefore greater frequency. For complex eigenvalues the total variation depends on the distance between $|\lambda_{\max}|$ and λ_n in the complex plane. If λ_i and λ_j are two complex eigenvalues such that λ_i is located closer to $|\lambda_{\max}|$ in the complex plane than λ_j, we have $\mathsf{TV}(\mathbf{v}_i) < \mathsf{TV}(\mathbf{v}_j)$, i.e., λ_i has lower frequency than λ_j. Figure 2.2 illustrates the notion of frequency ordering for the Laplacian and the adjacency matrix.

2.1.6. GRAPH FOURIER TRANSFORM

The graph Fourier transform (GFT) of signal $\mathbf{x}$ w.r.t. a graph $\mathbf{S} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^{-1}$ is defined as

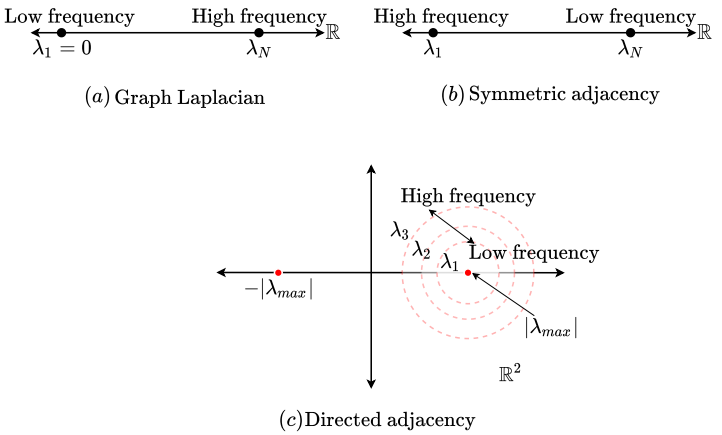
$$\hat{\mathbf{x}} = \mathbf{V}^{-1}\mathbf{x}. \tag{2.12}$$

The ith component $\hat{x}_i$ is the projection of $\mathbf{x}$ on to the ith row of $\mathbf{V}^{-1}$. Since the ith eigenvector denotes the variation associated with frequency λ_i, $\hat{x}_i$ denotes the component of $\mathbf{x}$ along the ith frequency. Depending on the frequency ordering of $\mathbf{S}$, we can analyse the frequency content in $\mathbf{x}$. For example when $\mathbf{S} = \mathbf{L}$, we have $\hat{x}_1 = \mathbf{v}_1^\top \mathbf{x} = \sum_{i=1}^N x_i$. The eigenvector $\mathbf{v}_1$ is a constant vector corresponding to the lowest Laplacian eigenvalue (zero). The value $\hat{x}_1$ is the lowest frequency component of $\mathbf{x}$ obtained by a sum of its elements. A higher value of $\hat{x}_i$ suggests that $\mathbf{x}$ has a higher component along the frequency corresponding to λ_i.

The inverse graph Fourier transform (IGFT) expresses the signal in the vertex domain in terms of its GFT $\hat{\mathbf{x}}$ as

$$\mathbf{x} = \mathbf{V}\hat{\mathbf{x}} = \sum_{i=1}^{N} \hat{x}_i \mathbf{v}_i. \tag{2.13}$$

For a graph signal, the IGFT expresses $\mathbf{x}$ as a linear combination of its eigenvectors.



(*a*) Graph Laplacian (*b*) Symmetric adjacency

(*c*) Directed adjacency

Figure 2.2.: Frequency ordering for some shift operators. (*a*) Graph Laplacian, where lower eigenvalues correspond to lower frequencies; (*b*) symmetric adjacency matrix, where lower eigenvalues correspond to higher frequencies; (*c*) directed adjacency, where eigenvalues closer to $|\lambda_{\max}|$ in the complex plane correspond to lower frequency.

2.2. GRAPH FILTERS

To understand graph filtering, we begin with the shift operation of a graph signal [7]. When the GSO acts on $\mathbf{x}$, we get

$$\mathbf{y} = \mathbf{S}\mathbf{x} \tag{2.14}$$

with output at node i

$$y_i = \sum_{j=1}^{N} S_{i,j} x_j = \sum_{j \in \mathcal{N}_i} S_{i,j} x_j. \tag{2.15}$$

Thus, the output at each node is the weighted sum of the signals at its neighbors. Successive applications of the shift operation spread the input graph signal $\mathbf{x}$ over different neighbourhoods in the graph.

Graph filters are the analogue of discrete-time filters in classical signal processing. For a graph $\mathcal{G}$ with GSO $\mathbf{S}$, a graph filter of order K is defined as

$$\mathbf{H}(\mathbf{S}) = \sum_{k=0}^{K} h_k \mathbf{S}^k \tag{2.16}$$

which is a polynomial in $\mathbf{S}$ with parameter h_k scaling $\mathbf{S}^k$. The vector $\mathbf{h} = [h_0, \ldots, h_K]^\top \in \mathbb{R}^{K+1}$ denotes the filter coefficients. Given a graph signal $\mathbf{x}$, we denote its filter output $\mathbf{y}$ as

$$\mathbf{y} = \mathbf{H}(\mathbf{S})\mathbf{x} = \sum_{k=0}^{K} h_k \mathbf{S}^k \mathbf{x} \tag{2.17}$$

which is a weighted sum of the successive signal shifts $\mathbf{S}^k\mathbf{x}$. The filter output $\mathbf{y}$ is the weighted combination of the different shifted versions of the graph signal $\mathbf{x}$ over $\mathcal{G}$. Figure 2.3 shows the pipeline for computing the filter output for an order three graph filter.

To compute (2.17), we note that each shift operation $\mathbf{Sx}$ is localized and incurs a complexity of order $\mathcal{O}(M)$ for a graph with M edges. We calculate in total K shifts recursively (as $\mathbf{S}^k\mathbf{x} = \mathbf{S}(\mathbf{S}^{k-1}\mathbf{x})$) and then scale and add them. Thus, the total complexity is of order $\mathcal{O}(MK)$.

Graph filters in the frequency domain. One distinct feature of graph filters is that they are interpretable in the frequency domain. By substituting the GSO eigendecomposition $\mathbf{S}^k = \mathbf{V}\mathbf{\Lambda}^k\mathbf{V}^{-1}$ into (2.17) we have

$$\mathbf{y} = \sum_{k=0}^{K} h_k \mathbf{V}\mathbf{\Lambda}^k\mathbf{V}^{-1}\mathbf{x}. \tag{2.18}$$

Multiplying on both sides by $\mathbf{V}^{-1}$, we have

$$\mathbf{V}^{-1}\mathbf{y} = \sum_{k=0}^{K} h_k \mathbf{\Lambda}^k\mathbf{V}^{-1}\mathbf{x}. \tag{2.19}$$

Following the definition of the GFT, we have the GFT of the filter output $\hat{\mathbf{y}}$ as

$$\hat{\mathbf{y}} = \sum_{k=0}^{K} h_k \mathbf{\Lambda}^k\hat{\mathbf{x}}. \tag{2.20}$$

The matrix $\mathbf{H}(\mathbf{\Lambda}) = \sum_{k=0}^{K} h_k\mathbf{\Lambda}^k$ is diagonal with the ith diagonal element $h(\lambda_i) = \sum_{k=0}^{K} h_k\lambda_i^k$. This represents the filter frequency response at ith graph frequency. The output GFT is thus a point-wise multiplication between the input GFT and the filter frequency response. Note that the frequency response of the graph filter is a polynomial $\mathsf{h}(\lambda) = \sum_{k=0}^{K} h_k\lambda^k$ evaluated at the N eigenvalues. This provides us a way to visualize and interpret the graph filter as a combination of this polynomial and its spectrum. Next, we highlight how graph filters perform graph signal de-noising and interpolation. We focus on these two applications as we will encounter them throughout the thesis.

De-noising with graph filters. Graph signal de-noising concerns recovering the true signal $\mathbf{x}_{\text{true}}$ given a noisy sample $\mathbf{x}$ [52]. Let $\mathbf{x} = \mathbf{x}_{\text{true}} + \mathbf{n}$ be the observed signal where $\mathbf{n}$ is the additive random noise. To recover $\mathbf{x}_{\text{true}}$, the filter learning problem translates to an optimization problem of the form

$$\mathbf{h}_{\text{den}} = \underset{\mathbf{h}}{\text{argmin}} \quad \mathbb{E}\left[||\mathbf{x} - \sum_{k=1}^{K} h_k\mathbf{S}^k\mathbf{x}||_2^2 \right] + \mathsf{r}(\mathbf{h}) \tag{2.21}$$

where $\mathbb{E}[\cdot]$ is the statistical expectation operator, $\mathsf{r}(\mathbf{h})$ is a regularizer to avoid overfitting such as the squared l_2 norm $||\mathbf{h}||_2^2$. Interested readers can refer to the works in [7, 53–55] which build on graph filters.

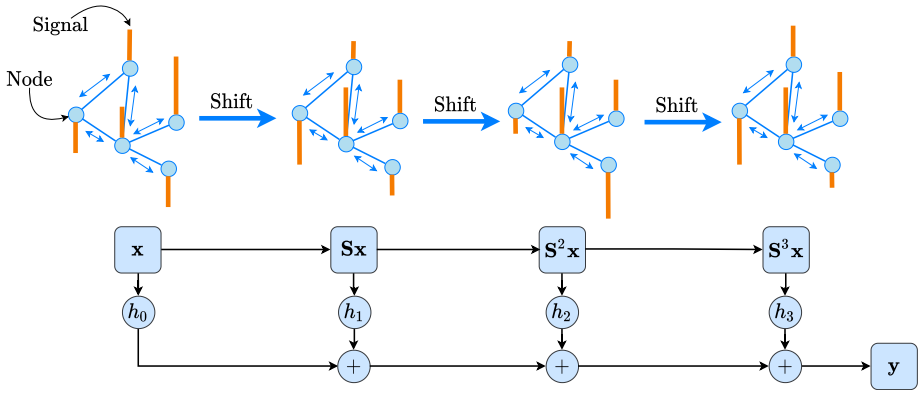


Figure 2.3.: Graph filter output for a filter of order three. The input is the signal $\mathbf{x}$, while $\mathbf{S}$ is the graph shift operator. The signal is successively shifted three times giving rise to the shifted signals $\mathbf{S}^k\mathbf{x}$. These are weighted via the filter coefficients h_k and combined to give the output $\mathbf{y} = \sum_{k=0}^{3} h_k \mathbf{S}^k \mathbf{x}$.

Signal interpolation. Graph signal interpolation concerns estimating the signal at unobserved nodes given the structure and the signals at some observed nodes [56]. Let $\mathbf{\Phi}$ be an $M \times N$ binary sampling matrix. We observe the signal $\mathbf{\Phi x}$, which is a subset of the original signal $\mathbf{x}$. The filter $\mathbf{H}(\mathbf{S})\mathbf{x}$ for this task is learnt as

$$\mathbf{h}_{\text{int}} = \underset{\mathbf{h}}{\operatorname{argmin}} \quad \mathbb{E}\left[||\mathbf{\Phi}(\mathbf{x} - \sum_{k=1}^{K} h_k \mathbf{S}^k \mathbf{x})||_2^2 \right] + \mathsf{r}(\mathbf{h}). \tag{2.22}$$

The trained filter $\mathbf{h}_{\text{int}}$ is then evaluated at the nodes with unobserved values. The works in [7, 57–60] also use graph filters to solve interpolation-related problems.

2.3. DYNAMIC GRAPHS

In this section, we focus on dynamic graphs, mostly concerning temporal dynamics, which is essential to this dissertation. A dynamic graph is a sequence of graphs $\mathcal{G}_0, \mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_T$ with the graph at time t being $\mathcal{G}_t = \{\mathcal{V}_t, \mathcal{E}_t\}$ with node set $\mathcal{V}_t$ of N_t nodes and edge set $\mathcal{E}_t$. Let $\mathbf{A}_t \in \mathbb{R}^{N_t \times N_t}$ be the adjacency matrix of $\mathcal{G}_t$. Considering the scope of this dissertation, we divide dynamic graphs into two distinct categories: dynamic non-expanding graphs and dynamic expanding graphs.

Dynamic non-expanding graphs. This category involves graphs where the nodes remain static, i.e., $N_t = N_0$ for all t, but the edge set $\mathcal{E}_t$ changes with time. Figure 2.4 illustrates this type of dynamic graph. We can further categorize this class of dynamic graphs into two categories: deterministic and stochastic dynamic non-expanding graphs.

- *Deterministic dynamic non-expanding graphs.* Here we know precisely how the topology changes, i.e., $\mathcal{E}_{t+1}$ is deterministic, given $\mathcal{E}_t$. A common example can

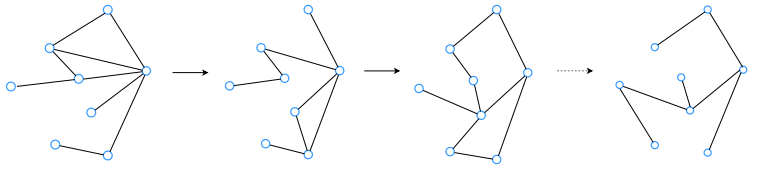


Figure 2.4.: A dynamic graph with fixed node set $\mathcal{V}$ but changing edge set.

be seen in recommender systems where the graph between the existing users changes with time along with the preferences of the users [61]. Mathematically, we can represent $\mathbf{A}_{t+1}$ as a function of $\mathbf{A}_t$ as

$$\mathbf{A}_{t+1} = \mathbf{A}_t + \mathbf{\Delta}_t \tag{2.23}$$

where $\mathbf{\Delta}_t$ is a known $N_0 \times N_0$ matrix which can modify the edge weights. For example, if an edge exists between nodes v_i and v_j at time t but disappears at time $t+1$, we have $[\mathbf{\Delta}_t]_{i,j} = -[\mathbf{A}_t]_{i,j}$. Likewise, if an edge is absent between v_i and v_j at time t but appears at time $t+1$, we have $[\mathbf{\Delta}_t]_{i,j} = [\mathbf{A}_{t+1}]_{i,j}$.

- *Stochastic dynamic non-expanding graphs.* Here, the evolution of the topology follows a stochastic model. Thus, we cannot determine exactly what the topology at a certain time will be, but can formulate it stochastically. Common examples include graph sequences where edges are added or deleted at each time instant obeying some probabilities [62]. Another popular example is that of edge-rewiring, popular in network science models [63, 64]. We represent the adjacency matrices of such a dynamic graph via their expectations, i.e.,

$$\mathbb{E}[\mathbf{A}_{t+1} | \{\mathbf{A}_0, \ldots, \mathbf{A}_t\}] = \mathbf{P}_{t+1} \tag{2.24}$$

where $\mathbf{P}_t$ is the $N_0 \times N_0$ probability matrix and $[\mathbf{P}_{t+1}]_{i,j}$ denotes the probability with which an edge between v_i and v_j exists at time $t+1$. Knowledge of all the $\mathbf{P}_t$s allows us to fully describe the dynamic graph model. Note that the expectation in (2.24) is conditional as the dependence on the previous topologies is determined by the nature of the dynamics.

Expanding graphs. This category involves graph sequences where the number of nodes grows. Thus, we have $\mathcal{V}_0 \subseteq \mathcal{V}_1 \ldots \subseteq \mathcal{V}_T$ with $N_1 \leq N_2 \ldots, \leq N_T$. At time t, new nodes attach to the graph, thus accounting for the growth in size. Figure 2.5 illustrates this process where at each time instant the graph grows by one node. In this figure, we assume that an edge once formed does not get removed, i.e., $\mathcal{E}_0 \subset \mathcal{E}_1 \ldots \subset \mathcal{E}_T$. We will also use this assumption throughout the dissertation, unless specified otherwise. Like in the non-expanding case, we also have a deterministic and stochastic counterpart of the expanding scenario.

- *Deterministic dynamic expanding graphs.* We know precisely how the incoming node attaches to the existing graph. To describe this mathematically, let $\mathbf{A}_t$ be

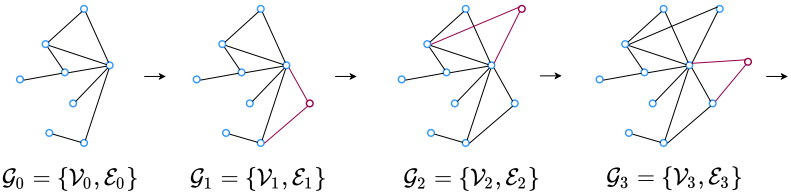


Figure 2.5.: A dynamic expanding graph starting from graph $\mathcal{G}_0 = \{\mathcal{V}_0, \mathcal{E}_0\}$. At each time instant one node (in magenta) attaches to the existing graph. The new edges are also shown in magenta to highlight the dynamic.

the adjacency matrix at time t. Let v_t be the incoming node, which leads to $\mathbf{A}_{t+1}$. The attachment of node v_t is characterized by vector $\mathbf{a}_t \in \mathbb{R}^{N_t}$ such that $[\mathbf{a}_t]_i = w_{t,i}$ if v_t attaches to $v_i \in \mathcal{V}_t$ with edge-weight $w_{t,i}$, and zero otherwise. If $\mathbf{A}_{t+1}$ is undirected, it writes in terms of $\mathbf{A}_t$ as

$$\mathbf{A}_{t+1} = \begin{bmatrix} \mathbf{A}_t & \mathbf{a}_t \\ \mathbf{a}_t^\top & 0 \end{bmatrix}. \tag{2.25}$$

For the case where $\mathbf{A}_{t+1}$ is directed, we have

$$\mathbf{A}_{t+1} = \begin{bmatrix} \mathbf{A}_t & \mathbf{b}_t \\ \mathbf{a}_t^\top & 0 \end{bmatrix} \tag{2.26}$$

where the vector $\mathbf{b}_t \in \mathbb{R}^{N_t}$ denotes edges directed from v_t to the existing nodes, while $\mathbf{a}_t$ has edges directed from the existing nodes to v_t. A specific case of directed attachment is the adjacency matrix

$$\mathbf{A}_{t+1} = \begin{bmatrix} \mathbf{A}_t & \mathbf{0} \\ \mathbf{a}_t^\top & 0 \end{bmatrix} \tag{2.27}$$

which has edges directed only from the existing to the incoming nodes. Here $\mathbf{A}_t$ represents the existing user similarity graph and $\mathbf{a}_t$ the vector of similarities between the new user and the existing ones. Often $\mathbf{a}_t$ is sparse, so only a finite number of similarities are retained. Differently from (2.25), such a directed nature is more suited for inference tasks at the incoming nodes. Normally for an inference task, we expect the existing nodes to influence the incoming ones and not the other way around. This is the case of cold-starters in graph-based collaborative filtering [10, 65]. Here, the nodes represent existing users, the edges capture similarities among them (e.g., Pearson correlation), and a cold starter is a new node that attaches to this user-user graph. The task is to collaboratively infer the preference of the cold-starter from the existing users [66]. This occurs in growing physical networks or in collaborative filtering where side information is used to establish the connectivity [10].

- *Stochastic dynamic expanding graphs.* Here we do not know precisely how the incoming node attaches to the existing graph. This is often the case when very

little information is available about the new nodes, a scenario broadly defined as the cold-start problem [17]. Like we do in the non-expanding scenario, we rely on stochastic models governed by probabilities to obtain an expected formulation of $\mathbf{A}_{t+1}$. We consider v_t connects independently to each existing $v_i \in \mathcal{V}_t$ with probability $p_{t,i}$, forming an edge with weight $w_{t,i}$. Thus, the attachment vector $\mathbf{a}_t$ is random with each entry being a weighted Bernoulli random variable; i.e,

$$[\mathbf{a}_t]_i = \begin{cases} w_{t,i} & \text{with probability } p_{t,i} \\ 0 & \text{with probability } (1 - p_{t,i}) \end{cases} \tag{2.28}$$

for $i = 1, \ldots, N_t$. The expected value of $\mathbf{a}_t$ is

$$\mathbb{E}[\mathbf{a}_t] = \mathbf{p}_t \circ \mathbf{w}_t \tag{2.29}$$

where $\mathbf{p}_t = [p_{t,1}, \ldots, p_{t,N}]^\top$ is the vector of probabilities for v_t, $\mathbf{w}_t = [w_{t,1}, \ldots, w_{t,N}]^\top$ the corresponding edge-weight vector, and $\circ$ is the Hadamard product. Likewise, the variance of $[\mathbf{a}_t]_i$ is $\text{var}([\mathbf{a}_t]_i) = w_{t,i}^2 p_{t,i}(1 - p_{t,i})$ and the covariance matrix of $\mathbf{a}_t$ is

$$\mathbf{\Sigma}_t = \text{diag}(\mathbf{w}_t^{\circ 2} \circ \mathbf{p}_t \circ (\mathbf{1} - \mathbf{p}_t)) \tag{2.30}$$

where $\mathbf{a}^{\circ 2} := \mathbf{a} \circ \mathbf{a}$. The expected adjacency matrix for undirected graphs is

$$\mathbb{E}[\mathbf{A}_{t+1}] = \begin{bmatrix} \mathbb{E}[\mathbf{A}_t] & \mathbf{p}_t \circ \mathbf{w}_t \\ (\mathbf{p}_t \circ \mathbf{w}_t)^\top & 0 \end{bmatrix} \tag{2.31}$$

and the directed counterpart for inference is

$$\mathbb{E}[\mathbf{A}_{t+1}] = \begin{bmatrix} \mathbb{E}[\mathbf{A}_t] & \mathbf{0} \\ (\mathbf{p}_t \circ \mathbf{w}_t)^\top & 0 \end{bmatrix}. \tag{2.32}$$

Thus, we can write the attachment vector of a new realization as $\mathbf{a}_t = \mathsf{SV}(\mathbf{p}_t) \circ \mathbf{w}_t$, where $\mathsf{SV}(\mathbf{p}_t)$ is a binary vector obtained by sampling $\mathbf{p}_t$ element-wise. An example of the above scenario can be found for the cold-start problem in recommender systems, where the incoming user does not have enough information to reliably estimate the similarities with the existing users.

Directed attachment for inference.

The stochastic dynamic expanding graphs are strongly characterized by the probabilities of attachment $\mathbf{p}_t$. The choice of $\mathbf{p}_t$ influences the network evolution and determines properties of the dynamic graph. Next, we will introduce two often-used choices of $\mathbf{p}_t$ within the scope of this dissertation.

2.3.1. RANDOM GRAPH MODELS FOR DYNAMIC EXPANDING GRAPHS

A random graph model is a generative model which is used to derive instances of graphs. An instance is derived by sampling edges from underlying probabilities.

Each edge is thus associated with a random variable. We discuss two important types of random graph models.

Erdős Rényi random graphs. An instance of this model is a graph of N nodes where each possible edge exists with probability $p > 0$. Each edge corresponds to an independent and identically distributed (i.i.d.) Bernoulli random variable with mean p and variance $p(1-p)$. Very low values of p lead to disconnected graphs, while a higher p leads to less sparse graphs.

The Erdős Rényi model generates the whole graph. However, this can be adapted to the expanding setting. One way to do this is to assume each incoming node attaches to each node in the existing graph with the same probability. For the new node v_t, this implies that an edge is formed with probability $p_{t,i} = \frac{1}{N_t}$ for each existing node v_i. Often m edges are sampled from the N_t possible edges, each with probability $p_{t,i}$. Figure 2.6 illustrates how this attachment works with one incoming node.

Barabási-Albert random graphs. The Barabási-Albert (BA) or scale-free model [67] is based on the principle of preferential attachment, where new nodes do not attach uniformly-at-random but exhibit a preference in their attachment behaviour. This preference depends on the degrees of the existing nodes. Let the existing node v_i in an existing graph $\mathcal{G}_0$ of N nodes have degree d_i. A new node attaches to an existing node v_i with a probability $p_{t,i} = \frac{d_i}{\sum_{i=1}^{N} d_i}$. At each time step, each new node can form m links to the existing nodes.

There exists other attachment models which can generate expanding graphs, outside of the two mentioned above. Readers can refer to [19] for relevant examples.

Examples. To conclude this section, we focus on two examples of dynamic graphs as discussed in Section 2.3 and tie them to tasks we are interested in.

- *Identifying dominant interactions in social communication networks.* We consider an email communication network among the employees of an organization [68]. The employees are treated as nodes and edges exist between employees who communicate using emails. By observing the emails sent over days, we can have a dynamic graph over time. Graph $\mathcal{G}_t$ represents the emails sent between users at time slot t. Figure 2.7 illustrates this concept. Another example of a communication network is that of school children interacting with each other over a span of time, where the interactions are recorded through sensors [69]. These examples belong to the dynamic non-expanding graph scenario. Sometimes, there can also be data associated with the nodes, also known as graph signals. These can be attributes like the age and address of employees and the age, gender and social background of the schoolkids.

- *Predicting ratings for new users.* Our second example involves a movie recommender system. Consider a graph with users as nodes. Edges exist between users who have a similar preference of movies based on their previous ratings. The graph signal is the set of ratings provided by these users for a particular movie. Now we have a new user and we want to predict how this user would rate this particular movie. We incorporate this new user into

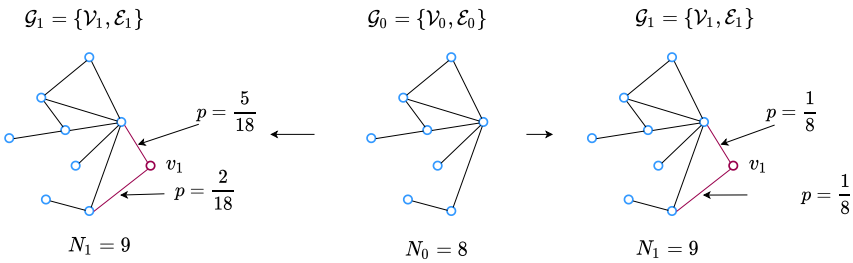


Figure 2.6.: Addition of a new node v_1 (magenta) to an existing graph $\mathcal{G}_0$ with $N_0 = 8$ nodes following two types of stochastic attachment: (Left) the degree-based preferential attachment. The probability of formation of edges is proportional to the degree of the existing node. The first edge is formed with an existing node of degree five with probability $\frac{5}{18}$. The other edge is formed with an existing node of degree two with probability $\frac{2}{18}$. The node with higher degree will have a higher tendency to form edges with incoming nodes. In this example node v_1 forms two edges; (Right) the uniformly-random attachment, where the probability of formation of all the eight possible edges is $\frac{1}{8}$.

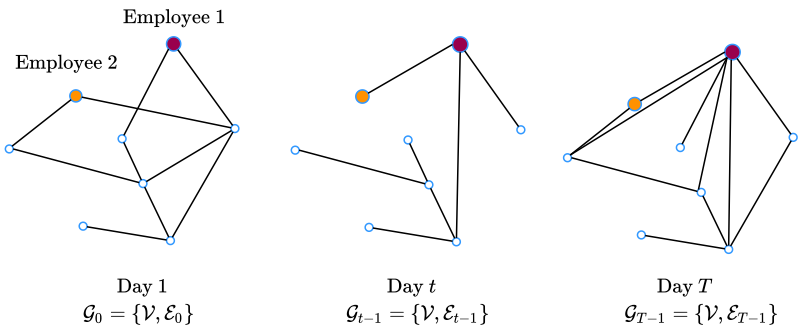


Figure 2.7.: A dynamic email communication networks. The nodes represent the employees, the edges the email patterns between them, and the signal the respective attributes. We assume the process starts from day one, which corresponds to $t = 0$. The tth graph $\mathcal{G}_t$ represents all the emails sent on day t. Also highlighted are two coloured nodes representing two employees.

the graph after making the prediction. This corresponds to a graph signal interpolation problem, where we predict the signal at the incoming node. One way to do this is via graph filters designed for this purpose. The scenario here can be both the deterministic and stochastic dynamic expanding graph. In the deterministic case, the updated topology is readily available. In the stochastic

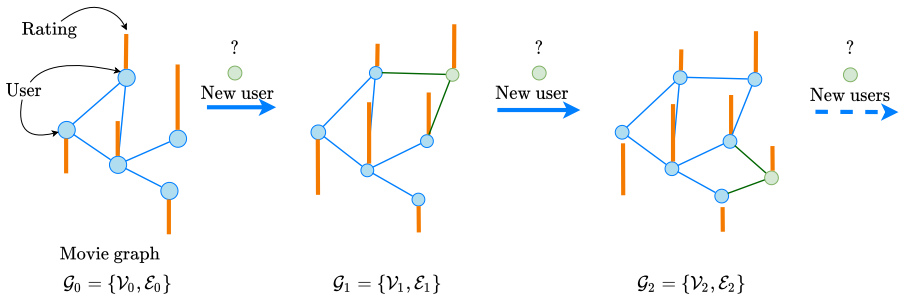


Figure 2.8.: Product recommendation via user-user collaborative filtering in an expanding graph setting. The blue nodes denote the existing users, while the green nodes denote new users. The blue edges denote similarities between existing users while the green edges denote those between the existing and new users at a given time. The graph signal denotes the ratings given to a particular movie by the users, both existing and new.

case, we do not know how a new node attaches, so we depend on the expected topology [cf. (2.31)] to design filters. In this way, new users keep arriving and the graph size grows. This creates a dynamic expanding graph. Note that the information about the new links may not always be available. We will discuss these cases throughout the dissertation. Irrespective of the scenario, filter design is challenging as the graph and signal may both change with time.

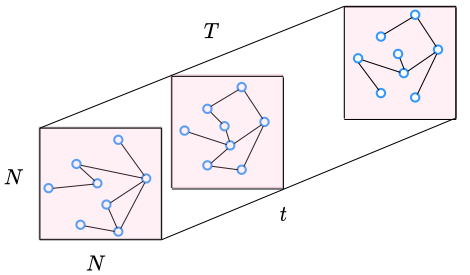


Figure 2.9.: A three-dimensional tensor representation of a dynamic network. We have N and T as the number of nodes and temporal samples, respectively. Each slice along time denotes a graph observed at that time.

2.4. TENSORS FOR DYNAMIC TOPOLOGY REPRESENTATION

Let us recall our definition of dynamic graphs, i.e., a sequence of graphs $\mathcal{G}_1, \ldots, \mathcal{G}_T$ with graph $\mathcal{G}_t = \{\mathcal{V}_t, \mathcal{E}_t\}$ having adjacency matrix $\mathbf{A}_t$. We consider the dynamic non-expanding scenario, where each graph $\mathcal{G}_t$ has N nodes. We represent this dynamic graph through a three-dimensional tensor $\underline{\mathbf{A}} \in \mathbb{R}^{N \times N \times T}$ where the first two

dimensions are for the nodes and third dimension is along time [70]. That is, the adjacency matrices $\mathbf{A}_1,\ldots,\mathbf{A}_T$ are stacked over the third dimension in $\underline{\mathbf{A}}$. Such a representation structure can be exploited for a variety of tasks like network anomaly detection, community detection, link prediction, and representation [32–38]. Figure 2.9 illustrates this representation for a dynamic network of N nodes over T time instants.

2.4.1. DYNAMIC GRAPH TENSOR DECOMPOSITION

The structure of $\underline{\mathbf{A}}$ can be utilized to represent the evolving graph structure and provide insights about it. One way to accomplish this is by decomposing the three dimensional tensor $\underline{\mathbf{A}}$. These decompositions approximate a tensor with its low-rank components [71]. There are two relevant decompositions for this thesis: the canonical polyadic decomposition and the block term decomposition.

Canonical polyadic decomposition. The canonical polyadic decomposition (CPD) [71], also known as the PARAFAC, is one of the most common low-rank methods for tensors. Given a three-dimensional tensor $\underline{\mathbf{A}} \in \mathbb{R}^{N\times N\times T}$, the R-term CPD decomposition is

$$\underline{\mathbf{A}} \approx [[\mathbf{F},\mathbf{G},\mathbf{H}]] = \sum_{r=1}^{R} \mathbf{f}_r \circ \mathbf{g}_r \circ \mathbf{h}_r \tag{2.33}$$

where $\mathbf{F} = [\mathbf{f}_1,\ldots,\mathbf{f}_R]$, $\mathbf{G} = [\mathbf{g}_1,\ldots,\mathbf{g}_R]$, and $\mathbf{H} = [\mathbf{h}_1,\ldots,\mathbf{h}_R]$ are the factor matrices of size $N\times R$, $N\times R$, and $T\times R$, respectively[4]. The rth decomposition term in (2.33) is a $N\times N\times T$ tensor equal to $\mathbf{f}_r\circ\mathbf{g}_r\circ\mathbf{h}_r$, which is the outer product of three vectors. Figure 2.10 illustrates the CPD decomposition of $\underline{\mathbf{A}}$. The ith element of $\mathbf{f}_r$ can be thought as a one-dimensional embedding of node i. The same holds for the ith element of $\mathbf{g}_r$. If the adjacency matrices are directed, then the elements of $\mathbf{f}_r$ and $\mathbf{g}_r$ can encode embeddings corresponding to the nature of directed edges. The tth element of $\mathbf{h}_r$ scales the matrix $\mathbf{f}_r\mathbf{g}_r^\top$ and thus controls its presence over time. In literature, there are some works which apply CPD to the adjacency tensor $\underline{\mathbf{A}}$ for downstream tasks. For example, [32, 33] use it to detect communities among a set of fixed nodes in a dynamic graph. The authors in [34] do the same for dynamic graph summarization. The CPD has also been used for tasks like link prediction [37] and anomaly detection [38].

A drawback of the CPD is that each low-rank factors is an outer product of vectors. While this may be helpful for task-dependent embeddings, it limits the type of structure each factor can represent. The outer product of two vectors has a very specific structure, whereas the structure of observed adjacency matrices is significantly different. It may be possible to improve upon this representation by assuming each factor to be an outer product of matrices, thus leading us to the Black Term Decomposition.

Block term decomposition. The Block Term Decomposition (BTD) is another low-rank approximation for a tensor [72]. The $(L_r, L_r, 1)$ R-term BTD decomposition

[4] $\mathbf{H}$ and $\mathbf{h}_i$ do not denote a filter and filter parameters but general parameters that we will use disjointly in the upcoming chapters.

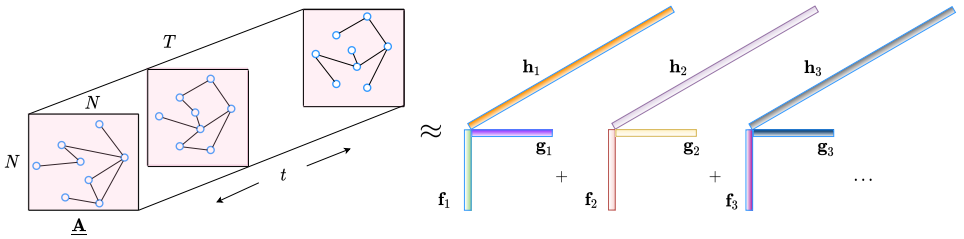


Figure 2.10.: Canonical polyadic decomposition of the adjacency tensor $\underline{\mathbf{A}}$ formed by stacking the adjacency matrices. The rth low-rank factor on the right is the outer product of three vectors: $\mathbf{f}_r$, $\mathbf{g}_r$, and $\mathbf{h}_r$. The first two represent the features corresponding to each node. Vector $\mathbf{h}_r$ is the temporal factor which scales the representation matrix $\mathbf{f}_r\mathbf{g}_r^\top$ along time.

is

$$\underline{\mathbf{A}} \approx [[\mathbf{F},\mathbf{G},\mathbf{H}]] = \sum_{r=1}^{R} \mathbf{F}_r \circ \mathbf{G}_r \circ \mathbf{h}_r \tag{2.34}$$

where $\mathbf{F} = [\mathbf{F}_1,\ldots,\mathbf{F}_R]$, $\mathbf{G} = [\mathbf{G}_1,\ldots,\mathbf{G}_R]$, and $\mathbf{H} = [\mathbf{h}_1,\ldots,\mathbf{h}_R]$ are the factor matrices. The rth term in the decomposition is the outer product of $\mathbf{F}_r$, $\mathbf{G}_r$ and $\mathbf{h}_r$, where the rank of each $\mathbf{F}_r$ and $\mathbf{G}_r$ is L_r, ultimately, justifying the terminology $(L_r, L_r, 1)$. Differently from the CPD, each element of $\mathbf{h}_r$ now scales the matrix $\mathbf{F}_r\mathbf{G}_r^\top$ which can represent more structure than the rank one matrix $\mathbf{f}_r\mathbf{g}_r^\top$. Additionally the embedding corresponding to each node in the rth low-rank factor is now a vector instead of a scalar, meaning the BTD can capture embeddings with more information about the nature of the node in the dynamic graph. Figure 2.11 illustrates the $(L_r, L_r, 1)$ BTD for $\underline{\mathbf{A}}$. There are not many works that apply BTD to graph sequences. The work in [36] applies BTD on tensors formed by multi-view graphs, i.e., each slice of the tensor representing a different relationship between the nodes and not necessarily a temporal evolution of the relationship. Other examples of tensor decompositions include [73–75].

2.5. CONCLUSION

In this chapter, we defined graphs and their algebraic representations in terms of the shift operators, namely the adjacency and Laplacian matrix. We also introduced the graph signal and its variability following the convention of the graph signal processing literature. Next, we discussed the spectrum of a graph through the eigen-decomposition of its shift operator. We use this to analyze the frequency response of filters. We introduced graph filters as a combination of successively shifted graph signals via the filter parameters. In Chapters 3, 4, and 5, the learning of such filters shall be the key tool used for processing graph signals over dynamic and expanding graphs. We will focus mostly on interpolation and de-noising tasks, but extensions to other tasks are straightforward.

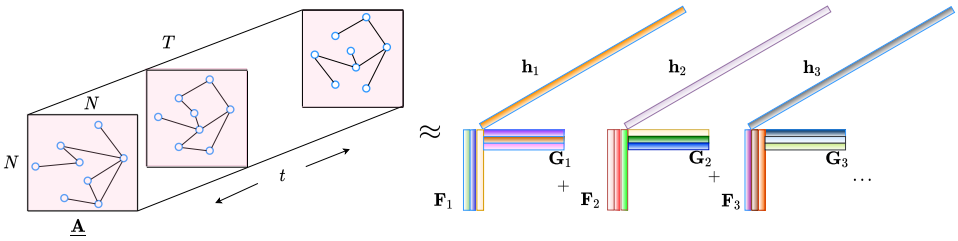


Figure 2.11.: Block Term Decomposition of the adjacency tensor $\underline{\mathbf{A}}$ formed by stacking the adjacency matrices. The rth low-rank factor on the right is the outer product of matrices $\mathbf{F}_r$ and $\mathbf{G}_r$, and vector $\mathbf{h}_r$. A row of $\mathbf{F}_r$ represents a vector embedding of a node in the dynamic graph. Similarly, a row of $\mathbf{G}_r$ represents another embedding of the same dimension for a node. The vector $\mathbf{h}_r$ is a temporal factor which scales the representation matrix $\mathbf{F}_r\mathbf{G}_r^\top$ along time.

In the second part of this chapter, we focused on dynamic graphs and their representation. We defined dynamic graphs over a sequence and focused on two types: one involving fixed nodes with varying edges, and the other involving growing graphs with incoming nodes and an addition of edges. For both, we discuss their deterministic and stochastic counterparts. We will focus on filter design for deterministic and stochastic expanding graphs in Chapters 3, 4, and 5. In Chapter 6 we consider dynamic non-expanding graphs for their representation. Related to this, we focused on two random graph models, namely the uniformly-at-random and preferential attachment models and showed how they can be used to make a growing graph model. We will use these two models as baselines throughout Chapters 3, 4, and 5. In Chapter 3, we deal with growing graph models based on probabilities of attachment but we opt to learn them instead based on different constraints and scenarios. Next, we highlighted how dynamic graphs can be represented by tensors by stacking their adjacency matrices. We defined two low-rank decompositions of these tensors to find an embedding for the nodes or solve downstream tasks. We focus particularly on a variation of the block term decomposition as we want to highlight its main difference with the canonical polyadic decomposition. We use them as baselines in Chapter 6, where we propose a new decomposition of dynamic graphs along a different philosophy.

3

LEARNING TASK-AWARE EXPANDING GRAPHS

In this chapter, we concern ourselves with the task of processing data over incoming nodes where their connectivity information is unknown. Existing approaches assume full topology information, thus enabling standard signal processing tools over graphs; Moreover, existing approaches to model the growth of graphs are often unaware of the data existing on the graphs and any downstream task. We aim to fill this gap by considering a parametric stochastic attachment model that can adapt to a particular task from the existing topology and graph signals. In particular, we assume that the attachment behaviour and information associated with new nodes can be learned from how previous nodes are attached to the same graph, concerning an underlying data-processing or topological task. We learn the parameters via empirical risk minimisation for two specific tasks, namely graph signal interpolation using graph filters and signal smoothness.

The rest of this chapter is organised as follows: Section 3.1 introduces the problem and highlights our contributions; Section 3.2 elaborates how different disciplines approach the task of incoming node attachment; Section 3.3 formulates the task of inferring the stochastic attachment behaviour as an empirical risk minimization, given a training data-set of previously attached nodes; Section 3.4 adapts the formulated problem to two graph signal processing related themes: interpolation and smoothness; Section 3.5 conducts a perturbation analysis of the stochastic attachment behaviour w.r.t the model parameters; Section 3.6 contains the numerical results. Section 3.7 concludes the chapter. All proofs are collected in Appendix A.[1]

[1]This chapter is based on the publication: Das, B., Hanjalic, A., & Isufi, E. (2022). Task-aware connectivity learning for incoming nodes over growing graphs. IEEE Transactions on Signal and Information Processing over Networks, 8, 894-906.

3.1. INTRODUCTION

Graph topology identification is a crucial step preceding the analysis of relationships of users in social networks [2], proteins in biological networks [1], and entities in recommender systems [10], to name just a few. Typical approaches infer a topology with a fixed number of nodes [76, 77] but graphs often grow with new nodes attaching to the existing ones [14]. This attachment is often unknown, making the downstream tasks more challenging. One such task is the cold start recommendation in collaborative filtering [17]. Here, a new user enters the system but cannot attach to the existing ones in the absence of prior information, thereby affecting the subsequent recommendation. In another scenario, a new online political blog becomes available and we want to know how it associates with the existing political blogs without knowing its affiliation and its influence on the overall network [78].

We want to process data at the incoming node in situations when the true connectivity is not known. This may be the case when side information in collaborative filtering is unavailable or when new blogs are not associated with a particular category. At the same time, we want to process data in the most informed way possible. One way is to consider how the previous nodes connect and apply that rule for the incoming node. Differently, we want to handle the attachment in the context of a data-processing task over the network, focused at the new node. Taking the task into consideration will lead to more relevant attachments, possibly improving upon the performance of other attachment rules.

Existing graph identification approaches infer the full [76] or partial [77] connectivity of a fixed graph but do not consider incoming nodes, while stochastic approaches model the connectivity with a known attachment model [13, 14, 27, 79] but ignore the existing data and how it ought to be processed. Some recent works that process data over expanding graphs require the connectivity knowledge [28, 80], which is often unavailable. Thus, information processing for incoming nodes in such situations is challenging.

One way to overcome this challenge is to consider a data-driven stochastic model, where the available data is leveraged to learn the mapping between the incoming nodes and the existing graph for the task at hand. When combined with a prediction mechanism, such a hybrid approach can overcome the limitations of purely stochastic or data-oriented predictions.

3.1.1. CONTRIBUTIONS OF THIS CHAPTER

The main contribution of this chapter is the development of a stochastic attachment model parameterized by the probabilities of attachment and the edge-weights for incoming nodes given a graph signal processing (GSP) task without connectivity information. More specific contributions are as follows:

1. We specialize the model to two tasks. The first task is graph signal interpolation, where we predict the signal value at the incoming node. The second task is to learn the attachment of the incoming node such that the signal is smooth over the expanded graph. Thus, we combine preferential attachment with GSP and topology identification for modelling the incoming node connectivity.

2. We propose an empirical risk minimization problem to estimate the model parameters and solve them using alternating projected gradient descent. We discuss the convergence properties of this approach to a local minimum.

3. We study the learned connectivity from a perturbation viewpoint. We look at the small perturbation analysis of the eigenvalues of the nominal graph relative to the model realizations. We corroborate the proposed approach with numerical results on synthetic and real data from recommender systems and blog networks.

3.2. RELATED WORKS

Inferring node connections has been approached from different viewpoints ranging from GSP to statistical models. Here, we cast our work w.r.t. existing frameworks.

Graph signal processing (GSP). Topology identification via GSP estimates a fixed topology from data by leveraging different priors, such as signal smoothness [81–83], realizations of a diffusion process [84–87], or a Gaussian process [88–90], to name a few. These priors have also been used to estimate time-varying topologies where a fixed topology is estimated per batch of data [91, 92]. More recently, *online methods* avoid batch processing and estimate the topology on the fly from time-varying signals [93–97]. Differently, we will learn a stochastic model for incoming nodes rather than a fixed topology. And differently from the online methods, we consider an expanding topology but with a fixed time-invariant signal. As in these approaches, we will also consider the smoothness criterion, which is typically encountered in practice because of homophily (i.e., connected nodes share similar values) [98].

Statistical methods. Connectivity of incoming nodes is commonly approached in network science via stochastic models, such as the *Erdős-Rényi* (ER) and the *preferential attachment*. The ER model assumes each incoming node connects uniformly at random with any of the existing nodes [13], while preferential attachment assumes each incoming node connects with a probability proportional to a node's degree [14]. More complex models include a competition factor between nodes [27, 79]. These methods focus on the existing topology and do not account for the data over it. Accounting for the data is paramount to solving network learning tasks because of the implicit data-topology coupling. Therefore, we propose to estimate the attachment model parameters, i.e., attachment probabilities and edge weights of the incoming nodes by incorporating both the data and the task into the learning.

Link prediction. Modelling the connectivity of incoming nodes can also be seen as a link prediction task given some topological and nodal features [77, 99]. Link prediction techniques can be grouped into three categories: i) *probabilistic approaches* that use random models to predict links using, for example, hierarchical graphs and stochastic block models [100, 101]; ii) *similarity-based approaches* that predict a link between any two nodes based on their common neighborhood features [102, 103] or global graph features [104, 105]; and iii) *classifier-based approaches* that train a machine learning model based on node features. However, most of

these approaches fail in the incoming node scenario because we have no topological information about the incoming nodes and, in the absence of node features, classifier-based approaches are also inapplicable.

Learning on expanding graphs. Lastly, recent works consider specific expanding graph models or solve a specific task with the knowledge of the connectivity. The works in [106, 107] focus on estimating node connectivity for ER and Bollobás-Riordan models by observing auto-regressive signals on some nodes. Differently, we propose a data-driven approach that is agnostic to the graph and signal model. The works in [28, 80] solve regression tasks over expanding graphs but assume the connectivity is known. Instead, we consider the connectivity to be unknown. All in all, the proposed method stands at the intersection of preferential attachment and data-driven topology estimation to learn the stochastic model parameters w.r.t. a task-specific cost function.

3.3. PROBLEM FORMULATION

Consider an undirected graph $\mathcal{G}_0 = (\mathcal{V}_0, \mathcal{E}_0)$ of N nodes in set $\mathcal{V}_0 = \{v_{0,1}, \dots, v_{0,N}\}$ and M edges in set $\mathcal{E}_0 \subseteq \mathcal{V}_0 \times \mathcal{V}_0$. Let $\mathbf{A}$ be the graph adjacency matrix with $A_{i,j} > 0$ only if $(v_{0,i}, v_{0,j}) \in \mathcal{E}$ and $\mathbf{L}_0 = \text{diag}(\mathbf{A1}) - \mathbf{A}$ be the graph Laplacian. When an incoming node v_1 connects to $\mathcal{G}_0$, it forms the expanded graph $\mathcal{G}_1 = (\mathcal{V}_1, \mathcal{E}_1)$ with node set $\mathcal{V}_1 = \mathcal{V}_0 \cup v_1$ and edge set $\mathcal{E}_1 = \mathcal{E}_0 \cup (v_1, v_{0,i})$ for all new connections $(v_1, v_{0,i})$. The attachment of node v_1 is characterized by vector $\mathbf{a}_1 \in \mathbb{R}^N$ such that $[\mathbf{a}_1]_i = w_i$ if v_1 attaches to $v_{0,i}$ with edge-weight w_i, and zero otherwise. This leads to the respective expanded adjacency and Laplacian matrices

$$\mathbf{A}_1 = \begin{bmatrix} \mathbf{A} & \mathbf{a}_1 \\ \mathbf{a}_1^\top & 0 \end{bmatrix}, \ \mathbf{L}_1 = \begin{bmatrix} \mathbf{L}_0 + \text{diag}(\mathbf{a}_1) & -\mathbf{a}_1 \\ -\mathbf{a}_1^\top & \mathbf{a}_1^\top \mathbf{1} \end{bmatrix} \tag{3.1}$$

in which the last row and column represent the connectivity of v_1 with the nodes in $\mathcal{V}_0$.[2]

We consider v_1 connects independently to each existing $v_{0,i} \in \mathcal{V}_0$ with probability p_i. Thus, the attachment vector $\mathbf{a}_1$ is random with each entry being a weighted Bernoulli random variable; i.e,

$$[\mathbf{a}_1]_i = \begin{cases} w_i & \text{with probability } p_i \\ 0 & \text{with probability } (1 - p_i) \end{cases} \tag{3.2}$$

for $i = 1, \dots, N$. The expected value of $\mathbf{a}_1$ is $\mathbb{E}[\mathbf{a}_1] = \mathbf{p} \circ \mathbf{w}$ where $\mathbf{p} = [p_1, \dots, p_N]^\top$, $\mathbf{w} = [w_1, \dots, w_N]^\top$, and $\circ$ is the Hadamard product. Likewise, the variance of $[\mathbf{a}_1]_i$ is $\text{var}([\mathbf{a}_1]_i) = w_i^2 p_i (1 - p_i)$ and the covariance matrix of $\mathbf{a}_1$ is

$$\mathbf{\Sigma}_1 = \text{diag}(\mathbf{w}^{\circ 2} \circ \mathbf{p} \circ (\mathbf{1} - \mathbf{p})) \tag{3.3}$$

[2] Like in the fundamental studies about growing networks [13, 14], we consider for simplicity of exposition the attachment of a single node. However, our findings extend to multiple incoming nodes with appropriate modifications. E.g., making vector $\mathbf{a}_1$ in (3.1) a matrix, in which each column corresponds to one incoming node.

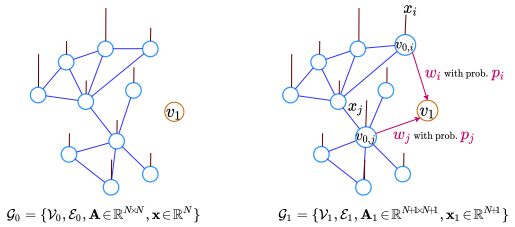


Figure 3.1.: Stochastic attachment model of incoming node. Incoming node v_1 attaches to each existing node with a certain weight and probability. In this illustration it forms two edges with node $v_{0,i}$ and $v_{0,j}$ of weights w_i and w_j with probability p_i and p_j, respectively.

where $\mathbf{a}^{\circ 2} := \mathbf{a} \circ \mathbf{a}$. The expected topology of $\mathcal{G}_1$ has the adjacency matrix

$$\mathbb{E}[\mathbf{A}_1] = \begin{bmatrix} \mathbf{A} & \mathbf{p} \circ \mathbf{w} \\ (\mathbf{p} \circ \mathbf{w})^\top & 0 \end{bmatrix}. \tag{3.4}$$

Thus, we can write the attachment vector of a new realization as $\mathbf{a}_1 = \mathsf{SV}(\mathbf{p}) \circ \mathbf{w}$, where $\mathsf{SV}(\mathbf{p})$ is a binary vector obtained by sampling $\mathbf{p}$ element-wise. Figure 3.1 illustrates this attachment for v_1 which forms two edges.

On the vertices of the existing graph $\mathcal{G}_0$, we have a graph signal $\mathbf{x}_0 = [x_{0,1}, \ldots, x_{0,N}]^\top$ in which entry $[\mathbf{x}_0]_i$ is the value on node $v_{0,i}$. Processing this signal by accounting for its coupling with $\mathcal{G}_0$ is key to several network data tasks. For instance, we use such coupling to predict the rating of a specific item in nearest-neighbour collaborative filtering [10]. In the incoming node setting, this translates to identifying the signal value x_1 for node v_1, e.g., the rating of a new user. Since we do not have the connectivity of v_1, we rely on stochastic models governed by the attachment probabilities $\mathbf{p}$ and weights $\mathbf{w}$, which are in turn unknown. We rely on the existing users and their connections to predict ratings because we don't know the connectivity, which stems from not knowing the existing preference of v_1. The latter works when the ratings obey some distribution over the sample space of users and the existing graph. To identify a task-specific connectivity for the incoming nodes, we merge data-driven solutions with statistical models.

More specifically, given a fixed graph $\mathcal{G}_0$ and a training set of incoming nodes $\mathcal{T} = \{(v_n, x_n, \mathbf{a}_n, \mathbf{b}_n)\}_n$, we infer the attachment probabilities $\mathbf{p}$ and weights $\mathbf{w}$ in an empirical risk minimization fashion. Each element in $\mathcal{T}$ comprises an incoming node v_n, its signal x_n, the attachment vector $\mathbf{a}_n$, and its binary form $\mathbf{b}_n$. We define a task-specific loss $\mathsf{f}_{\mathcal{T}}(\mathbf{p}, \mathbf{w}, \mathbf{a}_n, \mathbf{x}_n)$ measuring the incoming nodes performance. E.g., in collaborative filtering with cold starters, we build a user-user graph $\mathcal{G}_0$ with some existing users and treat some other users as cold starters with known connectivity and ratings in $\mathcal{T}$. Estimating the task-aware connectivity translates into solving the

statistical optimization problem

$$\begin{aligned} \min_{\mathbf{p},\ \mathbf{w}} \quad & \mathbb{E}\big[\mathsf{f}_{\mathcal{T}}(\mathbf{p},\mathbf{w},\mathbf{a}_n,\mathbf{x}_n)\big] + \mathsf{g}_{\mathcal{T}}(\mathbf{p},\mathbf{b}_n) + \mathsf{h}_{\mathcal{T}}(\mathbf{w},\mathbf{a}_n) \\ \text{subject to} \quad & \mathbf{p} \in [0,1]^N, \mathbf{w} \in \mathscr{W} \end{aligned} \tag{3.5}$$

where $\mathsf{g}_{\mathcal{T}}(\mathbf{p},\mathbf{b}_n)$ and $\mathsf{h}_{\mathcal{T}}(\mathbf{w},\mathbf{a}_n)$ are regularizers imposing a prior between $\mathbf{p}$ and the binary attachment $\mathbf{b}_n$, and between $\mathbf{w}$ and training attachment $\mathbf{a}_n$, respectively; and $\mathscr{W}$ is a convex set constraining the edge-weights, e.g., non-negative or finite. Upon estimating the probabilities $\mathbf{p}^*$ and weights $\mathbf{w}^*$ from (3.5), we generate realizations for $v_1 \notin \mathcal{T}$.

Problem statement. *Given graph $\mathcal{G}_0$ with signal $\mathbf{x}_0$ and a training set $\mathcal{T}$ of incoming nodes, our goal is to estimate the attachment probabilities $\mathbf{p}^\star$ and weights $\mathbf{w}^\star$ of a preferential attachment model w.r.t. a task-specific cost function $f_{\mathcal{T}}(\cdot)$ by solving problem* (3.5).

We will particularize the cost function in (3.5) to the signal interpolation error at the incoming node and to the graph signal smoothness [6]. Since such problems are in general jointly non-convex in $\mathbf{p}$ and $\mathbf{w}$, we develop an alternating projected-gradient descent and discuss its marginal convexity and convergence (Section 3.4). And since each connectivity realization perturbs the graph from its nominal form, we conduct a statistical perturbation analysis [81, 82] to show the effects of the attachment model on the nominal spectrum (Section 3.5).

3.4. TASK-AWARE CONNECTIVITY LEARNING

In this section, we consider first the task of signal reconstruction at the incoming node through percolation via graph filtering [108]. Graph filters facilitate data processing at each node locally through a combination of successive shift operations and compare well with alternatives in these problems [5]. Second, we consider the task of estimating a topology such that the signal is smooth on the expanded graph.

3.4.1. SIGNAL INTERPOLATION

Consider the graph signal $\mathbf{x}_1 = [\mathbf{x}_0^\top, 0]^\top$ for $\mathcal{G}_1$, where zero is the unknown signal at v_1. The output $\mathbf{y_1}$ of an order K graph filter is

$$\mathbf{y}_1 = \mathbf{H}(\mathbf{A}_1)\mathbf{x}_1 = \sum_{k=1}^{K} h_k \mathbf{A}_1^k \mathbf{x}_1 \tag{3.6}$$

where $\mathbf{h} = [h_1, \ldots, h_K]^\top$ are the filter coefficients and $\mathbf{H}(\mathbf{A}_1) = \sum_{k=1}^{K} h_k \mathbf{A}_1^K$ is the filtering matrix. The filter order K implies that nodes up to K-hops away contribute to the interpolated signal of v_1. Also, the direct term $k = 0$ is ignored in (3.6) because it does not contribute to the output at v_1. Given the percolated signal $[\mathbf{y}_+]_{N+1}$ at node v_1 is random, the following proposition quantifies the signal reconstruction MSE as a function of the model parameters $\mathbf{p}$ and $\mathbf{w}$.

Proposition 1 *Given graph $\mathcal{G}_0 = \{\mathcal{V}_0, \mathcal{E}_0\}$ with adjacency matrix $\mathbf{A}$ and signal $\mathbf{x}_0$, let matrix $\mathbf{A}_x = [\mathbf{x}_0, \ldots, \mathbf{A}^{K-1}\mathbf{x}_0]$ contain the first $K-1$ shifted versions of $\mathbf{x}_0$. Let the incoming node v_1 attach to $\mathcal{G}_0$ with probability vector $\mathbf{p}$ and edge weight vector $\mathbf{w}$, forming graph $\mathcal{G}_1$ with the expanded adjacency matrix $\mathbf{A}_1$ [cf.(3.1)]. The MSE of the interpolated signal $[\mathbf{y}_1]_{N+1}$ at node v_1 by an order K graph filter $\mathbf{H}(\mathbf{A}_1)$ [cf.(3.6)] is approximately*

$$\mathsf{MSE}(\mathbf{p}, \mathbf{w}) \approx ((\mathbf{w} \circ \mathbf{p})^\top \mathbf{A}_x \mathbf{h} - x_1)^2 + \mathbf{h}^\top \mathbf{A}_x^\top \mathbf{\Sigma}_1 \mathbf{A}_x \mathbf{h} \tag{3.7}$$

where $\mathbf{h} = [h_1, \ldots, h_K]^\top$ are the filter coefficients and x_1 is the true signal at v_1.

Proof. See Appendix A.1. □

Proposition 1 provides insights on the role of $\mathbf{p}$ and $\mathbf{w}$ on the signal interpolation MSE. The first term on the r.h.s. of (3.7) captures the model bias w.r.t. the true signal x_1. Essentially, the model output is the dot product between the filter output $\mathbf{A}_x\mathbf{h}$ with the expected attachment vector $\mathbf{w} \circ \mathbf{p}$. Minimizing the bias implies selecting a pair ($\mathbf{p}$, $\mathbf{w}$) that combines the signal at each $v \in \mathcal{V}_0$ to make a prediction for x_1 accurately. The second term $\mathbf{h}^\top \mathbf{A}_x^\top \mathbf{\Sigma}_1 \mathbf{A}_x \mathbf{h} = \|\mathbf{A}_x\mathbf{h}\|^2_{\mathbf{\Sigma}_1}$ measures the percolated signal norm w.r.t. the uncertainty of the new connections, which is also the prediction variance. Minimizing this term might give solutions such as $\mathbf{p} = \mathbf{1}_N$ where incoming nodes connect to all $v \in \mathcal{V}_0$ and $\mathbf{p} = \mathbf{0}_N$ which prevents any connections. So, regularizers are needed for each variable. In the MSE expression, we remark that the Kth shift $\mathbf{A}^K\mathbf{x}_1$ does not appear in (3.7) because of the structure of matrix $\mathbf{A}_1$ in (3.1). We also remark that the MSE is (3.7) is only an approximation because for filter order $K \geq 3$ the MSE expression becomes intractable due to the higher order moments of $\mathbf{a}_1$. Instead, if the filter order is smaller, expression (3.7) holds also with equality. The MSE in (3.7) holds also with equality for any order K when the expanded graph has directed edges landing at v_1.

Corollary 1 *If each incoming node v_1 forms directed edges leaving from the nodes in $\mathcal{G}_0$ and landing on v_1, the MSE in (3.7) holds with equality for any filter of order K.*

Proof. See Appendix A.1. □

Applications with directed links on incoming nodes include collaborative filtering [10] and a new user in a social network interacting with the existing ones.

Regularizers. The MSE plays the role of $\mathsf{f}_{\mathcal{T}}(\cdot)$ in (3.5). Functions $\mathsf{g}_{\mathcal{T}}(\cdot)$ and $\mathsf{h}_{\mathcal{T}}(\cdot)$ regularize the problem with priors on $\mathbf{p}$ and $\mathbf{w}$, respectively. While there are several choices for the latter, we focus on the following two.

- For the probability attachment $\mathbf{p}$, we consider

$$\mathsf{g}_{\mathcal{T}}(\mathbf{p}, \mathbf{b}_n) = \mu_p \sum_{n=1}^{|\mathcal{T}|} ||\mathbf{p} - \mathbf{b}_n||_q^q \tag{3.8}$$

  where $q \in \{1, 2\}$ and $\mu_p > 0$ is a scalar. For $q = 1$, (3.8) enforces sparsity on the attachment probabilities $\mathbf{p}$, i.e., the incoming node will connect only with a few of the nodes in $\mathcal{V}_0$. This is intuitive as graphs are sparse. However, if only a few entries in $\mathbf{p}$ are nonzero, this may lead to no connections. Using $q = 2$ may overcome this as it allows v_1 to connect in expectation to any other node but with a small probability.

- Likewise, for the weights $\mathbf{w}$ we consider

$$\mathsf{h}_{\mathscr{T}}(\mathbf{w}) = \mu_w \sum_{n=1}^{|\mathscr{T}|} ||\mathbf{w} - \mathbf{a}_n||_q^q \tag{3.9}$$

  where $\mu_w > 0$. Imposing sparsity on $\mathbf{w}$ results in zero weights for many edges. This implies even if the attachment probability is one, it may incur a zero edge weight. So, we prefer a two-norm penalty.

Alternatively, another approach is to consider a joint regularizer $\mathsf{g}_{\mathscr{T}}(\mathbf{p},\mathbf{w}) = ||\mathbf{w}\circ\mathbf{p} - \mathbf{a}_n||_q^q$. However, this might limit our control over the connectivity and the edge weights.[3] We may also consider $\mathbf{w}$ to be a random variable drawn from a normal distribution $\mathscr{N}(\boldsymbol{\mu}_w, \boldsymbol{\Sigma}_w)$. In this case, we need priors for the mean $\boldsymbol{\mu}_w$ and covariance matrix $\boldsymbol{\Sigma}_w$. The proposed approach is modular to such choices and we leave their evaluation to interested readers.

Optimization problem. With this in place, we can formulate problem (3.5) as

$$\begin{aligned} \min_{\mathbf{p},\ \mathbf{w}} \quad & \mathsf{C}_I(\mathbf{p},\mathbf{w}) = \mathsf{MSE}_{\mathscr{T}}(\mathbf{p},\mathbf{w}) + \sum_{n=1}^{|\mathscr{T}|} \left(\mu_p ||\mathbf{p} - \mathbf{b}_n||_q^q + \mu_w ||\mathbf{w} - \mathbf{a}_n||_q^q\right) \\ \text{subject to} \quad & \mathbf{p} \in [0,1]^N,\ \mathbf{w} \in \mathscr{W},\ q \in \{1,2\} \end{aligned} \tag{3.10}$$

where $\mathsf{MSE}_{\mathscr{T}}(\mathbf{p},\mathbf{w})$ [cf. (3.7)] is the empirical MSE over the training set $\mathscr{T}$. Problem (3.10) is non-convex in $\mathbf{w}$ and $\mathbf{p}$, but it is marginally convex in $\mathbf{w}$ and not always in $\mathbf{p}$ due to the variance term in (3.7). We solve (3.10) with alternating projected gradient descent. Algorithm 1 summarizes the main steps. The gradient of $\mathsf{C}_I(\mathbf{p},\mathbf{w})$ w.r.t. $\mathbf{p}$ and $\mathbf{w}$ for $q = 2$ are respectively

$$\begin{aligned} \nabla_p \mathsf{C}_I(\mathbf{p},\mathbf{w}) = 2 \sum_{n=1}^{|\mathscr{T}|} ((\mathbf{w}\circ\mathbf{p})^\top \mathbf{A}_x\mathbf{h} - x_n)(\mathbf{w}\circ\mathbf{A}_x\mathbf{h}) + |\mathscr{T}|(\mathbf{A}_x\mathbf{h})^{\circ 2} \circ (\mathbf{w}^{\circ 2}) \circ (\mathbf{1} - 2\mathbf{p}) \\ + 2\mu_p \sum_{n=1}^{|\mathscr{T}|} (\mathbf{p} - \mathbf{b}_n), \end{aligned} \tag{3.11}$$

$$\begin{aligned} \nabla_w \mathsf{C}_I(\mathbf{p},\mathbf{w}) = 2 \sum_{n=1}^{|\mathscr{T}|} ((\mathbf{w}\circ\mathbf{p})^\top \mathbf{A}_x\mathbf{h} - x_n)(\mathbf{p}\circ\mathbf{A}_x\mathbf{h}) + 2|\mathscr{T}|(\mathbf{A}_x\mathbf{h})^{\circ 2} \circ \mathbf{w} \circ \mathbf{p} \circ (\mathbf{1} - \mathbf{p}) \\ + 2\mu_w \sum_{n=1}^{|\mathscr{T}|} (\mathbf{w} - \mathbf{a}_n). \end{aligned} \tag{3.12}$$

Instead, for $q = 1$, we replace terms $2\mu_p(\mathbf{p} - \mathbf{b}_n)$ and $2\mu_w(\mathbf{w} - \mathbf{a}_n)$ in the above gradient expressions with $\text{sign}(\mathbf{p} - \mathbf{b}_n)$ and $\text{sign}(\mathbf{w} - \mathbf{a}_n)$ respectively, where $\text{sign}(x) = 1$ for $x > 0$, -1 for $x < 0$, and zero otherwise. To select an appropriate μ_p and μ_w, one can perform cross validation over a range of candidate values. The complexity of the

[3] Imposing joint sparsity, we have $w_i p_i = 0$ for some $v_{0,i} \in \mathscr{V}_0$. If $p_i \approx 1$, i.e., the incoming node connects to $v_{0,i}$ with a high probability, w_i would have to be zero, which will make the connection meaningless.

Algorithm 1 Alternating projected gradient descent for problems (3.10) and (3.15)

Input: Graph $\mathcal{G}_0$, training set $\mathcal{T}$, graph signal $\mathbf{x}_0$, adjacency matrix $\mathbf{A}$, number of iterations U, cost $\mathsf{C} \in \{\mathsf{C}_I, \mathsf{C}_S\}$, learning rates η_p, η_w.
Initialization: $\mathbf{p} = \mathbf{p}^0$, $\mathbf{w} = \mathbf{w}^0$ at random, $u = 0$.
for $u \leq U$ **do**
  $\mathbf{p}$ gradient update: $\tilde{\mathbf{p}}^{u+1} = \mathbf{p}^u - \eta_p \nabla_{\mathbf{p}} \mathsf{C}(\mathbf{p}^u, \mathbf{w}^u)$;
  Projection: $\mathbf{p}^{u+1} = \prod_{[0,1]^N} (\tilde{\mathbf{p}}^{u+1})$;
  $\mathbf{w}$ gradient update: $\tilde{\mathbf{w}}^{u+1} = \mathbf{w}^u - \eta_w \nabla_w \mathsf{C}(\mathbf{p}^{u+1}, \mathbf{w}^u)$;
  Projection: $\mathbf{w}^{u+1} = \prod_{\mathscr{W}} (\tilde{\mathbf{w}}^{u+1})$;
end for

algorithm is of the order $\mathcal{O}(KMU + UN)$, where K is the filter order, M the number of edges in the $\mathcal{G}_0$, N the number of existing nodes, and U the number of update steps in each of the variables. The complexity $\mathcal{O}(KMU)$ is due to the term $\mathbf{A}_x\mathbf{h}$, which incurs a complexity of the output of an order K FIR graph filter, equal to $\mathcal{O}(KM)$. The term $\mathcal{O}(UN)$ is due the projection operation over N elements.

While we can use Algorithm 1 to solve the general non-convex case of problem (3.10), the following corollary provides a sufficient condition for problem (3.10) to be marginally convex also in $\mathbf{p}$.

Corollary 2 *Problem* (3.10) *is marginally convex in* $\mathbf{p}$ *if the regularization weight* μ_p *satisfies*

$$\mu_p \geq w_h^2 \max_i ([\mathbf{A}_x\mathbf{h}]_i)^2 - ||\mathbf{w} \circ \mathbf{A}_x\mathbf{h}||_2^2. \tag{3.13}$$

Proof. See Appendix A.3. □

While ensuring convexity, hence a guarantee for a minima, condition (3.13) may lead to an optimum that is worse than the local optima of its non-convex counterpart. This is because of the greater focus on the training attachment patterns than on the cost. When $\mathbf{A}_x$, $\mathbf{h}$ and w_h are known, we can evaluate the r.h.s. of equation (3.13) and set μ_p so that we avoid this condition. We shall corroborate this in Section 3.6.1.

3.4.2. SIGNAL SMOOTHNESS

We now learn the connectivity of the incoming nodes such that expanded graph signal is smooth. The smoothness of the graph signal $\mathbf{x}_0$ w.r.t. $\mathcal{G}_0$ is $\mathsf{QV}(\mathbf{x}) = \mathbf{x}_0^\top \mathbf{L}_0 \mathbf{x}_0$, where a lower value implies connected nodes have similar signals and vice-versa. Upon attachment of v_1 with signal x_1, the smoothness of the new graph signal $\mathbf{x}_1 = [\mathbf{x}_0^\top, x_1]^\top$ w.r.t. the expanded graph Laplacian $\mathbf{L}_1$ as a function of the connectivity vector $\mathbf{a}_1$ is

$$\mathsf{QV}(\mathbf{a}_1) = \mathbf{x}_0^\top (\mathbf{L}_0 + \mathrm{diag}(\mathbf{a}_1))\mathbf{x}_0 - 2x_1\mathbf{x}_0^\top \mathbf{a}_1 + x_1^2 \mathbf{a}_1^\top \mathbf{1} = \mathbf{a}_1^\top \hat{\mathbf{x}} + \mathsf{QV}(\mathbf{x}_0)$$

where $\hat{\mathbf{x}} = \mathbf{x}_0 \circ \mathbf{x}_0 - 2x_1\mathbf{x}_0 + x_1^2 \mathbf{1}_N$. I.e., the smoothness of the expanded graph signal is linked to the connectivity of the incoming node. We use this relationship to learn a connectivity model that ensures the expanded graph signal is smooth.

Let $\mathbf{a}_1$ be the true connectivity of v_1 and $\mathbf{a}$ be the connectivity pattern obeying the model. We are interested in the expected squared smoothness error between the model smoothness and the true smoothness, i.e., $\mathbb{E}\big[\mathsf{QV}(\mathbf{a}) - \mathsf{QV}(\mathbf{a}_1)\big]^2$. The following proposition quantifies the latter.

Proposition 2 *Let $\mathcal{G}_0 = (\mathcal{V}_0, \mathcal{E}_0)$ be a graph with Laplacian $\mathbf{L}_0$ and signal $\mathbf{x}_0$. Let an incoming node v_1 with signal x_1 attach to $\mathcal{G}_0$ forming graph $\mathcal{G}_1 = (\mathcal{V}_1, \mathcal{E}_1)$ with attachment probability $\mathbf{p}$, weight $\mathbf{w}$, and covariance matrix $\mathbf{\Sigma}_1$ [cf. (3.3)]. Let $\mathbf{a}_1$ be the true attachment. The expected squared smoothness error (ESSE) for signal $\mathbf{x}_1 = [\mathbf{x}_0, x_1]^\top$ is*

$$\mathit{ESSE}(\mathbf{p}, \mathbf{w}) = \hat{\mathbf{x}}^\top \mathbf{\Sigma}_1 \hat{\mathbf{x}} + \hat{\mathbf{x}}^\top (\mathbf{w} \circ \mathbf{p}) \big((\mathbf{w} \circ \mathbf{p})^\top \hat{\mathbf{x}} - 2\mathbf{a}_1^\top \hat{\mathbf{x}} \big) + \hat{\mathbf{x}}^\top \mathbf{a}_1 \mathbf{a}_1^\top \hat{\mathbf{x}} \tag{3.14}$$

where $\hat{\mathbf{x}} = (\mathbf{x}_0 - x_1 \mathbf{1}_N)^{\circ 2}$.

Proof. See Appendix A.4. □

Result (3.14) shows the relationship between the attachment model, the existing graph signal, and the signal at the incoming node w.r.t. the overall smoothness. The first term is the quadratic norm of $\hat{\mathbf{x}}$ w.r.t. the covariance matrix $\mathbf{\Sigma}_1$. It contributes to a lower ESSE when the variance of the attachment is low at nodes with a high signal difference. The second term is the alignment between the expected attachment pattern $\mathbf{w} \circ \mathbf{p}$ and the squared difference signal $\hat{\mathbf{x}}$; the ESSE reduces when this is smaller than twice the alignment between the true attachment and the difference signal. Thus, the ESSE reduces when $\hat{\mathbf{x}}^\top (\mathbf{w} \circ \mathbf{p})$ is small.

Optimization problem. The ESSE plays the role of $\mathsf{f}_{\mathscr{T}}(\cdot)$ in (3.5) as the MSE did in problem (3.10). Differently though from (3.10), the ESSE captures the interaction between $\mathbf{p}$ and the true connectivity $\mathbf{a}_1$ in the second term in (3.14). Thus, we drop the regularizer $\mathsf{g}_{\mathscr{T}}(\cdot)$ on $\mathbf{p}$. Particularizing then problem (3.5) w.r.t. the smoothness cost, we get

$$\begin{aligned} \min_{\mathbf{p},\ \mathbf{w}} \quad & \mathsf{C}_S(\mathbf{p}, \mathbf{w}) = \mathsf{ESSE}_{\mathscr{T}}(\mathbf{p}, \mathbf{w}) + \mu_w \sum_{n=1}^{|\mathscr{T}|} ||\mathbf{w} - \mathbf{a}_n||_q^q \\ \text{subject to} \quad & \mathbf{p} \in [0,1]^N,\ \mathbf{w} \in \mathscr{W},\ q \in \{1,2\} \end{aligned} \tag{3.15}$$

where $\mathsf{ESSE}_{\mathscr{T}}(\mathbf{p}, \mathbf{w})$ is the empirical expression of (3.14) averaged over $\mathscr{T}$. The cost $\mathsf{C}_S(\mathbf{p}, \mathbf{w})$ in (3.15) is non-convex and also marginally non-convex in $\mathbf{p}$ because of the ESSE term. We again apply the alternating projected gradient in Algorithm 1. The gradients for $q = 2$ are

$$\nabla_p \mathsf{C}_S(\mathbf{p}, \mathbf{w}) = \sum_{n=1}^{|\mathscr{T}|} \Big(\mathbf{w}^{\circ 2} \circ (\mathbf{1} - 2\mathbf{p}) \circ \hat{\mathbf{x}}_n^{\circ 2} + 2\big((\mathbf{w} \circ \mathbf{p} - \mathbf{a}_n)^\top \hat{\mathbf{x}}_n \big) \mathbf{w} \circ \hat{\mathbf{x}}_n \Big), \tag{3.16}$$

$$\nabla_w \mathsf{C}_S(\mathbf{p}, \mathbf{w}) = \sum_{n=1}^{|\mathscr{T}|} \Big(2\mathbf{w} \circ \mathbf{p} \circ (\mathbf{1} - \mathbf{p}) \circ \hat{\mathbf{x}}_n^{\circ 2} + 2\big((\mathbf{w} \circ \mathbf{p} - \mathbf{a}_n)^\top \hat{\mathbf{x}}_n \big) \mathbf{p} \circ \hat{\mathbf{x}}_n + 2\mu_w (\mathbf{w} - \mathbf{a}_n) \Big). \tag{3.17}$$

We considered Alg. 1 also for (3.10) to provide a unified approach for both problems despite problem (3.10) being also marginally non-convex. But we could also simply choose a joint stochastic gradient method. The choice of alternating descent approach is rather standard, as seen in [109, 110] but the alternating one allows us to characterize the convergence for both costs [cf. Appendix A.6].

3.4.3. CONVERGENCE

To comment on the convergence of the alternating projected gradient descent approach, we assume the following.

Assumption 1 *The Hessians of the costs* (3.10) *and* (3.15) *w.r.t. the variables* $\mathbf{w}$ *and* $\mathbf{p}$ *are upper bounded by*

$$\nabla_p^2 C(\mathbf{p},\mathbf{w}) \preceq L_p \mathbf{I}, \quad \nabla_w^2 C(\mathbf{p},\mathbf{w}) \preceq L_w \mathbf{I}. \tag{3.18}$$

where L_p and L_w are the upper bounds of the eigenvalues of the Hessian of the two loss functions w.r.t. $\mathbf{p}$ and $\mathbf{w}$, respectively. This implies that the maximum eigenvalue of the Hessian is upper-bounded for both costs w.r.t. both variables. This can be easily verified for (3.10) and (3.15).

Theorem 1 *Given costs* (3.10) *and* (3.15) *satisfy Assumption 1 and given Algorithm 1 runs with step-sizes* η_p *and* η_w*. Then, it holds that:*

1. *If the step sizes satisfy* $0 < \eta_p \leq \frac{3}{4L_p}$, $0 < \eta_w \leq \frac{3}{4L_w}$, *the cost is non-increasing over the iterations, i.e.,* $sC(\mathbf{p}^{u+1},\mathbf{w}^{u+1}) \leq C(\mathbf{p}^u,\mathbf{w}^u)$.
2. *If there exist a feasible local minimum* $(\mathbf{p}^*,\mathbf{w}^*)$ *and the step sizes satisfy* $0 < \eta_p \leq \frac{1}{2L_p}$, $0 < \eta_w \leq \frac{1}{2L_w}$, *Algorithm 1 reaches this local minimum* $(\mathbf{p}^\star,\mathbf{w}^\star)$ *with convergence rate* $\mathcal{O}(1/U)$*, where* U *is the number of iterations.*

Proof. See Appendix A.6. □

Since the choice of the local minima is arbitrary, this shows that Algorithm 1 can converge to any of the local minima. One way to deal with this is to run Algorithm 1 for multiple initializations and select the pair ($\mathbf{p}$, $\mathbf{w}$) that gives the lowest training cost. We show in Section 3.6.1 that this may not always be needed as we have seen consistently that different initializations lead to similar costs.

3.5. PERTURBATION ANALYSIS

During testing, we draw samples from the learned attachment probabilities $\mathbf{p}$ to obtain the expanded graph realizations. Such realizations are edge sampled versions of the nominal graph $\bar{\mathcal{G}}_1$ that contains all possible edges to sample. This leads to differences both in the vertex and spectral domain [111], which have an impact on the task of interest. To characterize such an impact, we look at the spectral difference between the realized and nominal graphs and link it with our cost functions.

Given $\mathbf{p}$ and $\mathbf{w}$, we have a nominal graph $\overline{\mathcal{G}}_1$ with adjacency matrix $\overline{\mathbf{A}}_1 = \begin{bmatrix} \mathbf{A} & \mathbf{w} \\ \mathbf{w}^\top & 0 \end{bmatrix}$ and Laplacian $\overline{\mathbf{L}}_1$. I.e., realization $\mathbf{A}_1$ is a probabilistic edge-sampled version of

$\overline{\mathbf{A}}_1$ where edge $(v_1, v_{0,i})$ in $\overline{\mathbf{A}}_1$ is removed with probability $(1-p_i)$. To study the difference of each realization $\mathbf{A}_1$ from its nominal version $\overline{\mathbf{A}}_1$, we analyze the nominal matrices in the spectral domain via a perturbation analysis [112]. Consider their eigendecompositions

$$\overline{\mathbf{A}}_1 = \mathbf{V}_{\bar{A}} \mathbf{\Lambda}_{\bar{A}} \mathbf{V}_{\bar{A}}^\top, \quad \overline{\mathbf{L}}_1 = \mathbf{V}_{\bar{L}} \mathbf{\Lambda}_{\bar{L}} \mathbf{V}_{\bar{L}}^\top \tag{3.19}$$

where $\mathbf{V}_{\bar{A}} = [\mathbf{v}_{\bar{A},1}, \ldots, \mathbf{v}_{\bar{A},N+1}]$, $\mathbf{\Lambda}_{\bar{A}} = \text{diag}(\lambda_{\bar{A},1}, \ldots, \lambda_{\bar{A},N+1})$ and $(\lambda_{\bar{A},i}, \mathbf{v}_{\bar{A},i})$ is the ith eigenpair. Let also $[\mathbf{v}_{\bar{A},i}]_{1:N}$ be the first N elements of $\mathbf{v}_{\bar{A},i}$ and $[\mathbf{v}_{\bar{A},i}]_j$ be the j-th entry of vector $\mathbf{v}_{\bar{A},i}$. Similarly, for $\overline{\mathbf{L}}_1$, define $\mathbf{V}_{\bar{L}} = [\mathbf{v}_{\bar{L},1}, \ldots, \mathbf{v}_{\bar{L},N+1}]$, $\mathbf{\Lambda}_L = \text{diag}(\lambda_{\bar{L},1}, \ldots, \lambda_{\bar{L},N+1})$, $[\mathbf{v}_{\bar{L},i}]_{1:N}$, and $[\mathbf{v}_{\bar{L},i}]_j$.

Assumption 2 *There exists finite positive constants c_1 and c_2 such that*

$$\left|\left|2[\mathbf{v}_{\bar{A},i}]_{N+1}[\mathbf{v}_{\bar{A},i}]_{1:N})\right|\right|_2^2 \leq c_1, \quad \left|\left|([\mathbf{v}_{\bar{L},i}]_{1:N} - [\mathbf{v}_{\bar{L},i}]_{N+1}\mathbf{1}_N)^{\circ 2}\right|\right|_2^2 \leq c_2. \tag{3.20}$$

These constants depend on eigenvectors $\mathbf{V}_{\bar{A}}$ and $\mathbf{V}_{\bar{L}}$ which are deterministic as $\overline{\mathbf{A}}_1$ and $\overline{\mathbf{L}}_1$ are in turn fixed. So, we can always evaluate c_1 and c_2. Generating realizations $\mathbf{A}_1$ and $\mathbf{L}_1$ leads to the perturbations

$$\mathbf{\Delta A}_1 = \mathbf{A}_1 - \overline{\mathbf{A}}_1, \quad \mathbf{\Delta L}_1 = \mathbf{L}_1 - \overline{\mathbf{L}}_1. \tag{3.21}$$

We then assume the following to study the spectral effect of the perturbation.

Assumption 3 *The perturbation is small in nature, i.e.,*

$$||\mathbf{\Delta A}_1||_F \ll ||\overline{\mathbf{A}}_1]||_F, \quad ||\mathbf{\Delta L}_1||_F \ll ||\overline{\mathbf{L}}_1]||_F \tag{3.22}$$

where $||\cdot||_F$ denotes the Frobenius norm.

This is a standard assumption for robustness in the graph spectral domain [111, 112]. For it to hold, $\mathbf{p}$ should have high values for most nodes in $\mathcal{V}_0$ or be sparse, which can be set during training. Then, the spectral deviations in the ith eigenvalue $\lambda_{\bar{A},i}$ of the nominal adjacency and $\lambda_{\bar{L},i}$ of the nominal Laplacian due to the sampling perturbation are given by

$$\mathbf{\Delta}\lambda_{\bar{A},i} = \mathbf{v}_{\bar{A},i}^\top \mathbf{\Delta A}_1 \mathbf{v}_{\bar{A},i}, \quad \mathbf{\Delta}\lambda_{\bar{L},i} = \mathbf{v}_{\bar{L},i}^\top \mathbf{\Delta L}_1 \mathbf{v}_{\bar{L},i}. \tag{3.23}$$

I.e., they are dictated to how aligned are the respective eigenvectors to the perturbed graph. With this in place, we claim the following.

Proposition 3 *Given $\overline{\mathbf{A}}_1$, $\overline{\mathbf{L}}_1$ and their eigendecompositions in (3.19). Let Assumption 2 hold with constants c_1 and c_2. Let the covariance matrix of attachment be $\mathbf{\Sigma}_1$ and vector $\bar{\mathbf{p}}$ be such that $[\bar{\mathbf{p}}]_i = 1/\sqrt{p_i}$, if $p_i \neq 0$, and zero otherwise. When v_1 joins, the expected squared deviation in the ith eigenvalues of $\overline{\mathbf{A}}_1$ and $\overline{\mathbf{L}}_1$ are respectively upper bounded as*

$$\mathbb{E}[\mathbf{\Delta}^2 \gamma_i] \leq c_1 \bar{\mathbf{p}}^\top \mathbf{\Sigma}_1 \bar{\mathbf{p}}, \tag{3.24}$$

$$\mathbb{E}[\mathbf{\Delta}^2 \pi_i] \leq c_2 \bar{\mathbf{p}}^\top \mathbf{\Sigma}_1 \bar{\mathbf{p}}. \tag{3.25}$$

Proof. See Appendix A.7. □

Proposition 3 shows that both bounds have the common term $\tilde{\mathbf{p}}^\top \boldsymbol{\Sigma}_1 \tilde{\mathbf{p}}$, which is similar to the variance term in the MSE, i.e., $\frac{1}{2}\mathbf{h}^\top \mathbf{A}_x^\top \boldsymbol{\Sigma}_1 \mathbf{A}_x \mathbf{h}$ [cf. (3.7)] and the ESSE term of $\hat{\mathbf{x}}^\top \boldsymbol{\Sigma}_1 \hat{\mathbf{x}}$ [cf. (3.14)]. If $\mathbf{A}_x \mathbf{h} = \sqrt{2c_1}\tilde{\mathbf{p}}$ and $\hat{\mathbf{x}} = \sqrt{c_2}\tilde{\mathbf{p}}$, we have equality in (3.24) and (3.25), respectively. Since the MSE and the ESSE already contain similar terms, minimizing these over $\mathbf{w}$ and $\mathbf{p}$ helps minimizing the expected squared eigenvalue perturbation. So, the optimization problems (3.10) and (3.15) account implicitly for minimizing this measure of the spectral perturbation. In the next section, we contrast the perturbation achieved with other attachment methods.

3.6. NUMERICAL RESULTS

In this section, we evaluate our approach and compare it with related methods with synthetic and real data. For comparison, we consider three attachment rules:

1. **Uniform attachment.** The node attaches uniformly, i.e., $\mathbf{p}_u = \frac{1}{N}\mathbf{1}$.

2. **Preferential attachment.** The node v_1 attaches to $v_{0,i}$ with probability $p_i \propto d_i$ with d_i the degree of $v_{0,i}$, and $\mathbf{p}_d = \frac{\mathbf{d}}{\mathbf{1}^\top \mathbf{d}}$ where $\mathbf{d} = [d_1, \ldots, d_N]^\top$ is the degree vector.

3. **Training attachment only.** It relies only on the attachment patterns available during training to build $\mathbf{p}$ and $\mathbf{w}$, i.e., we ignore the MSE and ESSE costs in their respective cost formulations. They are given by $\mathbf{p}_g = \frac{1}{|\mathcal{T}|}\sum_{n=1}^{|\mathcal{T}|} \mathbf{b}_n$ and $\mathbf{w}_g = \frac{1}{|\mathcal{T}|}\sum_{n=1}^{|\mathcal{T}|} \mathbf{a}_n$.

The first two rules serve as baselines to assess how the proposed data-driven stochastic model compares with conventional statistical models, while the latter assesses the importance of the task-specific cost.

3.6.1. SYNTHETIC DATA

We build two synthetic graphs of $N = 100$ nodes following the Erdős Rényi (ER) and the Barabasi-Albert (BA) model. We consider the tasks of signal interpolation at an incoming node and the prediction of the ESSE for an incoming node.

Experimental setup. For each graph, we generated the existing graph signal $\mathbf{x}_0$ by combining the first 30 eigenvectors of its Laplacian matrix with weights from a normal distribution. Then, we normalized the signal to be zero mean. The edge formation probabilities for these graphs were set as $\mathbf{p}_u$ and $\mathbf{p}_d$, respectively. We select $\mathbf{w}$ to be the vector of all ones for both graphs. We use a filter of order one with $h_1 = 1$ to percolate the signal, which amounts to one shift operation.

The training set comprises 1000 data points divided into an 80-20 train-test split. The regularization weights μ_p, μ_w for problems (3.10) and (3.15) were selected via ten-fold cross validation from the set $[10^{-5}, 10^0]$. We performed $U = 3000$ iterations of alternating projected descent [cf. Alg. 1]. The learning rates η_p, η_w are 10^{-5}. We

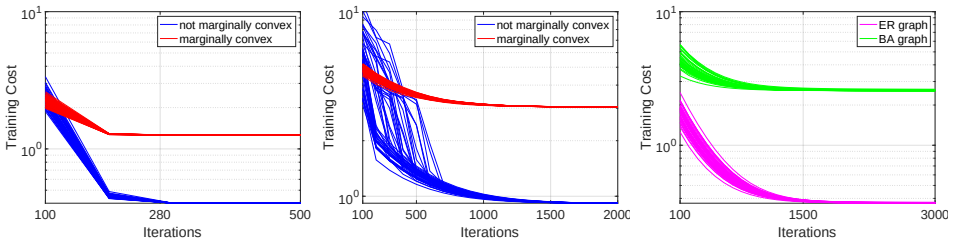


Figure 3.2.: Semilog plots of training error vs. iterations over 50 initializations each for left: ER graph with MSE; centre: BA graph with MSE; right: ER and BA graphs with ESSE. For the MSE, the blue line represents non-convexity in $\mathbf{p}$ while the red represents otherwise. For all initializations, the proposed algorithm converges to the same training cost.

Table 3.1.: MSE performance comparison between proposed, preferential and random attachment over the Erdős Rényi and Barabasi-Albert graph for training over (left two columns) both and (right two columns) one variable(s).

Graph	Erdős-Rényi			Barabasi-Albert		
Rule	Prop.	Pref.	Rand.	Prop.	Pref.	Rand.
MSE	**0.37**	0.64	0.73	**0.84**	1.72	1.35
Std Dev.	**0.04**	0.04	0.04	**0.11**	0.11	0.11
Rule	both **p**,**w**	only **p**	only **w**	both **p**,**w**	only **p**	only **w**
MSE	**0.34**	0.94	0.37	**0.84**	6.50	0.84
Std Dev.	**0.04**	0.07	0.04	**0.10**	5.55	0.10

Table 3.2.: ESSE performance comparison between proposed, preferential and random attachment over the Erdős Rényi and Barabasi-Albert graph for training over (top half) both and (bottom half) one variable(s).

Graph	Erdős-Rényi			Barabasi-Albert		
Rule	Prop.	Pref.	Rand.	Prop.	Pref.	Rand.
ESSE	**0.43**	1.83	2.20	**3.20**	14.00	10.63
Std Dev.	**0.17**	0.23	0.25	**0.81**	1.83	1.68
Rule	both **p**,**w**	only **p**	only **w**	both **p**,**w**	only **p**	only **w**
ESSE	**0.43**	2.20	0.56	**3.20**	7.48.	4.86
Std Dev.	**0.17**	0.23	0.25	**0.81**	1.83	1.68

averaged the performance over 100 realizations and 50 train-test splits to get the error for each test node.

Algorithm convergence. We solved (3.10) for each graph under two scenarios, one where μ_p satisfies the convexity criterion (3.13), and one where it does not. Figure 3.2 shows the training costs as a function of the number of iterations for 50 random

Table 3.3.: MSE and ESSE comparison for the proposed method over the Erdős Rényi and Barabasi-Albert graph for $q = 2$ and $q = 1$.

Graph	Erdős-Rényi		Barabasi-Albert	
q	$q = 2$	$q = 1$	$q = 2$	$q = 1$
MSE	0.37	0.67	0.84	1.01
Std. dev.	0.04	0.06	0.11	0.13
ESSE	0.43	0.82	3.2	3.5
Std. dev.	0.17	0.31	0.81	1.5

Table 3.4.: Squared Eigenvalue perturbation.

MSE	Proposed	Random	Preferential
Erdős Rényi	3.2×10^{-4}	4.2×10^{-4}	6.3×10^{-4}
Barabasi-Albert	1.05×10^{-4}	4.1×10^{-4}	6.4×10^{-4}
ESSE	Proposed	Random	Preferential
Erdős Rényi	5.5×10^{-4}	4.8×10^{-4}	4.4×10^{-4}
Barabasi-Albert	6.59×10^{-4}	3.3×10^{-4}	5.2×10^{-4}

initializations. The non-convex cost (blue) and the marginally convex cost in both variables (red) for $\mu_p = 30$ (satisfying (3.13)) converge following Algorithm 1. Most importantly, optimizing over the non-convex cost yields a lower training cost because a higher weight μ_p on the regularizer $\sum_{n=1}^{|\mathcal{T}|} ||\mathbf{p} - \mathbf{b}_n||^2$ results in $\mathbf{p}$ fitting the binary training attachments $\mathbf{b}_n$ than reducing the MSE. Figure 3.2 (right) shows the ESSE training error for both graphs. We see multiple local minima and that they all lead to the same training cost.

Mean square error. Here, we evaluate the signal interpolation performance. We choose $\mu_p = 1, \mu_w = 1$ for the ER graph and $\mu_p = 1, \mu_w = 0.1$ for the BA graph. Table 3.1 compares the different methods on the left two columns. The proposed approach outperforms the others in both settings in expectation and has a comparable standard deviation.

To investigate the effect of jointly training $\mathbf{p}$ and $\mathbf{w}$, we train for each of them separately while keeping the other fixed to the true value used for data generation. In Table 3.1 (bottom half), we find that training the weights, given the true $\mathbf{p}$ provides a performance comparable to the proposed for the ER graph and similar to that of the BA graph. On the other hand, training $\mathbf{p}$ given the true $\mathbf{w}$ degrades the performance appreciably, performing worse than $\mathbf{p}_u$ and $\mathbf{p}_d$. This is because when we train on $\mathbf{w}$, we deal with a convex function and reach a global optimum, whereas training on $\mathbf{p}$ leads to local minima, affecting the performance. However, these results show that the proposed approach is able to learn both p and w to solve the task.

Expected smoothness squared error. Now, we look at the ESSE performance in the same setting with $\mu_w = 0.1$ in Problem (3.15). Table 3.2 shows in the left two columns that the data-driven attachment outperforms the random and preferential attachment, with a lower standard deviation. Table 3.2 also shows in the right two

columns the ESSE for training with one parameter fixed, as done for the MSE. We see a similar trend as before when we train only **w** for the ER graph.

Choice of regularizer. Table 3.3 highlights the role of q in estimating the attachment model, through the MSE and ESSE for the two synthetic graphs. We observe that for $q = 2$, the MSE and ESSE are lower than for $q = 1$, which promotes a sparse **p** and **w**. For a sparse **p**, the model restricts attachment to some nodes, and for a sparse **w**, even an attachment results in zero weights, incurring a higher error in the inference.

Perturbation. We now analyse the mean squared deviation for each eigenvalue over multiple realizations. To give a graph-wide representative metric we report the average taken over all eigenvalues and compare with the *uniform* and *preferential attachment* with $\mathbf{w} = \mathbf{1}$. We focus on the effect of edge perturbation only. Table 3.4 showcases that the proposed approach achieves the lowest perturbation for both graphs while training for MSE. However, for the ESSE, the edge perturbation is higher for the ER graph due to more links being formed. In turn, more links changes affect more eigenvalues, thereby causing a higher perturbation.

3.6.2. COLLABORATIVE FILTERING

We consider the task of cold start rating prediction on the Movielens 100K data-set. This amounts to rating prediction for unknown users, i.e., we start with some existing users as nodes of a user-user graph and interpolate the rating of a new user when joining the network. We use the graph collaborative filter in [10] to percolate the ratings of the existing users and the learnt attachment to predict the rating at the cold starter.

Experimental setup. We retained all users and items with more than ten interactions, giving 943 users and 1152 items. We considered 50 existing users and build the adjacency matrix based on the Pearson correlation between their ratings. Next, for each item i we built the corresponding adjacency matrix by 1) retaining all outgoing links from users who rated that item; 2) building its 35 nearest neighbour graph following [10]. The remaining users were treated as cold starters and were divided into train (793) and test (100). We used an order five FIR graph filter obtained by optimally solving the rating prediction problem over the existing users and items [10]. For the interpolation cost (3.10), we impose an ℓ_1-norm constraint on **p** an ℓ_2-norm constraint on **w**. We applied Algorithm 1 for $U = 1000$ iterations with learning rates $\eta_p, \eta_w = 10^{-4}$. We predicted the ratings for the test users and for each item we averaged the performance over 100 connectivity realizations drawn from **p**. As a baseline, we considered the *mean prediction* which uses the mean of the item ratings in training to predict how cold start users will rate the item.

Item-specific learning. First, we learnt $(\mathbf{p}, \mathbf{w})$ for each item separately, which is preferred for personalized recommendations. We focused on three categories of items with a high, medium and a low number of training samples. For each category, report in Table 3.5 two examples. We evaluated the performance via the Root Median Square Error (RMedSE) between the predicted and the true ratings, which is more robust than the mean to outliers that are inevitably present in all stochastic approaches.

Table 3.5.: Item details. Items 1 and 48 have high, 459 and 550 have medium, 57 and 877 have low training samples.

Item	**1**	**48**	**459**	**550**	**57**	**877**
Training Samples	362	457	196	139	64	42
Test Samples	44	65	29	22	8	8

Table 3.6.: RMedSE of all approaches. In brackets we show relative performance difference in % to the proposed.

Item	Proposed	Attachment Only	Random	Preferential	Mean Prediction
1	**0.494**	0.537(+8.7)	0.5417(+9.7)	0.527(+6.7)	0.669(+35.4)
48	**0.492**	0.611(+24.2)	0.53(+7.7)	0.62(+26)	0.55(+11.8)
459	0.462	0.49(+6)	0.52(+12.5)	**0.40**(+12.5)	1.07(+131)
550	**0.512**	0.678(+32.4)	0.66(+29)	0.692(+35.2)	0.643(+25.6)
57	**0.049**	0.057(+16.3)	0.41(+736)	0.20(+308)	0.32(+553)
877	**0.049**	1.04(+5)	1.07(+8)	1.05(+6)	1.01(+1.72)
All	**0.799**	0.802(+0.38)	0.821(+2.75)	0.820(+2.63)	0.832(+4.1)

In Table 3.6 we show the relative performance difference – worse (red) or better (green) between the proposed approach and the alternatives. The proposed method outperforms the alternatives convincingly for items with high training samples (1 and 48) and does well even with low training samples. When compared with the training attachment only, it is clear that including the graph structure and ratings along with the attachment patterns is more beneficial. The training only and the preferential attachment strategies prioritise users who have rated the item as only those users have links directed outwards. The performance of our approach suggests that such attachments are not always optimal for the cold start. The poor performance of the uniform and the preferential attachment (except item 459) shows the importance of using a task-aware connectivity approach. The mean prediction performance is dependent on the quality and quantity of the available ratings. For example, it is considerably worse off for all items except for item 877, even though it has few training samples.

Learning for all items. Second, we learnt a common $(\mathbf{w}, \mathbf{p})$ across all the 1152 items in the data set. We select $\mu_p = 1$ and $\mu_w = 10^{-3}$. The results are in the last column of Table 3.6. We notice that with all the item data, even though the proposed performs the best, the performance gap reduces, which is somewhat expected as we are not personalizing recommendations. This suggests that to improve the cold start performance, we should approach each item individually following the spirit of nearest neighbour collaborative filtering. The attachment only method performs well because the attachment rule is cognizant of a diverse set of node attachments over many graphs and ratings. The proposed still does substantially well compared to alternatives.

3.6.3. BLOG NETWORK

We consider a political blog network with blogs as nodes and their political orientation (liberal vs. conservative) as signals [78]. We study how nodes attaching based on the ESSE influence the structure of the existing graph. This is because the ESSE is low when the signal varies slowly within a cluster than arbitrarily between clusters [81, 82]

Experimental setup. We extracted a connected sub-graph from the main network with $N = 600$ blogs such that this graph retains the clustering. The remaining 622 nodes are split into train (400) and test (200). The existing adjacency matrix is binary, so we set $\mathbf{w} = \mathbf{1}$, i.e., we train to minimise the ESSE only w.r.t. $\mathbf{p}$. We apply Algorithm 1 for 500 iterations with learning rates $\eta_p = 10^{-5}$ and $\eta_w = 10^{-6}$. We consider the clustering coefficient of a graph as a measure of how well it is clustered [19]. A large value implies a more clustered graph. Upon learning $\mathbf{p}$, we calculate the clustering coefficient of the expanded graph formed upon its attachment. We contrast this with the clustering coefficient with that of the true attachment.

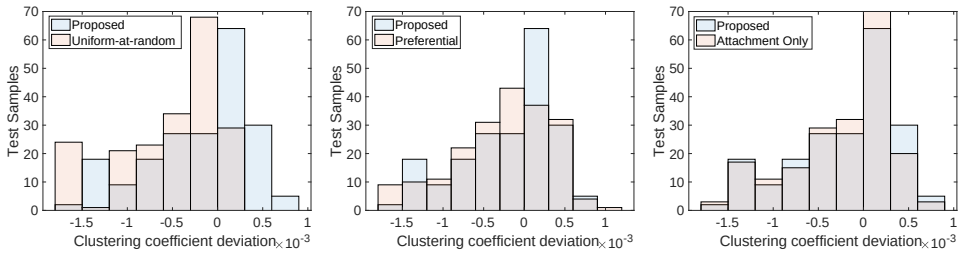


Figure 3.3.: Histogram of the clustering coefficient deviation of the proposed approach compared with the (left) *uniform*; center *preferential attachment*; right *training attachment only*. The proposed approach causes positive deviation for the most test nodes, while causing the fewest negative deviations as well.

Figure 3.3 compares the histograms of the clustering coefficient difference between the realization and the true attachment between the proposed and other approaches. A positive deviation improves upon the clustered nature of the graph, while a negative deviation reduces it. Ideally, we want more positive and fewer negative deviations. In Figure 3.3 the proposed approach outperforms the random attachment, which is likelier to make an incoming node connect to both clusters and incur a negative deviation. In Figure 3.3, the preferential attachment incurs negative deviation for more test nodes but also reports the highest positive deviation for a few nodes. In Figure 3.3 the proposed approach influences positive deviation for more test nodes in the two furthest bins and fewer negative deviations than training attachment only. By minimizing the ESSE, new nodes attach in a way that is likelier to retain/ improve upon the overall clustering for unknown nodes. Hence,

the data-driven attachment follows the true network properties if the cost function matches with the task; here, preserve its clustering structure.

3.7. CONCLUSION

We proposed an approach to learn the random connectivity model for incoming nodes by solving a signal processing task over the expanding graph. Incorporating the data-processing task to determine the attachment is beneficial compared to relying on the knowledge of previous node attachments and standard stochastic attachments. We formulated a stochastic optimization problem w.r.t. the attachment parameters for graph signal interpolation and signal smoothness. The problem is solved via an alternating projected descent approach with provable convergence to local minima. By conducting a perturbation analysis, we show that our method implicitly controls the spectral perturbation caused by such nodes.

For undirected graphs, the higher-order statistics limit the MSE analysis to be an approximation of the true one. This might be addressed by learning two graphs with incoming and outgoing directed attachments. Thus said, this method lays the foundation for approaching signal processing on expanding graphs by relying only on its stochastic connectivity model. Throughout, we consider the addition of only one node to an existing graph. To extend this approach to a continuously expanding graph, one has to generate or make available a corresponding training set. Moreover, with an increase in the dimension of the existing graph, the dimensions of $\mathbf{p}$ and $\mathbf{w}$ will also grow, which requires a treatment outside of the scope of this chapter.

4

GRAPH FILTER FOR INCOMING NODES

In Chapter 3, we inferred the attachment of incoming nodes for a specific data processing task performed through a fixed graph filter. In this chapter, we focus on designing the filters in this setting. Most existing filter design approaches are suited for fixed size graphs with the topology fully known. This gives rise to the following question: How does filter design with known topological information differ from one built without the full topological information? This is the case for incoming nodes in the cold-start scenario. Moreover, if the incoming nodes contain data, how do they influence the task over the rest of the graph? To come to an answer, we focus on designing filters for one incoming node. We formulate the filter design problem and derive solutions for the filter parameters. Different from the previous chapter, we consider signal processing tasks, namely, graph signal de-noising and semi-supervised learning over the incoming as well as the existing nodes.

The rest of this chapter is organised as follows: Section 4.1 introduces the problem and highlights our contributions; Section 4.2 describes the attachment of the incoming node forming two separate graphs, and how this leads to a two-filter bank; Section 4.3 provides the filter bank output and describes the filter design approach for two tasks: de-noising ad semi-supervised classification; Section 4.4 evaluates the proposed filter design strategy with numerical experiments; Section 4.5 concludes the chapter. All proofs are collected in Appendix B.[1]

[1]This chapter is based on the publication: Das, B., & Isufi, E. (2022, May). Graph filtering over expanding graphs. In 2022 IEEE Data Science and Learning Workshop (DSLW) (pp. 1-8). IEEE.

4.1. INTRODUCTION

Graph filters are a flexible, parametric, and localized information processing operator for network data [6, 8] with wide applicability in signal de-noising [52, 113], recommender systems [10, 65], semi-supervised learning [57, 114], and graph-based dictionary representations [115]. By relying on information exchange between neighbouring nodes, graph filters extend the convolution operation to the graph domain [114, 116]. In turn, they can account for the network data-topology coupling to solve the task at hand. However, graph filters are used only for graphs with a fixed number of nodes despite the evidence that practical graphs often grow in size [13, 14, 27]. Filtering network data in this setting is challenging not only because of the increase in graph dimension but also because we do not know how the new nodes attach to the graph. The importance of processing data over expanding graphs and the challenges arising when learning a filter on them have been recently recognized in a few works. Authors in [31] focus on semi-supervised learning with incoming nodes. First, a filter is learned to solve the task on the existing nodes, and then the filter output is used as a feature vector to predict the label of a new incoming data-point node. The work in [117] learns a graph filter-based neural network over a sequence of growing graphs, which are generated from a common graphon model [118]. However, the generated graphs are not related to each other. The works in [28, 80] perform regression tasks over graphs but rely on the attachment of the incoming nodes.

Despite showing potential, these works rely on the exact topological connectivity or available node features. In many cases, we only have the stochastic attachment model for this growing graph. Hence, deploying graph filters in this setting leads to a stochastic output, which requires a statistical approach in the filter learning phase. To fill this gap, we propose a stochastic graph filtering framework over an expanding graph without knowing the connectivity of the incoming nodes.

4.1.1. CONTRIBUTIONS OF THIS CHAPTER

Our contribution is threefold:

1. We propose a filtering model over expanding graphs that relies only on the preferential attachment information. The model comprises two parallel graph filters: one operating on a graph in which directed edges land at the incoming node; and one operating on an expanded graph in which directed edges depart from the incoming node. Such a procedure allows more flexibility to control the information inflow and outflow on the expanded graph and greater mathematical tractability compared with a filter running over a single graph.

2. We adapt the proposed approach to signal de-noising and semi-supervised learning over expanded graphs. We characterise the filter output stochastically and show the role played by the filter parameters and the attachment model.

3. We develop an empirical risk minimization framework to learn the filters. This framework is inspired by multi-kernel learning and balances the information between the two graphs via a single parameter.

Numerical results with synthetic and real data from sensor and blog networks corroborate our findings.

4.2. PROBLEM FORMULATION

Consider a graph $\mathcal{G}_0 = (\mathcal{V}_0, \mathcal{E}_0)$ with node set $\mathcal{V}_0 = \{v_{0,1}, \ldots, v_{0,N}\}$, edge set $\mathcal{E}_0$, and adjacency matrix $\mathbf{A}$. An incoming node v_1 attaches to $\mathcal{G}_0$ and forms two sets of directed edges: a set $\{(v_{0,n}, v_1)\}$ starting from v_1 and landing at the existing nodes $v_{0,n}$, whose weights are collected in vector $\mathbf{b}_1^{\mathrm{i}} \in \mathbb{R}^N$; and another set of $\{(v_1, v_{0,n})\}$ starting from the existing nodes $v_n \in \mathcal{V}_0$ and landing at v_1, whose weights are collected in vector $\mathbf{a}_1^{\mathrm{o}} \in \mathbb{R}^N$. We represent these connections with two directed graphs $\mathcal{G}_1^{\mathrm{i}} = \big(\{\mathcal{V}_0 \cup v_1\}, \{\mathcal{E}_0 \cup (v_{0,n}, v_1)\}\big)$ and $\mathcal{G}_1^{\mathrm{o}} = \big(\{\mathcal{V}_0 \cup v_1\}, \{\mathcal{E}_0 \cup (v_1, v_{0,n})\}\big)$ whose adjacency matrices

$$\mathbf{A}_1^{\mathrm{i}} = \begin{bmatrix} \mathbf{A} & \mathbf{b}_1^{\mathrm{i}} \\ \mathbf{0}^\top & 0 \end{bmatrix} \quad \text{and} \quad \mathbf{A}_1^{\mathrm{o}} = \begin{bmatrix} \mathbf{A} & \mathbf{0} \\ \mathbf{a}_1^{\mathrm{o}\top} & 0 \end{bmatrix} \tag{4.1}$$

where $\mathbf{0}$ is the all-zero vector. A conventional way to model the attachment of incoming nodes is via stochastic models [19] in which node v_1 connects to v_l with probability p_l^{i} and weight w_l^{i} in graph $\mathcal{G}_1^{\mathrm{i}}$, and probability p_i^{o} and weight w_l^{o} in graph $\mathcal{G}_1^{\mathrm{o}}$. Hence, $\mathbf{b}_1^{\mathrm{i}}$ and $\mathbf{a}_1^{\mathrm{o}}$ are random vectors with expected values $\boldsymbol{\mu}^{\mathrm{i}} = \mathbf{w}^{\mathrm{i}} \circ \mathbf{p}^{\mathrm{i}}$ and $\boldsymbol{\mu}^{\mathrm{o}} = \mathbf{w}^{\mathrm{o}} \circ \mathbf{p}^{\mathrm{o}}$, and covariance matrices $\boldsymbol{\Sigma}^{\mathrm{i}}$ and $\boldsymbol{\Sigma}^{\mathrm{o}}$, respectively. Here we define $\mathbf{w}^{\mathrm{i}} = [w_1^{\mathrm{i}}, \ldots, w_N^{\mathrm{i}}]^\top$, $\mathbf{w}^{\mathrm{o}} = [w_1^{\mathrm{o}}, \ldots, w_N^{\mathrm{o}}]^\top$ and denote by $\circ$ the Hadamard product.

While analyzing expanding graphs is an important topic, we are interested in processing data defined over the nodes of these graphs. Let then $\mathbf{x}_1 = [\mathbf{x}^\top, x_1]^\top \in \mathbb{R}^{N+1}$ be a set of signal values over nodes $\mathcal{V}_0 \cup v_1$ in which vector $\mathbf{x} = [x_1, \ldots, x_N]^\top \in \mathbb{R}^N$ collects the signals for the existing nodes $\mathcal{V}_0$ and x_1 is the signal at the incoming node v_1. Processing signal $\mathbf{x}_1$ amounts to designing graph filters that can capture its coupling w.r.t. the underlying directed graphs $\mathcal{G}_1^{\mathrm{i}}$ and $\mathcal{G}_1^{\mathrm{o}}$. To do so, we consider a filter bank of two convolutional filters [8, 114], one operating on graph $\mathcal{G}_1^{\mathrm{i}}$ and one on graph $\mathcal{G}_1^{\mathrm{o}}$. Mathematically, with $\mathbf{h}^{\mathrm{i}} = [h_0^{\mathrm{i}}, \ldots, h_K^{\mathrm{i}}]^\top$ and $\mathbf{h}^{\mathrm{o}} = [h_0^{\mathrm{o}}, \ldots, h_K^{\mathrm{o}}]^\top$ representing the vector of coefficients for filters $\mathbf{H}_1^{\mathrm{i}}(\mathbf{A}_1^{\mathrm{i}})$ and $\mathbf{H}_1^{\mathrm{o}}(\mathbf{A}_1^{\mathrm{o}})$, respectively, the filter bank output is

$$\mathbf{y}_1 = \mathbf{y}_1^{\mathrm{i}} + \mathbf{y}_1^{\mathrm{o}} := \underbrace{\sum_{k=0}^{K} h_k^{\mathrm{i}} [\mathbf{A}_1^{\mathrm{i}}]^k \mathbf{x}_1}_{\mathbf{H}_1^{\mathrm{i}}(\mathbf{A}_1^{\mathrm{i}})} + \underbrace{\sum_{k=0}^{K} h_k^{\mathrm{o}} [\mathbf{A}_1^{\mathrm{o}}]^k \mathbf{x}_1}_{\mathbf{H}_1^{\mathrm{o}}(\mathbf{A}_1^{\mathrm{o}})} \tag{4.2}$$

where without loss of generality we consider $\mathbf{y}_1 = [\mathbf{y}^\top, y_+]^\top$ and $\mathbf{y}_1^{\mathrm{i}}$ and $\mathbf{y}_1^{\mathrm{o}}$ are the outputs over graphs $\mathcal{G}_1^{\mathrm{i}}$ and $\mathcal{G}_1^{\mathrm{o}}$, respectively.

The stochastic nature of the attachment yields a random $\mathbf{y}_1$. It is notoriously challenging to compute statistical moments of powers of the adjacency matrices of increasing graphs because third or higher-order moments of the attachment pattern of the incoming node appear[2] [119]. However, because of the decoupled nature

[2]This is a challenge if we consider a single graph $\mathcal{G}_1 = (\mathcal{V}_1, \mathcal{E}_1)$ whose edge set contains both the incoming and outgoing edges at node v_1.

between the incoming and outgoing edges at node v_1, the kth powers of the adjacency matrices have the block structure

$$[\mathbf{A}_1^{\mathrm{i}}]^k = \begin{bmatrix} \mathbf{A}^k & \mathbf{A}^{k-1}\mathbf{b}_1^{\mathrm{i}} \\ \mathbf{0}^\top & 0 \end{bmatrix} \quad \text{and} \quad [\mathbf{A}_1^{\mathrm{o}}]^k = \begin{bmatrix} \mathbf{A}^k & \mathbf{0} \\ \mathbf{a}_1^{\mathrm{o}\top}\mathbf{A}^{k-1} & 0 \end{bmatrix} \tag{4.3}$$

which facilitate the stochastic analysis as we shall elaborate in Section 4.3. Our goal translates into estimating the filter coefficients $\mathbf{H}_1^{\mathrm{i}}$ and $\mathbf{h}^{\mathrm{o}}$ to solve specific learning tasks in a statistical fashion [120]. Specifically, we consider a training set $\mathcal{T} = \{(v_1, \mathbf{x}_1, \mathbf{t}_1)\}$ comprising a set of incoming nodes v_1 w.r.t. a fixed existing graph $\mathcal{G}_0$, an expanded graph signal $\mathbf{x}_1$ (e.g., noisy or partial observations), and a target output signal $\mathbf{t}_1 = [\mathbf{t}^\top, t_1]^\top$ (e.g., true signal, or class labels). Then, we learn the filter by solving

$$\min_{\mathbf{H}_1^{\mathrm{i}}, \mathbf{h}^{\mathrm{o}}} \frac{1}{2\gamma}\mathbb{E}\big[\mathrm{f}_{\mathcal{T}}(\mathbf{h}^{\mathrm{i}}, \mathbf{h}^{\mathrm{o}}, \mathbf{t}_1)\big] + \frac{1}{2\alpha}\mathrm{g}(\mathbf{H}_1^{\mathrm{i}}) + \frac{1}{2(1-\alpha)}\mathrm{j}(\mathbf{h}^{\mathrm{o}}) \tag{4.4}$$

where $\mathbb{E}[\mathrm{f}_{\mathcal{T}}(\mathbf{h}^{\mathrm{i}}, \mathbf{h}^{\mathrm{o}}, \mathbf{t}_1)]$ is the expected task-specific loss with the expectation taken w.r.t. both the graph attachment vectors $\mathbf{b}_1^{\mathrm{i}}$, $\mathbf{a}_1^{\mathrm{o}}$ and the data distribution; and $\mathrm{g}(\cdot)$, $\mathrm{j}(\cdot)$ are filter-specific regularizers (e.g., norm two of coefficient vectors) to avoid overfitting. The regularization weight $\gamma > 0$ controls the trade-off between fitting and regularization and scalar $0 < \alpha < 1$ balances the impact between the two filters inspired by multi-kernel learning [121]. Next, we shall particularize problem (4.4) to graph signal de-noising (Sec. 4.3.2) and graph-based semi-supervised learning (Sec. 4.3.3). For both settings, we shall characterize the filter output stochastically, use its first- and second-order moments in (4.4), and show the role played by the attachment models on $\mathcal{G}_1^{\mathrm{i}}$ and $\mathcal{G}_1^{\mathrm{o}}$.

4.3. FILTERING WITH INCOMING NODES

Before defining the filter bank in (4.2), we first rearrange it in a compact form, instrumental for our analysis. This form will also show the influence of the incoming node connectivity on the filter output.

4.3.1. COMPACT FORM

The goal of this section is to combine the filter coefficients $\mathbf{h} = [\mathbf{h}^{\mathrm{i},\top}, \mathbf{h}^{\mathrm{o},\top}]^\top$ and write (4.2) as $\mathbf{y}_1 = \mathbf{W}_1\mathbf{h}$, where $\mathbf{W}_1$ contains the coupling between the stochastic expanded graphs and the signal. Analyzing filter $\mathbf{H}_1^{\mathrm{i}}(\mathbf{A}_1^{\mathrm{i}})$, it is possible to write its output as

$$\mathbf{y}_1^{\mathrm{i}} = [\mathbf{x}_1, \mathbf{A}_1^{\mathrm{i}}\mathbf{x}_1, \ldots, [\mathbf{A}_1^{\mathrm{i}}]^K\mathbf{x}_1]\mathbf{h}^{\mathrm{i}}. \tag{4.5}$$

Then, leveraging the structure of the input $\mathbf{x}_1 = [\mathbf{x}^\top, x_1]^\top$ and the block-structure of $[\mathbf{A}_1^{\mathrm{i}}]^k$ in (4.3), we can write (4.5) as

$$\mathbf{y}_1^{\mathrm{i}} = \begin{bmatrix} \widehat{\mathbf{L}}_x \\ \mathbf{x}_K^\top \end{bmatrix} \mathbf{H}_1^{\mathrm{i}} = \begin{bmatrix} \mathbf{L}_x + x_1\overline{\mathbf{L}}_b \\ \mathbf{x}_K^\top \end{bmatrix} \mathbf{H}_1^{\mathrm{i}} \tag{4.6}$$

where $\widehat{\mathbf{L}}_x = \mathbf{L}_x + x_1\overline{\mathbf{L}}_b$, $\mathbf{L}_x = [\mathbf{x}, \mathbf{A}\mathbf{x}, \ldots, \mathbf{A}^K\mathbf{x}]$ and $\overline{\mathbf{L}}_b = [\mathbf{0}, \mathbf{b}^{\text{i}}_1, \ldots, \mathbf{A}^{K-1}\mathbf{b}^{\text{i}}_1]$ are $N \times (K+1)$ matrices and $\mathbf{x}_K = [x_1, 0, \ldots, 0]^\top \in \mathbb{R}^{K+1}$. Eq. (4.6) shows that the output $\mathbf{y}^{\text{i}}$ on the existing nodes nodes $\mathcal{V}_0$ is influenced by propagating their own signal $\mathbf{x}$ [cf. $\mathbf{L}_x$] and by propagating signal x_1 of v_1 w.r.t. the incoming attachments $\mathbf{b}_+$ [cf. $\overline{\mathbf{L}}_b$]. Instead, the filter output at the incoming node y^{i}_+ is just a scaled version of the input x_1 by coefficient h^{i}_0. The latter is because edges on graph $\mathcal{G}^{\text{i}}_1$ leave node v_1 and land on $\mathcal{V}_0$; hence, governing the direction of the signal propagation.

Likewise, analyzing filter $\mathbf{H}^{\text{o}}(\mathbf{A}^{\text{o}}_+)$, we can write its output as

$$\mathbf{y}^{\text{o}}_1 = [\mathbf{x}_1, \mathbf{A}^{\text{o}}_+\mathbf{x}_1, \ldots, [\mathbf{A}^{\text{o}}_+]^K\mathbf{x}_1]\mathbf{h}^{\text{o}}. \tag{4.7}$$

Leveraging again the structure of the input and that of the matrix powers $[\mathbf{A}^{\text{o}}_+]^k$ in (4.3), allows writing (4.7) as

$$\mathbf{y}^{\text{o}}_1 = \begin{bmatrix} \mathbf{M}_x \\ \widehat{\mathbf{m}}_x^\top \end{bmatrix}\mathbf{h}^{\text{o}} = \begin{bmatrix} \mathbf{M}_x \\ \mathbf{a}_1^{\text{o}\top}\overline{\mathbf{M}}_x + \mathbf{x}_K^\top \end{bmatrix}\mathbf{h}^{\text{o}} \tag{4.8}$$

where $\mathbf{M}_x = [\mathbf{x}, \mathbf{A}\mathbf{x}, \ldots, \mathbf{A}^K\mathbf{x}]$ is an $N \times (K+1)$ matrix, $\widehat{\mathbf{m}}_x = \overline{\mathbf{M}}_x^\top\mathbf{a}^{\text{o}}_1 + \mathbf{x}_K$, $\overline{\mathbf{M}}_x = [\mathbf{0}, \mathbf{x}, \ldots, \mathbf{A}^{K-1}\mathbf{x}]$ are of dimensions $(K+1) \times 1$ and $N \times (K+1)$, respectively, and $\mathbf{x}_K = [x_1, 0, \ldots, 0]^\top \in \mathbb{R}^{K+1}$. That is, the output $\mathbf{y}^{\text{o}}$ on the existing nodes $\mathcal{V}_0$ is influenced only by propagating their own signal $\mathbf{x}$. Instead, the output y^{o}_+ at node v_1 comprises: (i) the match between the attachment pattern $\mathbf{a}^{\text{o}\top}_1$ and the signal shifted over the existing graph $\mathcal{G}_0$, $\overline{\mathbf{M}}_x$; i.e., $\mathbf{a}^{\text{o}\top}_1\overline{\mathbf{M}}_x$; and (ii) a scaled version of its own signal x_1 by coefficient h^{o}_0. The output on the existing nodes $\mathcal{V}_0$ is not influenced by signal x_1 because the edges in $\mathcal{G}^{\text{o}}_1$ leave those nodes and land on v_1. The latter is also justified by the structure of matrix $\overline{\mathbf{M}}_x$; i.e., the existing signal $\mathbf{x}$ is first percolated over $\mathcal{G}_0$ and then mapped onto v_1 through its attachment pattern $\mathbf{a}_+$ in a *matched filtering* principle [122].

Bringing together (4.6) and (4.8), leads to the desired compact form

$$\mathbf{y}_1 = \mathbf{y}^{\text{i}}_1 + \mathbf{y}^{\text{o}}_1 = \mathbf{W}_1\mathbf{h} \quad \text{and} \quad \mathbf{W}_1 = \begin{bmatrix} \widehat{\mathbf{L}}_x & \mathbf{M}_x \\ \mathbf{x}_K^\top & \widehat{\mathbf{m}}_x^\top \end{bmatrix}. \tag{4.9}$$

Statistical identity. Throughout the statistical analysis of the filter output $\mathbf{y}_1$, we will deal with expectations of the form $\mathbb{E}[\mathbf{L}_x^\top\mathbf{C}\mathbf{M}_x]$ for some $N \times N$ square matrix $\mathbf{C}$. In the remainder of this section, we derive a handy formulation for it by using the compact form (4.9). For this, we will need the block-trace operator as defined next.

Definition 4.3.1 (Block trace) *Let $\mathbf{Z}$ be a block matrix comprising $N \times N$ sub-matrices $\mathbf{Z}_{i,j}$. The block trace operator takes as arguments matrix $\mathbf{Z}$ and an $N \times N$ matrix $\mathbf{Y}$ to yield a matrix $\mathbf{U} = \text{blktr}(\mathbf{Z}, \mathbf{Y})$ with (i,j) entry $U_{i,j} = \text{tr}(\mathbf{Y}\mathbf{Z}_{i,j})$ and $\text{tr}(\cdot)$ being the trace operator.*

Lemma 1 *Given an existing graph $\mathcal{G}$ with a noisy graph signal $\mathbf{x} = \mathbf{t} + \mathbf{n}$, where $\mathbf{t}$ is the true signal and $\mathbf{n}$ a Gaussian noise $\mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I})$. Consider also matrices $\mathbf{L}_x$ and $\mathbf{M}_x$ [cf. (4.6) and (4.8)], which can be expanded further as*

$$\mathbf{L}_x = \mathbf{L}(\mathbf{I}_{K+1} \otimes \mathbf{x}) \quad \text{and} \quad \mathbf{M}_x = \mathbf{M}(\mathbf{I}_{K+1} \otimes \mathbf{x}) \tag{4.10}$$

where $\mathbf{L}=[\mathbf{I},\mathbf{A},\dots,\mathbf{A}^K]$, $\mathbf{M}=[\mathbf{I},\mathbf{A},\dots,\mathbf{A}^K]$, $\mathbf{I}_N$ *is the* $N\times N$ *identity matrix, and* $\otimes$ *is the Kronecker product. Then, for any* $N\times N$ *square matrix* $\mathbf{C}$ *the following identity holds:*

$$\mathbb{E}[\mathbf{L}_x^\top \mathbf{C}\mathbf{M}_x]=\mathbf{L}_t^\top \mathbf{C}\mathbf{M}_t+\sigma^2\text{blktr}(\mathbf{L}^\top\mathbf{C}\mathbf{M},\mathbf{I}_N) \tag{4.11}$$

where $\mathbf{L}_t=\mathbf{L}_x|_{x=t}$, $\mathbf{M}_t=\mathbf{M}_x|_{x=t}$, *and* $\text{blktr}(\cdot)$ *is the block operator in Def. 4.3.1.*

Proof. See Appendix B.1. □

4.3.2. SIGNAL DENOISING

The first task we are interested in is recovering a true signal $\mathbf{t}_1$ from its noisy observations $\mathbf{x}_1$ by knowing only the stochastic attachment pattern of the incoming node. For this, we consider as cost the mean squared error $\mathbb{E}\big[\mathsf{f}_{\mathcal{T}}(\mathbf{h},\mathbf{t}_1)\big]:=\mathbb{E}[||\mathbf{W}_1\mathbf{h}-\mathbf{t}_1||^2_{\mathbf{D}}]$, where $\mathbf{D}=\text{diag}(d_1,\dots,d_{N+1})\in\{0,1\}^{N+1\times N+1}$ is a diagonal matrix with $d_n=1$ only if account for the MSE at node n and zero otherwise. The following proposition quantifies the latter.

Proposition 4 *Given a graph* $\mathcal{G}_0=(\mathcal{V}_0,\mathcal{E}_0)$ *with adjacency matrix* $\mathbf{A}$ *and an incoming node* v_1 *connecting to* $\mathcal{G}_0$ *with random attachment vectors* $\mathbf{b}^{\text{i}}$ *and* $\mathbf{a}^{\text{o}}$ *with respective means* $\boldsymbol{\mu}^{\text{i}}$, $\boldsymbol{\mu}^{\text{o}}$ *and covariance matrices* $\boldsymbol{\Sigma}^{\text{i}}$, $\boldsymbol{\Sigma}^{\text{o}}$ *[cf. (4.1)]. Consider a noisy signal* $\mathbf{x}_1=\mathbf{t}_1+\mathbf{n}_1$ *over the nodes* $\mathcal{V}_0\cup v_1$, *with* $\mathbf{x}_1=[\mathbf{x}^\top,x_1]$, $\mathbf{t}_1=[\mathbf{t}^\top,t_1]$, *and* $\mathbf{n}_1\sim\mathcal{N}(\mathbf{0},\sigma^2\mathbf{I}_{N+1})$. *Let also* $\mathbf{y}_1=\mathbf{W}_1\mathbf{h}$ $[cf.(4.9)]$ *be the filtered output. Then, the MSE of the filter output* $MSE_{\mathbf{D}}(\mathbf{h})=\mathbb{E}[||\mathbf{W}_1\mathbf{h}-\mathbf{t}_1||^2_{\mathbf{D}}]$ *computed on a set of nodes sampled by the diagonal matrix* $\mathbf{D}=\text{diag}(d_1,\dots,d_{N+1})\in\{0,1\}^{N+1\times N+1}$ *is*

$$\mathsf{MSE}_{\mathbf{D}}(\mathbf{h})=\mathbf{h}^\top\boldsymbol{\Delta}\mathbf{h}-2\mathbf{h}^\top\boldsymbol{\theta}+||\mathbf{t}_1||^2_{\mathbf{D}} \tag{4.12}$$

where $\boldsymbol{\Delta}=[\boldsymbol{\Delta}_{11},\boldsymbol{\Delta}_{12};\boldsymbol{\Delta}_{21},\boldsymbol{\Delta}_{22}]$ *is a* 2×2 *block matrix with:*

$$\begin{aligned}\boldsymbol{\Delta}_{11}=&\mathbf{L}_t^\top\mathbf{D}_N\mathbf{L}_t+\sigma^2\text{blktr}(\mathbf{L}^\top\mathbf{D}_N\mathbf{L},\mathbf{I}_N)+t_1\mathbf{L}_t^\top\mathbf{D}\overline{\mathbf{L}}_{\mu^{\text{i}}}+t_1\overline{\mathbf{L}}_{\mu^{\text{i}}}^\top\mathbf{D}_N\mathbf{L}_t+(t_1^2+\sigma^2)(\overline{\mathbf{L}}_{\mu^{\text{i}}}^\top\mathbf{D}\overline{\mathbf{L}}_{\mu^{\text{i}}}\\&+\text{blktr}(\overline{\mathbf{L}}^\top\mathbf{D}\overline{\mathbf{L}},\boldsymbol{\Sigma}^{\text{i}}))+d_{N+1}\text{diag}(t_1^2+\sigma^2,\mathbf{0}_K)\end{aligned} \tag{4.13}$$

where $\mathbf{D}=\text{diag}(d_1,\dots,d_N)$ *contains the indices of the sampled nodes in* $\mathcal{G}$, $\mathbf{L}=[\mathbf{I},\dots,\mathbf{A}^K]$, $\mathbf{L}_t=[\mathbf{t},\mathbf{A}\mathbf{t},\dots,\mathbf{A}^K\mathbf{t}]$, $\overline{\mathbf{L}}=[\mathbf{0},\mathbf{I},\mathbf{A},\dots,\mathbf{A}^{K-1}]$, $\overline{\mathbf{L}}_t=[\mathbf{0},\mathbf{t},\mathbf{A}\mathbf{t},\dots,\mathbf{A}^{K-1}\mathbf{t}]$, *and* $\overline{\mathbf{L}}_{\mu^{\text{i}}}=\overline{\mathbf{L}}_t|_{t=\mu^{\text{i}}}$.

$$\boldsymbol{\Delta}_{12}=\boldsymbol{\Delta}_{12}^\top=\mathbf{L}_t^\top\mathbf{D}\mathbf{M}_t+t_1\overline{\mathbf{L}}_{\mu^{\text{i}}}\mathbf{D}\mathbf{M}_t+\sigma^2\text{blktr}(\mathbf{L}^\top\mathbf{D}\mathbf{M},\mathbf{I}_N)+d_{N+1}(\mathbf{t}_K\boldsymbol{\mu}^{\text{o}\top}\overline{\mathbf{M}}_t+\mathbf{t}_{LM}) \tag{4.14}$$

where $\mathbf{M}=[\mathbf{I},\dots,\mathbf{A}^K]$, $\mathbf{M}_t=[\mathbf{t},\mathbf{A}\mathbf{t},\dots,\mathbf{A}^K\mathbf{t}]$, $\overline{\mathbf{M}}_t=[\mathbf{0},\mathbf{t},\mathbf{A}\mathbf{t},\dots,\mathbf{A}^{K-1}\mathbf{t}]$, $\mathbf{t}_K=[t_1,\mathbf{0}_K]$, *and* $\mathbf{t}_{LM}\in\mathbb{R}^{L+1\times M+1}$ *with* $(t_1^2+\sigma^2)$ *in location* $(1,1)$ *and zero elsewhere.*

$$\begin{aligned}\boldsymbol{\Delta}_{22}=&\mathbf{M}_x^\top\mathbf{D}\mathbf{M}_x+\sigma^2\text{blktr}(\mathbf{M}^\top\mathbf{D}_N\mathbf{M},\mathbf{I}_N)+d_{N+1}\big(\text{blktr}(\overline{\mathbf{M}}^\top\mathbf{R}^{\text{o}}\overline{\mathbf{M}},(\sigma^2\mathbf{I}+\mathbf{t}\mathbf{t}^\top))+\overline{\mathbf{M}}_t^\top\boldsymbol{\mu}^{\text{o}}\mathbf{t}_K^\top\\&+\mathbf{t}_K\boldsymbol{\mu}^{\text{o}\top}\overline{\mathbf{M}}_t+\text{diag}(t_1^2+\sigma^2,\mathbf{0}_K)\big)\end{aligned} \tag{4.15}$$

where $\overline{\mathbf{M}}=[\mathbf{0},\mathbf{I},\dots,\mathbf{A}^{K-1}]$, $\mathbf{R}^{\text{o}}=\boldsymbol{\Sigma}^{\text{o}}+\boldsymbol{\mu}^{\text{o}}\boldsymbol{\mu}^{\text{o}\top}$, *and* $\mathbf{t}_K=[t_1,\mathbf{0}_K]$. *Vector* $\boldsymbol{\theta}\in\mathbb{R}^{2(K+1)}$ *is of the form*

$$\boldsymbol{\theta}=\begin{bmatrix}(\mathbf{L}_t+t_1\overline{\mathbf{L}}_{\mu^{\text{i}}})^\top\mathbf{t}+t_1\mathbf{t}_K\\\mathbf{M}_t^\top\mathbf{t}+t_1\overline{\mathbf{M}}_t^\top\boldsymbol{\mu}^{\text{i}}+t_1\mathbf{t}_K\end{bmatrix}. \tag{4.16}$$

Proof. See Appendix B.2. □

The MSE in (4.12) is governed by the interactions between the statistics of the attachment vectors, and those of the percolated signals $\mathbf{t}$ and $\boldsymbol{\mu}^{\text{i}}$. Using it as the cost, problem (4.4) becomes

$$\min_{\mathbf{h}=[\mathbf{h}^{\text{i},\top}\mathbf{h}^{\text{o},\top}]^\top} \frac{1}{2\gamma}\mathsf{MSE}_{\mathbf{D}}(\mathbf{h}) + \frac{1}{2\alpha}||\mathbf{H}_1^{\text{i}}||_2^2 + \frac{1}{2(1-\alpha)}||\mathbf{h}^{\text{o}}||_2^2. \tag{4.17}$$

The scalar $\gamma>0$ controls how much we want to reduce the MSE over the nodes in $\mathbf{D}$; for $\gamma\to 0$ the importance of minimizing the MSE increases, while for $\gamma\to\infty$ it decreases. Instead, scalar $\alpha\in]0,1[$ controls the role of the filters $\mathbf{H}_1^{\text{i}}(\mathbf{A}_1^{\text{i}})$ and $\mathbf{H}_1^{\text{o}}(\mathbf{A}_1^{\text{o}})$ in (4.2). For $\alpha\to 0$ we prioritise more the filter over graph $\mathcal{G}_1^{\text{i}}$; i.e., leverage the information on the existing nodes $\mathcal{V}$ towards the incoming node v_1. And for $\alpha\to 1$ we prioritise the filter over graph $\mathcal{G}_1^{\text{o}}$; i.e., leverage the information on the incoming node v_1 towards the existing nodes $\mathcal{V}$. Such a formulation would work for any additive noise or attachment model as long as respective parameters are known.

Problem (4.17) is quadratic and convex only if matrix $\boldsymbol{\Delta}$ is positive semi-definite (PSD). However, proving the latter is challenging because of the structure of this matrix; hence, we can find local minima via descent algorithms [123]. But since we estimate $\boldsymbol{\Delta}$ from the training set $\mathcal{T}$, we can check if it is PSD and for a positive outcome we can find the closed-form solution for (4.17)

$$\mathbf{h}^\star = \left(\boldsymbol{\Delta}_{\mathcal{T}} + 2\gamma\boldsymbol{\Lambda}\right)^{-1}\boldsymbol{\theta}_{\mathcal{T}} \tag{4.18}$$

where matrix $\boldsymbol{\Lambda} = [1/2\alpha\mathbf{I}_{K+1}, \mathbf{0}; \mathbf{0}, 1/2(1-\alpha)\mathbf{I}_{K+1}] \in \mathbb{R}^{(2(K+1))\times(2(K+1))}$ and subscript $\mathcal{T}$ indicates that these quantities are estimated from data.

4.3.3. SEMI-SUPERVISED LEARNING

As second task, we perform inductive semi-supervised learning (SSL) [124]. As in [31, 57, 114], we use graph filters for such a task but now operating over the expanded graph. Specifically, we consider a binary classification problem with sparse label target vector $\mathbf{t}_1$ such that $[\mathbf{t}_1]_n = \pm 1$ if node $v_n \in \mathcal{V}_0 \cup v_1$ is labelled, or zero if unlabeled. As learning cost for problem (4.4), we consider the label fitting $\mathsf{MSE}_{\mathbf{D}}(\mathbf{h}) = \mathbb{E}[||\mathbf{y}_1 - \mathbf{t}_1||_{\mathbf{D}}^2] = \mathbb{E}[(\mathbf{y}_1-\mathbf{t}_1)^\top\mathbf{D}(\mathbf{y}_1-\mathbf{t}_1)]$ which is a typical convex approach for graph-based SSL with satisfactory results [57]. Then, in addition to regularizing the problem w.r.t. the ℓ_2-norm of the filter coefficients, we consider also graph-regularizers via the total variation w.r.t. both graphs $\mathcal{G}_1^{\text{i}}$ and $\mathcal{G}_1^{\text{o}}$; i.e., $\mathbb{E}[\mathsf{TV}(\mathbf{y}_1^{\text{i}})] = \mathbb{E}[||\mathbf{y}_1^{\text{i}} - \mathbf{A}_1^{\text{i}}\mathbf{y}_1^{\text{i}}||_2^2]$ and $\mathbb{E}[\mathsf{TV}(\mathbf{y}_1^{\text{o}})] = \mathbb{E}[||\mathbf{y}_1^{\text{o}} - \mathbf{A}_1^{\text{o}}\mathbf{y}_1^{\text{o}}||_2^2]$, respectively. The latter ensures that each filter output is a smooth over directed graphs, which has been validated for graph-based SSL [31, 57, 114, 124–126]. Then, Problem (4.4) becomes

$$\begin{aligned}\min_{\mathbf{h}=[\mathbf{h}^{\text{i},\top},\mathbf{h}^{\text{o},\top}]^\top} \; & \frac{1}{2\gamma}\mathsf{MSE}_{\mathbf{D}}(\mathbf{h}) + \frac{1}{2\alpha}||\mathbf{H}_1^{\text{i}}||_2^2 + \frac{1}{2(1-\alpha)}||\mathbf{h}^{\text{o}}||_2^2 \\ & + \frac{1}{2\beta}\mathbb{E}\left[\left|\left|\mathbf{y}_1^{\text{i}} - \mathbf{A}_1^{\text{i}}\mathbf{y}_1^{\text{i}}\right|\right|_2^2\right] + \frac{1}{2(1-\beta)}\mathbb{E}\left[\left|\left|\mathbf{y}_1^{\text{o}} - \mathbf{A}_1^{\text{o}}\mathbf{y}_1^{\text{o}}\right|\right|_2^2\right]\end{aligned} \tag{4.19}$$

where again $\gamma > 0$ controls the trade-off between the fitting term and the regularizer, $\alpha \in]0,1[$ controls the roles of the filter behavior over graphs $\mathcal{G}_1^{\mathrm{i}}$ and $\mathcal{G}_1^{\mathrm{o}}$ [cf. (4.17)], and $\beta \in]0,1[$ controls now the filter output smoothness w.r.t. graphs $\mathcal{G}_1^{\mathrm{i}}$ and $\mathcal{G}_1^{\mathrm{o}}$. For $\beta \to 0$, we bias filter $\mathbf{H}_1^{\mathrm{i}}(\mathbf{A}_1^{\mathrm{i}})$ to give an output $\mathbf{y}_1^{\mathrm{i}}$ that is smooth over graph $\mathcal{G}_1^{\mathrm{i}}$ and to *ignore* the behavior of filter output $\mathbf{y}_1^{\mathrm{o}}$ over graph $\mathcal{G}_1^{\mathrm{o}}$. This may be useful when the connectivity model of v_1 respects the clustering structure of $\mathcal{G}$. The opposite trend is observed for $\beta \to 1$.

The MSE in (4.19) is of the form (4.12) and encompasses both SSL cases with clean labels ($\sigma^2 = 0$) and noisy labels ($\sigma^2 > 0$). In (4.19), we also have the expected signal TV form that influences the filter behavior. The following proposition quantifies it.

Proposition 5 *Given the setting of Proposition 4 and considering* $\mathbf{x}_1 = \mathbf{t}_1$*, the TV of the filter outputs* $\mathbf{y}_1^{\mathrm{i}}$ *and* $\mathbf{y}_1^{\mathrm{o}}$ *over graphs* $\mathcal{G}_1^{\mathrm{i}}$ *and* $\mathcal{G}_1^{\mathrm{o}}$ *are respectively*

$$\mathbb{E}[\mathsf{TV}(\mathbf{y}_1^{\mathrm{i}})] = \mathbf{h}^{\mathrm{i}\top}\boldsymbol{\Psi}^{\mathrm{i}}\mathbf{h}^{\mathrm{i}} \quad \text{and} \quad \mathbb{E}[\mathsf{TV}(\mathbf{y}_1^{\mathrm{o}})] = \mathbf{h}^{\mathrm{o}\top}\boldsymbol{\Psi}^{\mathrm{o}}\mathbf{h}^{\mathrm{o}} \tag{4.20}$$

where

$$\begin{aligned}\boldsymbol{\Psi}^{\mathrm{i}} = {} & (\mathbf{L}_t + t_1\overline{\mathbf{L}}_{\mu^{\mathrm{i}}})^\top \boldsymbol{\Gamma}(\mathbf{L}_t + t_1\overline{\mathbf{L}}_{\mu^{\mathrm{i}}}) + t_1^2 \mathrm{blktr}(\overline{\mathbf{L}}^\top\boldsymbol{\Gamma}\overline{\mathbf{L}}, \boldsymbol{\Sigma}^{\mathrm{i}}) - 2\mathbf{t}_K\big(\boldsymbol{\mu}^{\mathrm{i}\top}\mathbf{L}_t + t_1\boldsymbol{\mu}^{\mathrm{i}\top}\overline{\mathbf{L}}_{\mu^{\mathrm{i}}} + t_1\mathrm{blktr}(\overline{\mathbf{L}}, \boldsymbol{\Sigma}^{\mathrm{i}})\big) \\ & - \mathbf{t}_K(\boldsymbol{\mu}^{\mathrm{i}\top}\mathbf{A}\mathbf{L}_t - t_1\boldsymbol{\mu}^{\mathrm{i}\top}\mathbf{A}\overline{\mathbf{L}}_{\mu^{\mathrm{i}}} - t_1\mathrm{blktr}(\mathbf{A}\overline{\mathbf{L}}, \boldsymbol{\Sigma}^{\mathrm{i}})) + (\mathbf{w}^\top\mathbf{p}^{\mathrm{i}} + 1)\mathrm{diag}(t_1, \mathbf{0})\end{aligned} \tag{4.21}$$

$$\begin{aligned}\boldsymbol{\Psi}^{\mathrm{o}} = {} & \mathbf{M}_t^\top(\boldsymbol{\Gamma} + \mathbf{R}^{\mathrm{o}})\mathbf{M}_t - \mathbf{M}_t^\top\mathbf{R}^{\mathrm{o}}\overline{\mathbf{M}}_t + \mathbf{M}_t^\top\boldsymbol{\mu}^{\mathrm{o}}\mathbf{t}_K^\top - \overline{\mathbf{M}}_t^\top\mathbf{R}^{\mathrm{o}}\mathbf{M}_t + \mathbf{t}_K\boldsymbol{\mu}^{\mathrm{o}\top}\mathbf{M}_t + \overline{\mathbf{M}}_t^\top\mathbf{R}^{\mathrm{o}}\overline{\mathbf{M}}_t + \overline{\mathbf{M}}_t^\top\boldsymbol{\mu}^{\mathrm{o}}\mathbf{t}_K^\top \\ & + \mathbf{t}_K\boldsymbol{\mu}^{\mathrm{o}\top}\overline{\mathbf{M}}_t + \mathbf{t}_K\mathbf{t}_K^\top\end{aligned} \tag{4.22}$$

are matrices that capture the attachment patterns and label propagation on the edges of the incoming node and $\boldsymbol{\Gamma} = (\mathbf{I} - \mathbf{A})^\top(\mathbf{I} - \mathbf{A})$.

Proof. See Appendix B.3. □

The expected TV forms depend on the attachment statistics in two ways: first, the expected attachments $\boldsymbol{\mu}^{\mathrm{i}}$ and $\boldsymbol{\mu}^{\mathrm{o}}$ control the label percolation from and towards the incoming node; second the in-attachment covariance $\boldsymbol{\Sigma}^{\mathrm{i}}$ and the out-attachment covariance $\boldsymbol{\Sigma}^{\mathrm{o}}$ influence the percolated labels through $\overline{\mathbf{L}}$ and $\mathbf{A}$ and $\mathbf{M}_t$ and $\overline{\mathbf{M}}_t$. Using then (4.20) in (4.19), we get

$$\min_{\mathbf{h}=[\mathbf{h}^{\mathrm{i}\top},\mathbf{h}^{\mathrm{o},\top}]^\top} \frac{1}{2\gamma}\mathsf{MSE}_{\mathbf{D}}(\mathbf{h}) + \mathbf{h}^\top\boldsymbol{\Lambda}_{\mathcal{T}}\mathbf{h} + \mathbf{h}^\top\boldsymbol{\Omega}_{\mathcal{T}}\mathbf{h} \tag{4.23}$$

where $\boldsymbol{\Lambda}$ is defined in (4.18) and $\boldsymbol{\Omega} = [1/2\beta\boldsymbol{\Psi}^{\mathrm{i}}, \mathbf{0}; \mathbf{0}, 1/2(1-\beta)\boldsymbol{\Psi}^{\mathrm{o}}]$ is an $(2(K+1))\times(2(K+1))$ matrix. As for (4.19), proving convexity for (4.23) is challenging but solvable with descent algorithms. And if we observe empirically that the matrix in the quadratic form of $\mathbf{h}$, $\boldsymbol{\Delta}_{\mathcal{T}} + \boldsymbol{\Lambda} + \boldsymbol{\Omega}$ is PSD, the solution of (4.23) is given by

$$\mathbf{h}^\star = \big(\boldsymbol{\Delta}_{\mathcal{T}} + 2\gamma(\boldsymbol{\Lambda} + \boldsymbol{\Omega}_{\mathcal{T}} + \boldsymbol{\Omega}_{\mathcal{T}}^\top)\big)^{-1}\boldsymbol{\theta}_{\mathcal{T}} \tag{4.24}$$

where again the subscript $\mathcal{T}$ indicates that the respective quantities are estimated from data.

4.4. NUMERICAL RESULTS

This section compares the proposed method with competing alternatives to illustrate the trade-offs inherent to graph filtering over expanding graphs with synthetic and real data. Our numerical tests have been focused to answer the following research questions:

RQ.1. How does the proposed approach compare with baselines that utilize the known attachment?

That is, we want to understand to what extent the proposed empirical learning framework compensates for the ignorance of the true connection. To answer this question, we compare with:

1. **Single filter with known connectivity ($\mathbf{KC}_1$).** This is the intuitive solution where the incoming node v_1 connects to the nodes in $\mathcal{V}_0$ forming a single graph $\mathcal{G}_1 = (\mathcal{V}_0 \cup v_1, \mathcal{E}_1)$, in which set $\mathcal{E}_1$ collects both the *known* incoming and outgoing edges w.r.t. v_1. Then, a single filter is trained on $\mathcal{G}_1$ as conventionally done by the state-of-the-art. This comparison validates the proposed scheme over the conventional strategy.

2. **Filter bank with known connectivity ($\mathbf{KC}_2$).** This is the proposed filter bank scheme in (4.2) with the *known* connectivity of node v_1. The rationale behind this choice is to factorize the filter degrees of freedom since $\mathbf{KC}_1$ employs a single filter and to highlight better the role of the topology.

RQ.2. How much does the information of the attached node contribute to the task performance over the existing graph?

We want to understand if the proposed approach exploits the incoming node signal without knowing the topology to improve the task over the existing graph instead of ignoring such information.

RQ.3. How does the proposed model compare on the incoming node w.r.t. inductive graph filtering?

Since graph filters have inductive bias capabilities [116], they can be learned on the exiting graph $\mathcal{G}_0$ and then transferred to expanded graphs without retraining. We want to understand if learning with a stochastic model is more beneficial than transference. To answer RQ.2 and RQ.3, we compare with:

3. **Inductive transference (IT).** I.e., we employ a single filter to solve the task over the existing graph and transfer it on the expanded graph under the same attachment model.

For all experiments, the incoming node attaches to the existing nodes ($\mathcal{G}_1^{\text{i}}$) uniformly at random with $\mathbf{p}^{\text{i}} = \mathbf{p}_u = \mathbf{1}_N/N$, and have edges landing at itself ($\mathcal{G}_1^{\text{o}}$) with a preferential attachment $\mathbf{p}^{\text{o}} = \mathbf{p}_d = \mathbf{d}/\mathbf{1}^\top\mathbf{d}$ where $\mathbf{D}$ is the degree vector; i.e., they are likelier to form links with nodes having a higher degree. The covariance matrices are

Table 4.1.: NMSE over all nodes and NMSE_1 of the different models for different SNRs for the Barabasi-Albert graph.

Barabasi-Albert						
	SNR 5dB		SNR 10dB		SNR 20dB	
Rule	NMSE	NMSE_+	NMSE	NMSE_+	NMSE	NMSE_+
Prop.	8×10^{-2}	0.79	0.073	0.8	7×10^{-4}	0.54
KC$_1$	7×10^{-2}	0.26	0.069	0.16	4×10^{-4}	9×10^{-4}
KC$_2$	8×10^{-2}	0.46	0.07	0.40	5×10^{-4}	0.03
IT	7×10^{-2}	3.61	0.069	3.5	7×10^{-4}	3.7

Table 4.2.: NMSE over all nodes and NMSE_1 of the different models for different SNRs for the NOAA data-set.

NOAA						
	SNR 5dB		SNR 10dB		SNR 20dB	
Rule	NMSE	NMSE_+	NMSE	NMSE_+	NMSE	NMSE_+
Prop.	0.103	0.156	0.054	0.110	9×10^{-3}	1.3×10^{-2}
KC$_1$	0.063	0.136	0.03	0.073	6×10^{-3}	1.2×10^{-2}
KC$_2$	0.103	0.09	0.054	0.07	9×10^{-3}	1.2×10^{-2}
IT	0.076	0.204	0.041	0.122	8×10^{-3}	1.4×10^{-2}

estimated from 10,000 generated samples of their respective attachment vectors. For simplicity, we set the expanded graph weights $\mathbf{w}^{\text{i}}=\mathbf{w}^{\text{o}}=w\mathbf{1}$ with w being the median of the non-zero existing edge weights in $\mathcal{G}$. We fixed the filter orders order $K=4$ and considered also a filter order of four for **KC**$_1$ and **IT**. We performed a $70-30$ train-test data split and selected parameters $\gamma\in[10^{-3},10]$, $\alpha,\beta\in]0,1[$ via five-fold cross-validation. We averaged the testing performance over 100 realizations per test node.

4.4.1. DENOISING

Following the chapter outline, we first answer the RQs for the de-noising task over a Barabasi-Albert (BA) graph model and the NOAA temperature data-set [127].

Experimental setup. For the BA model, we considered an existing graph of 100 nodes and 1000 incoming node realizations. For each realization, we generated a bandlimited graph signal by randomly mixing the first ten eigenvectors with the smallest variation of $\mathbf{A}_1\in\mathbb{R}^{101\times101}$ [114]. For the NOAA data set, we considered hourly temperature recordings over 109 stations across the continental U.S. in 2010. We built a five nearest neighbors (5k-NN) graph $\mathcal{G}_0$ of $N=100$ random stations as in [128][129]. We treated the remaining nodes as incoming, each forming 5k-NN on $\mathcal{G}_1^{\text{i}}$ and 5k-NN on $\mathcal{G}_1^{\text{o}}$. We considered 200 hours, yielding 1800 incoming data samples.

We corrupted the true signals with Gaussian noise of SNRs $\in\{5\text{dB},10\text{dB},20\text{dB}\}$. We measured the recovery performance over all existing and incoming nodes through the normalized mean squared error $\text{NMSE}=||\mathbf{y}_1-\mathbf{t}_1||_2^2/||\mathbf{t}_1||_2^2$ and we also measured

the NMSE only at the incoming node and denote it as NMSE_1.

Observations. Tables 4.1 and 4.2 reports the denoising performance on both datasets. The proposed approach compares well with the two baselines relying on the exact topology ($\mathbf{KC}_1$ and $\mathbf{KC}_2$). As regards the performance at the incoming node NMSE_1, we see that not knowing the topology leads to a worse performance. However, we see that in the NOAA dataset the gap is much smaller; a potential explanation for this is may be in the nearest-neighbor nature of the graph. Regarding then the last two research questions, we see that the proposed approach performs comparably well w.r.t. **IT** on the existing graphs but outperforms it by a margin when it comes to the performance of the incoming node (NMSE_1). Such findings show the advantages of the proposed scheme to keep a comparable performance with baselines relying on the exact topology and to improve substantially w.r.t. methods relying only on transference.

4.4.2. SEMI-SUPERVISED LEARNING

For SSL, we consider a synthetic sensor network graph from the GSP toolbox [130] and the political blog network [78].

Experimental setup. For the sensor network, the existing graph $\mathcal{G}_0$ has $N = 200$ nodes that are clustered into two classes (± 1) via spectral clustering to create the ground-truth. The training set $\mathcal{T}$ comprises 500 realizations of incoming nodes each making the same number of incoming and outgoing as the median degree of $\mathcal{G}_0$. The ground-truth label at the incoming node is assigned based on the class that has more edges with v_1. For the blog network, we considered 1222 blogs as nodes of a graph with directed edges being the hyperlinks between blogs and labels being their political orientation ($+1$ conservative vs. -1 liberal). We built a connected existing graph $\mathcal{G}_0$ of $N = 622$ blogs with a balanced number of nodes per class. The remaining 600 blogs are treated as incoming nodes.

We use only 10% of the labels in $\mathcal{G}_0$ and aim at inferring the missing labels in this graph by using also the information from the incoming node. These labels act also as the graph signal $[\mathbf{x}_1]_n = \pm 1$ for a labelled node and $[\mathbf{x}_1]_n = 0$ if unlabeled. We also considered two settings: first, all incoming nodes in the training set have labels (fully labelled), which allows identifying if the additional label contributes to the SSL task on $\mathcal{G}_0$; second, only half of the incoming nodes have labels (50% labelled), which adheres more to a real scenario where some of the incoming nodes are unlabeled. For the **IT** baseline, we solve the corresponding filters using [114], while for $\mathbf{KC}_1$ and $\mathbf{KC}_2$ we use the true connections. During training, standard SSL requires evaluating the loss at the nodes with available labels. Hence, when an incoming node has no label, we cannot account for its importance during training. Consequently, SSL models cannot predict labels when we do not know the connectivity. Thus, we measure only the performance of the existing nodes.

Observations. Tables 4.3 and 4.4 report the classification errors for the sensor and blog networks, respectively. The proposed approach achieves a comparable statistical performance with the two baselines ($\mathbf{KC}_1$ and $\mathbf{KC}_2$) that rely on the exact topology. This suggests that controlling the information in-flow and out-flow with a filter bank

Table 4.3.: Average (± std.) SSL error for the sensor network.

Error (%)	Fully labelled	50 % labelled
Prop.	4.64 (±3.43)	4.8 (±2.59)
KC$_1$	5.35 (±3.36)	5.6 (±2.6)
KC$_2$	4.62 (±3.41)	4.9 (±2.62)
IT	6.12 (±3.78)	5.3 (±2.63)

Table 4.4.: Average (± std.) SSL error for the blog network.

Error (%)	Fully labelled	50 % labelled
Prop.	2.8 (±0.4)	2.56 (±0.75)
KC$_1$	2.82 (±0.83)	2.42 (±0.59)
KC$_2$	2.8 (±0.4)	2.56 (±0.75)
IT	12.2 (±18)	6.58 (±0.59)

compensates effectively for the exact topology ignorance. The proposed approach reduces the error substantially compared to **IT**.

We also observe the models tend to perform better when 50% of the labels are present. We have identified two factors for this. First, some of the incoming nodes form misleading connections with both clusters. Hence, when their label diffuses it hampers the classification performance on the opposite cluster. Instead, when these nodes have no label they do influence the opposite class. This trend is observed also for **KC**$_1$ and **KC**$_2$, which shows that these wrong connections are present in the dataset. In the blog network, these are blogs with an unclear political position and have linked both with liberals and conservative groups [78]. Instead, in the sensor network, we do not see such a trend because nodes are better clustered. Second, this two-class classification problem has labels ±1 and we use the MSE as a criterion. Hence, the term $t_1^2 = 1$ affects the costs when the incoming node is present [cf. Prop.4,5] and does not help discriminating irrespective of the class. Thus, we conclude that when dealing with SSL classification in expanding graphs, the connectivity model plays also a central role in the performance.

4.5. CONCLUSION

We studied filtering of signals over expanding graphs by relying only on their attachment model connectivity. We used a stochastic model where incoming nodes connect to the existing graph, forming two directed graphs. A pair of graph filters, one for each graph, then process the expanded graph signal. To learn the filter parameters, we performed empirical risk minimisation for graph signal de-noising and graph semi-supervised learning. Numerical results over synthetic and real data show the proposed approach compares well with baselines relying on exact topology and outperforms the current solution relying on filter transference. However, the performance is strongly dependent on a fixed attachment model, prone to model

mismatch. Hence, potential future works may consider a joint filter and graph learning framework for expanding graphs.

5

ONLINE FILTERING OVER EXPANDING GRAPHS

In Chapters 3 and 4, we focus on filter design with one incoming node to study the effect of node attachment in more detail. However, expanding graphs are not limited to one node as they often keep growing with a sequential attachment of nodes in a streaming fashion. Thus, to process data over these nodes, the filter design should adapt to this streaming setting. Moreover, as we saw in the previous chapters, sometimes we do not know how each node will connect, which adds another layer of uncertainty and complexity. To meet these challenges, we propose filter design principles inspired by online machine learning for inference tasks at the incoming nodes. We design filters in two scenarios, the deterministic and in the stochastic attachment setting. Then, we estimate the gap involved in inference performance due to the absence of connectivity information.

The rest of this chapter is structured as follows: In Section 5.2, we elaborate on the sequentially expanding graph scenario, along with the basic formulation of online inference with graph filters. Sections 5.3 and 5.4 contain the online learning methods and their respective analysis in the deterministic and in the stochastic setting, respectively. Section 5.5 contains the numerical results, while Section 5.6 concludes the chapter. All proofs are collected in Appendix C.[1]

[1]This chapter is based on the publication: Das, B., & Isufi, E. (2024). Online Graph Filtering Over Expanding Graphs, IEEE Transactions on Signal Processing.

5.1. INTRODUCTION

Most of the filters in literature are designed out over graphs with a fixed number of nodes [7] despite graphs often growing through the addition of nodes, sometimes sequentially over time [14, 19]. An example is collaborative filtering in recommender systems where new users continuously join an existing network, e.g., a social network recommendation [131] or an abstract user-user collaborative filter network [10]. Such an expanding graph setting poses a three-fold challenge: (i) The data comes in a streaming nature, i.e., we do not have access to all the incoming nodes at once. This requires an on-the-fly filter design as batch-based solutions are no longer an alternative. (ii) The topology may evolve slowly or rapidly; hence, influencing the online filter design. (iii) The data over the incoming nodes is not guaranteed to follow a well-known distribution, thus requiring an adaptation of the filter to the task at hand. Often times, we may not even know how the incoming nodes connect to the existing graph. Typically, this happens in the absence of information for the incoming node, i.e., in pure cold-start recommendation, where we know nothing about user preferences, but we need to recommend items nevertheless [132]. The users may consume items later on, which can be used to infer their attachment but this can take time. Such challenges limit existing graph data processing methods which rely on the knowledge of the topology [8]. Another example where these challenges occur is in epidemic spreading over networks. We want to predict the future number of active cases for a city that is not yet affected but anticipates some cases shortly after. It may be difficult to obtain the underlying connections that influence the disease spread; hence, using statistical models is typically an option [13, 14]. In this scenario, filter design should account for the evolving topological model as well as for the data over it. This is possible by building upon online learning principles where the learning models are updated based on the incoming data stream [133, 134].

Existing works dealing with online learning over expanding graphs can be divided into Attachment, Feature Aware Methods, and Stochastic Methods.

Attachment and feature aware methods know the connectivity of the incoming nodes and their features. For example, the work in [28] performs online node regression over fixed-size graphs by using their connectivity pattern to generate random kernel features [135]. An extension of this is the work in [136] which considers multi-hop connectivity patterns. Works like [137], [138], and [139] track changing attachment patterns over time but for graphs with a fixed number of node, which can be relevant for a large-scale setting. Another instance of online processing on expanding graphs is the work in [80] which obtains embeddings for signals over expanding graphs. Some works such as [58] classify an incoming node by using its features and the filter trained over the existing graph. Then, there are works such as [30] that classify a stream of incoming nodes by using their features to estimate the attachment. The work in [140] classifies incoming nodes using spectral embeddings updated from the known attachment information. The kernel-based methods in this category [28, 136] rely on pre-selecting a suitable kernel that can fit the data which may be challenging to obtain. Additionally, the works in [141–144] develop distributed solutions to estimate the filter parameters locally at each node.

Differently, in this paper, we work with a centralized approach to estimate the filter, as we focus more on the expanding graph scenario. All in all, these methods concern either a graph with a fixed or a streaming number of nodes but with available attachment or feature information that may be unavailable.

Stochastic methods deal with unknown incoming node attachment and use models for it. For example, [145] uses heuristic stochastic attachment model to design graph filters only for one incoming node, while [146] learns an embedding by using a stochastic attachment to influence the propagation. In our earlier works [147], [148], we learn the attachment behaviour for inference with a fixed filter. However, this approach is limited to studying the effect of one node attaching with unknown connectivity and it assumes a pre-trained filter over the existing graph. Differently, here we consider the filter design over a stream of incoming nodes.

5.1.1. CONTRIBUTIONS OF THIS CHAPTER

The main contribution of this chapter is developing a framework for online graph filtering over a stream of incoming nodes when the topology is both known and unknown. Our contribution is threefold:

1. We develop an online filter design framework for inference over expanding graphs. This is done by casting the inference problem as a time-varying loss function over the existing topology, data, and the incoming node attachment. Subsequently, we update the filter parameters via online learning principles.

2. We adapt the online filter design problem to two scenarios: (i) the *deterministic* setting where the connectivity of each incoming node is available; (ii) the *stochastic* setting where this connectivity is unavailable. For both settings, we conduct a regret analysis to discuss the influence of the incoming node attachment and the role of the graph filter.

3. We develop an online ensemble and adaptive stochastic update where, in addition to the filter parameters, we also learn the combination parameters of the different stochastic attachment rules. This concerns the stochastic setting where a single attachment model might be insufficient. We also discuss the regret in this setting and analyze how the ensemble affects it.

5.2. PROBLEM FORMULATION

Consider a starting graph $\mathcal{G}_0 = \{\mathcal{V}_0, \mathcal{E}_0\}$ with node set $\mathcal{V}_0 = \{v_{0,1}, \ldots, v_{0,N_0}\}$ of N_0 nodes, edge set $\mathcal{E}_0$, and adjacency matrix $\mathbf{A}_0 \in \mathbb{R}^{N_0 \times N_0}$, which can be symmetric or not, depending on the type of graph (undirected or directed). Let $v_1, \ldots, v_T$ be a set of T sequentially incoming nodes where at time t, node v_t attaches to graph $\mathcal{G}_{t-1}$ forming the graph $\mathcal{G}_t = \{\mathcal{V}_t, \mathcal{E}_t\}$ with $N_t = N_0 + t$ nodes, M_t edges, and adjacency matrix $\mathbf{A}_t \in \mathbb{R}^{N_t \times N_t}$. The connectivity of v_t is represented by the attachment vector $\mathbf{a}_t = [a_1, \ldots, a_{N_{t-1}}]^\top \in \mathbb{R}^{N_{t-1}}$, where a non-zero element implies a directed edge from $v \in \mathcal{V}_{t-1}$ to v_t. This connectivity suits inference tasks at v_t, where the existing nodes influence the incoming ones. This is the case of cold-starters in graph-based

collaborative filtering [10, 65]. Here, the nodes represent existing users, the edges capture similarities among them (e.g., Pearson correlation), and a cold starter is a new node that attaches to this user-user graph. The task is to collaboratively infer the preference of the cold-starter from the existing users [66].

Depending on the availability of $\mathbf{a}_t$, we can have a deterministic attachment setting or a stochastic attachment setting. In a deterministic setting, the incoming node attachment vector $\mathbf{a}_t$ is known or it is estimated when v_t appears. This occurs in growing physical networks or in collaborative filtering where side information is used to establish the connectivity [10]. The expanded adjacency matrix $\mathbf{A}_t \in \mathbb{R}^{N_t \times N_t}$ reads as

$$\mathbf{A}_t = \begin{bmatrix} \mathbf{A}_{t-1} & \mathbf{0}_{N_{t-1}} \\ \mathbf{a}_t^\top & 0 \end{bmatrix} \tag{5.1}$$

where $\mathbf{A}_{t-1}$ is the $N_{t-1} \times N_{t-1}$ adjacency matrix and $\mathbf{0}_{N_{t-1}}$ is the all-zero vector of size (N_{t-1}). In a stochastic setting, $\mathbf{a}_t$ is unknown (at least before inference), which is the typical case in cold start collaborative filtering [148]. A new user/item enters the system and we have neither side information nor available ratings to estimate the connectivity. The attachment of v_t is modelled via stochastic models from network science [14]. Node v_t attaches to $v_i \in \mathcal{V}_{t-1}$ with probability $p_{i,t}$ forming an edge with weight $w_{i,t}$. The probability vector $\mathbf{p}_t = [p_{1,t}, \ldots, p_{N_{t-1},t}]^\top \in \mathbb{R}^{N_{t-1}}$ and the weight vector $\mathbf{w}_t = [w_{1,t}, \ldots, w_{N_{t-1},t}]^\top \in \mathbb{R}^{N_{t-1}}$ characterize the attachment and imply that $[\mathbf{a}_t]_i = w_{i,t}$ with probability $p_{i,t}$, and zero otherwise. We consider vector $\mathbf{a}_t$ be composed of independent, weighted Bernoulli random variables with respective mean and covariance matrix

$$\mathbb{E}[\mathbf{a}_t] = \mathbf{p}_t \circ \mathbf{w}_t \;;\; \mathbf{\Sigma}_t = \text{diag}(\mathbf{w}_t^{\circ 2} \circ \mathbf{p}_t \circ (\mathbf{1} - \mathbf{p}_t)) \tag{5.2}$$

where $\text{diag}(\mathbf{x})$ is a diagonal matrix with $\mathbf{x}$ comprising the diagonal elements, and $\mathbf{x}^{\circ 2} = \mathbf{x} \circ \mathbf{x}$ is the element-wise product of a vector $\mathbf{x}$ with itself. The new adjacency matrix for a realization $\mathbf{a}_t$ is the same as in (5.1). The attachment is revealed after the inference task; e.g., after a cold start user has consumed one or more items and we can estimate it.

5.2.1. FILTERING OVER EXPANDING GRAPHS

Let $\mathbf{x}_t \in \mathbb{R}^{N_t}$ be the graph signal over graph $\mathcal{G}_t$, which writes in terms of the previous signal $\mathbf{x}_{t-1} \in \mathbb{R}^{N_{t-1}}$ as $\mathbf{x}_t = [\mathbf{x}_{t-1}, x_t]^\top$ with x_t being the signal at the latest incoming node v_t. To infer x_t, we consider the temporary graph signal $\tilde{\mathbf{x}}_t = [\mathbf{x}_{t-1}, 0]^\top$ where the zero at v_t indicates that its value is unknown. To process such signals we use graph convolutional filters, which are linear and flexible tools for processing them [7]. A filter of order K acts on $\tilde{\mathbf{x}}_t$ to generate the output $\tilde{\mathbf{y}}_t$ on graph $\mathcal{G}_t$ as

$$\tilde{\mathbf{y}}_t = \sum_{k=0}^{K} h_k \mathbf{A}_t^k \tilde{\mathbf{x}}_t \tag{5.3}$$

where h_k is the weight given to the kth shift $\mathbf{A}_t^k \tilde{\mathbf{x}}_t$. Substituting the kth adjacency matrix power

$$\mathbf{A}_t^k = \begin{bmatrix} \mathbf{A}_{t-1}^k & \mathbf{0}_{N_{t-1}} \\ \mathbf{a}_t^\top \mathbf{A}_t^{k-1} & 0 \end{bmatrix} \tag{5.4}$$

into (5.3), we write the filter output as

$$\tilde{\mathbf{y}}_t = \begin{bmatrix} \sum_{k=0}^{K} h_k \mathbf{A}_{t-1}^k \mathbf{x}_t \\ \mathbf{a}_t^\top \sum_{k=1}^{K} h_k \mathbf{A}_{t-1}^{k-1} \mathbf{x}_t \end{bmatrix} \tag{5.5}$$

where we grouped w.l.o.g. the output at the incoming node v_t in the last entry. I.e.,

$$[\tilde{\mathbf{y}}_t]_{N_t} := \hat{x}_t = \mathbf{a}_t^\top \sum_{k=1}^{K} h_k \mathbf{A}_{t-1}^{k-1} \mathbf{x}_t = \mathbf{a}_t^\top \mathbf{A}_{x,t-1} \mathbf{h}. \tag{5.6}$$

Here $\mathbf{h} = [h_1 \dots, h_K]^\top \in \mathbb{R}^K$ collects the coefficients of the filter and $\mathbf{A}_{x,t-1} = [\mathbf{x}_t, \mathbf{A}_{t-1}\mathbf{x}_t, \dots, \mathbf{A}_{t-1}^{K-1}\mathbf{x}_t] \in \mathbb{R}^{N_{t-1} \times K}$ contains the higher-order shifts of $\mathbf{x}_t$. The coefficient h_0 does not play a role in the output $\hat{x}_t$, thus the zero in the N_tth position of $\tilde{\mathbf{x}}_t$ does not influence the inference task on the incoming node. In the stochastic setting, the output is random as it depends on the attachment rule. In turn, this needs a statistical approach to characterize both the filter and its output. We shall detail this in Section 5.4.

Remark 1 *The above discussion considers one incoming node at a time which is common in the streaming setting. The analysis can be extended to multiple nodes arriving at a certain time interval. Here, we consider inference tasks where the existing nodes affect the incoming streaming ones. For tasks where the influence is bidirectional, the adjacency matrix in* (5.1) *is symmetric and the analysis follows analogously. One way to do this is to build on [145], where we discuss the case for a single incoming node.*

5.2.2. ONLINE FILTER LEARNING

Our goal is to process signal $\tilde{\mathbf{x}}_t$ to make inference on the incoming nodes v_t by designing the filters in (5.3). We consider a data-driven setting where we estimate the filter parameters from a training set $\mathcal{T} = \{v_t, x_t, \mathbf{a}_t\}_{t=1:T}$ in which each datum comprises an incoming node v_t, its signal x_t, and the attachment vector $\mathbf{a}_t$. Given set $\mathcal{T}$, we find the filter parameters $\mathbf{h}$ by solving

$$\underset{\mathbf{h} \in \mathbb{R}^K}{\operatorname{argmin}} \sum_{t=1}^{T} \mathsf{f}_t(\hat{x}_t, x_t; \mathbf{h}) + \mathsf{r}(\mathbf{h}) \tag{5.7}$$

where $\mathsf{f}_t(\hat{x}_t, x_t; \mathbf{h}) := \mathsf{f}_t(\mathbf{h}, x_t)$ measures the goodness of fit between the prediction $\hat{x}_t$ and the true signal x_t and $\mathsf{r}(\mathbf{h})$ is a regularizer. For example, $\mathsf{f}_t(\cdot,\cdot)$ can be the least-squares error for regression problems such as signal denoising or interpolation; or the logistic error for classification problems such as assigning a class label to node v_t. For convex and differentiable $\mathsf{f}_t(\cdot,\cdot)$ and $\mathsf{r}(\mathbf{h})$, we can find an optimal filter that solves the batch problem over $\mathcal{T}$. However such a solution is not ideal since the incoming nodes v_t are streaming and evaluating a new batch for each v_t is computationally demanding. A batch solution also suffers in non-stationary environments where the test set distribution differs from $\mathcal{T}$. Targeting a non-stationary setting with incoming nodes, we turn to online learning to update the filter parameters on-the-fly [134].

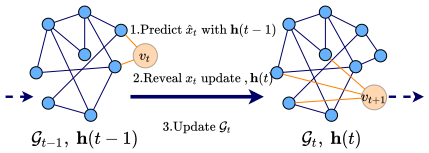


Figure 5.1.: Online filter learning process at time t through the addition of node v_t with signal x_t and the update of the filter $\mathbf{h}(t)$. (Left) node v_t attaches to the previous graph $\mathcal{G}_{t-1}$ forming $\mathcal{G}_t$; the edges in blue denote the existing edges while those in orange denote the edges formed by the incoming node; (Centre) Signal x_t is predicted, then the true value x_t is revealed, and the filter parameter $\mathbf{h}(t)$ is updated from $\mathbf{h}(t-1)$; (Right) the next node v_{t+1} attaches to $\mathcal{G}_t$.

We initialize the filter before the arrival of incoming nodes, $\mathbf{h}(0)$ by training a filter over $\mathcal{G}_0$, using $\mathbf{A}_0$ and $\mathbf{x}_0$. The training follows a regularized least square problem with ℓ_2 norm squared loss on the filter. We call this *Pre-training*. In high-level terms, the online filter update proceeds at time t as follows:

5

1. The environment reveals the node v_t and its attachment $\mathbf{a}_t$ in the deterministic setting.
2. We use the filter at time t, $\mathbf{h}(t-1)$ to infer the signal value $\hat{x}_t$ at the incoming node using (5.6).
3. The environment reveals the loss as a function of the filter $\mathbf{h}(t-1)$ and the true signal x_t as
$$\mathsf{l}_t(\mathbf{h}, x_t) = \mathsf{f}_t(\mathbf{h}, x_t) + \mathsf{r}(\mathbf{h}) \tag{5.8}$$
which is evaluated at $\mathbf{h}(t-1)$.
4. We update the filter parameters $\mathbf{h}(t)$ based on the loss and the current estimate $\mathbf{h}(t-1)$.
5. In the stochastic setting, the true attachment $\mathbf{a}_t$ is revealed.

With this in place, our problem statement reads as follows:

Problem statement. *Given the starting graph $\mathcal{G}_0 = \{\mathcal{V}_0, \mathcal{E}_0\}$, adjacency matrix $\mathbf{A}_0$, graph signal $\mathbf{x}_0$, and the training set $\mathcal{T}$, our goal is to predict online a sequence of graph filters $\{\mathbf{h}(t)\}$ w.r.t. loss functions $l_t(\mathbf{h}, x_t)$ to process signals at the incoming nodes for both the deterministic and the stochastic attachments.*

5.3. DETERMINISTIC ONLINE FILTERING

Targeting regression tasks[2], we can take $\mathsf{f}_t(\mathbf{h}, x_t)$ as the squared error and $\mathsf{r}(\mathbf{h})$ as the scaled ℓ_2-norm penalty to define the loss

$$\mathsf{l}_t(\mathbf{h}, x_t) = \frac{1}{2}(\mathbf{a}_t^\top \mathbf{A}_{x,t-1}\mathbf{h} - x_t)^2 + \mu||\mathbf{h}||_2^2 \tag{5.9}$$

[2]For classification tasks, we can consider the surrogate of the gradient of the logistic loss which is also convex and differentiable, as seen in [149].

Algorithm 2 Deterministic Online Graph Filtering (**D-OGF**)

Input. Graph $\mathcal{G}_0$, $\mathbf{A}_0$, $\mathbf{x}_0$, $\mathcal{T} = \{v_t, x_t, \mathbf{a}_t\}_{t=1:T}$.
Initialize: Pre-train $\mathbf{h}(0)$ over $\mathcal{G}_0$ using $\mathbf{A}_0$, $\mathbf{x}_0$.
for $t = 1 : T$ **do**
  Obtain v_t and true connection $\mathbf{a}_t$, update $\mathbf{A}_t$
  Predict $\hat{x}_t = \mathbf{a}_t^\top \mathbf{A}_{x,t-1}\mathbf{h}(t-1)$ (cf. (5.6))
  Reveal loss $\mathsf{l}_t(\mathbf{h}, x_t)$ (cf. (5.9))
  Update $\mathbf{h}(t)$ using (5.10)
  Update $\mathbf{x}_t$
end for

where $\mu > 0$. For the online update, we perform projected online gradient descent [133], which comprises one projected gradient descent step evaluated at $\mathbf{h}(t-1)$ as

$$\mathbf{h}(t) = \underset{\mathscr{H}}{\Pi}(\mathbf{h}(t-1) - \eta \nabla_h \mathsf{l}_t(\mathbf{h}, x_t)|_{\mathbf{h}(t-1)}) \tag{5.10}$$

with set $\mathscr{H}$ bounding the filter energy $\mathscr{E}(\mathbf{h}) = ||\mathbf{h}||_2^2$ and $\underset{\mathscr{H}}{\Pi}(\cdot)$ denotes the projection operator on $\mathscr{H}$. Here, $\eta > 0$ is the step size, and the gradient has the expression

$$\nabla_h \mathsf{l}_t(\mathbf{h}, x_t) = (\mathbf{a}_t^\top \mathbf{A}_{x,t-1}\mathbf{h} - x_t)\mathbf{A}_{x,t-1}^\top \mathbf{a}_t + 2\mu\mathbf{h}. \tag{5.11}$$

The gradient depends on $\mathbf{a}_t$ through the term $(\hat{x}_t - x_t)\mathbf{A}_{x,t-1}^\top \mathbf{a}_t$. Operation $\mathbf{A}_{x,t-1}^\top \mathbf{a}_t$ is a weighted combination of only those columns of $\mathbf{A}_{x,t-1}^\top$ where the corresponding entry of $\mathbf{a}_t$ is non-zero. In turn, each column of $\mathbf{A}_{x,t-1}^\top$ contains shifted graph signals at each node, which get scaled by the difference between the predicted and the true signal x_t, ultimately, indicating that a larger residue leads to a larger gradient magnitude. The online learner in (5.10) updates the filter parameters for every incoming node.

Algorithm 2 summarizes the learning process. The computational complexity of the online update at time t is of order $\mathcal{O}(K(M_t + M_{\max}))$, where $M_{\max}$ is the maximum number of edges formed by v_t across all t. Note that $M_{\max} \ll N_{t-1}$, i.e., the maximum number of edges formed by any incoming node is smaller than the existing number of nodes. Appendix C.4 breaks down this complexity.

Regret analysis. We analyze the deterministic online graph filtering algorithm to understand the effect of the filter updates and how the expanding graph influences it. Specifically, we conduct a regret analysis that quantifies the performance difference between the online updates and the static batch solution where all the incoming node information is available. The normalized regret w.r.t. a fixed filter $\mathbf{h}^\star$ is defined as

$$\frac{1}{T} R_T(\mathbf{h}^\star) = \frac{1}{T} \sum_{t=1}^{T} \mathsf{l}_t(\mathbf{h}(t-1), x_t) - \mathsf{l}_t(\mathbf{h}^\star, x_t) \tag{5.12}$$

where $\sum_{t=1}^{T} \mathsf{l}_t(\mathbf{h}(t-1), x_t)$ is the cumulative loss incurred by the online algorithm. The regret measures how much better or worse the online algorithm performs over the

sequence compared to a fixed learner. An upper bound on the regret indicates the worst-case performance and it is of theoretical interest. If this bound is sub-linear in time, the average regret tends to zero as the sample size grows to infinity, i.e., $\lim_{T\to\infty} \frac{1}{T} R_T(\mathbf{h}^\star) = 0$ [150]. This indicates that the algorithm is learning. We assume the following.

Assumption 4 *The incoming nodes form a maximum of $M_{max} \ll N_t$ edges for all t.*

Assumption 5 *The attachment vectors $\mathbf{a}_t$ and the stochastic model-based weight vectors $\mathbf{w}_t$ are upper-bounded by a scalar w_h. I.e., for all t we have*

$$[\mathbf{a}_t]_n \leq w_h, \ [\mathbf{w}_t]_n \leq w_h. \tag{5.13}$$

Assumption 6 *The filter parameters $\mathbf{h}$ are upper-bounded in their energy, i.e., $\mathscr{E}(\mathbf{h}) = ||\mathbf{h}||_2^2 \leq H^2$.*

Assumption 7 *For all attachment vectors $\mathbf{a}_t$, the residue $r_t = \mathbf{a}_t^\top \mathbf{A}_{x,t-1}\mathbf{h} - x_t$ is upper-bounded. I.e., there exists a finite scalar $R > 0$ such that $|r_t| \leq R$.*

5

Assumption 4 holds for graphs in the real world. A node makes very few connections compared to the total number of nodes, i.e., the attachment vector $\mathbf{a}_t$ will be sparse with a maximum of M_{max} non-zero entries. Note that our stochastic model does not take this into account. Assumption 5 bounds all edge-weights which is commonly observed. Assumption 6 ensures finite parameters which mean the filter output does not diverge. This can be guaranteed by projection on to $\mathscr{H}$. Assumptions 6 and 7 imply bounded filter outputs. Then, we claim the following.

Proposition 6 *Consider a sequence of Lipschitz losses $\{l_t(\mathbf{h}, x_t)\}$ with Lipschitz constants L_d, [cf. (5.9)] and a learning rate η [cf.(5.10)]. Let also Assumptions 4-7 hold. The normalized static regret $R_T(\mathbf{h}^\star)$ for the online algorithm generating filters $\{\mathbf{h}(t)\} \in \mathscr{H}$ relative to the optimal filter $\mathbf{h}^\star \in \mathscr{H}$ is upper-bounded as*

$$\frac{1}{T} R_T(\mathbf{h}^\star) \leq \frac{||\mathbf{h}^\star||_2^2}{2\eta T} + \frac{\eta}{2} L_d^2 \tag{5.14}$$

with $L_d = RC + 2\mu H$ where $||\mathbf{A}_{x,t-1}^\top \mathbf{a}_t||_2 \leq C$.

Proof. From Lemma 3 in Appendix C.4, we have that the loss functions are Lipschitz. Then, we are in the setting of [Thm. 2.13,[133]] from which the rest of the proof follows. □

There are two main filter-related factors that influence the regret bound in (5.14): the filter energy H^2 and the residual energy R^2. A smaller H can lead to a lower bound but it can also increase the prediction error by constraining the parameter set too much. Moreover, a higher regularization weight μ also penalizes high filter energies $||\mathbf{h}||_2^2$. So, for the projected online learner with a high regularization weight μ, a high H can help the inference task, even if it increases the regret bound. Second, from Assumption 7, the residue R is likely small when a filter approximates well the signal on the incoming node. This can happen when the signal values on the incoming

node and the existing nodes are similar or when the existing topology and signals over it are expressive enough to represent the incoming node values. Examples of the latter are locally smooth graph signals that can be approximated by a low order filter K. For high values of K, all nodes have similar signals, implying that many potential attachment patterns can generate x_t. This would make that the manner of attachment irrelevant.

5.4. STOCHASTIC ONLINE FILTERING

Often, the true attachment for the incoming nodes is initially unknown and it is only revealed afterwards. This is the case with rating prediction for cold start recommender systems, where users have initially little to no information, and thus, their connections cannot be inferred. However, their connections can be inferred after they have consumed some items. Instead of waiting for feedback, we can use expanding graph models to infer the signal value and subsequently update the filter online. To address this setting, we first propose an online stochastic update for the filters via specific heuristic models. Then, we propose an adaptive stochastic approach that learns also from an ensemble of topological expansion models.

5.4.1. HEURISTIC STOCHASTIC ONLINE FILTERING

We model the connectivity of node v_t via random stochastic models. Specifically, we use the existing topology $\mathbf{A}_{t-1}$ to fix the attachment probabilities $\mathbf{p}_t$ and weights $\mathbf{w}_t$ using a heuristic attachment rule. Given $\mathbf{a}_t$ is a random vector, the environment reveals the statistical loss

$$\mathsf{l}_t(\mathbf{h}, x_t) = \mathbb{E}[\mathsf{f}_t(\mathbf{h}, x_t)] + \mathsf{r}(\mathbf{h}) \tag{5.15}$$

where the expectation concerns the stochastic attachment model. For $\mathsf{f}_t(\mathbf{h}, x_t)$ being the squared loss, we have the mean squared error expression

$$\begin{aligned}\mathsf{l}_t(\mathbf{h}, x_t) &= \mathbb{E}\left[\frac{1}{2}(\mathbf{a}_t^\top \mathbf{A}_{x,t-1}\mathbf{h} - x_t)^2\right] + \mu||\mathbf{h}||_2^2 \\ &= \frac{1}{2}((\mathbf{w}_t \circ \mathbf{p}_t)^\top \mathbf{A}_{x,t-1}\mathbf{h} - x_t)^2 + \frac{1}{2}(\mathbf{A}_{x,t-1}\mathbf{h})^\top \mathbf{\Sigma}_t \mathbf{A}_{x,t-1}\mathbf{h} + \mu||\mathbf{h}||_2^2\end{aligned} \tag{5.16}$$

where $\mathbf{\Sigma}_t$ is the attachment covariance matrix [cf. (5.2)]. The first term on the r.h.s. of (5.16), $s_t^2 = \frac{1}{2}((\mathbf{w}_t \circ \mathbf{p}_t)^\top \mathbf{A}_{x,t-1}\mathbf{h} - x_t)^2$ is the squared bias between the expected model output $(\mathbf{w}_t \circ \mathbf{p}_t)^\top \mathbf{A}_{x,t-1}\mathbf{h}$ and the true signal x_t. The second term $\frac{1}{2}(\mathbf{A}_{x,t-1}\mathbf{h})^\top \mathbf{\Sigma}_t \mathbf{A}_{x,t-1}\mathbf{h}$ is the variance of the predicted output, and the third term penalizes a high ℓ_2-norm of $\mathbf{h}$.[3] The projected online gradient descent update of the filter parameters $\mathbf{h}(t)$ is

$$\mathbf{h}(t) = \prod_{\mathscr{H}}(\mathbf{h}(t-1) - \eta \nabla_h \mathsf{l}_t(\mathbf{h}, x_t)|_{\mathbf{h}(t-1)}) \tag{5.17}$$

[3] We could also consider adding a penalty parameter to the variance contribution if we want to tweak the bias-variance trade-off in the filter update.

Algorithm 3 Stochastic Online Graph Filtering (**S-OGF**)

Input. Graph $\mathscr{G}_0$, $\mathbf{A}_0$, $\mathbf{x}_0$, $\mathscr{T} = \mathscr{T} = \{v_t, x_t, \mathbf{a}_t\}_{t=1:T}$
Initialize. Pre-train $\mathbf{h}^s(0)$ over $\mathscr{G}_0$ using $\mathbf{A}_0$, $\mathbf{x}_0$.
for $t = 1 : T$ **do**
  Obtain v_t and $\mathbf{p}_t$, $\mathbf{w}_t$ following preset heuristics
  Predict $\hat{x}_t = (\mathbf{w}_t \circ \mathbf{p}_t)^\top \mathbf{A}_{x,t-1} \mathbf{h}^s(t-1)$
  Incur loss $\mathsf{l}_t^s(\mathbf{h}, x_t)$ [cf. (5.16)]
  Update $\mathbf{h}^s(t)$ using (5.17)
  Reveal $\mathbf{a}_t$, update $\mathbf{A}_t$ and $\mathbf{x}_t$
end for

with gradient

$$\nabla_h \mathsf{l}_t(\mathbf{h}, x_t) = ((\mathbf{w}_t \circ \mathbf{p}_t)^\top \mathbf{A}_{x,t-1}\mathbf{h} - x_t)\mathbf{A}_{x,t-1}^\top (\mathbf{w}_t \circ \mathbf{p}_t) + \mathbf{A}_{x,t-1}^\top \mathbf{\Sigma}_t \mathbf{A}_{x,t-1}\mathbf{h} + 2\mu\mathbf{h}. \tag{5.18}$$

The stochastic loss [cf. (5.16)] is differentiable and strongly convex in $\mathbf{h}$. Following Assumption 7, the bias s_t, and the gradient (5.18) are also upper-bounded, making the loss Lipschitz. Algorithm 3 summarizes learning in this setting. The complexity of the online stochastic filter learning at time t is of order $\mathscr{O}(K(M_t + N_t))$. Check Appendix C.4 for further details. Note the dependency on N_t, the size of the graph at time t. Since we do not know the true attachment at the time of making the prediction, the stochastic attachment model assigns probabilities to each node, along with the weights. This leads to the dependence on N_t while making the prediction [cf. (5.16)]. This does not exist in the deterministic case, as we know $\mathbf{a}_t$.

Regret analysis. To characterize the role of the stochastic topological model on the filter update, we compare the cumulative loss between the online stochastic update and the deterministic batch solution. This allows quantifying the performance gap by not knowing the attachment pattern. The regret reads as

$$\frac{1}{T} R_{s,T}(\mathbf{h}^\star) = \frac{1}{T} \sum_{t=1}^{T} \mathsf{l}_t^s(\mathbf{h}^s(t-1), x_t) - \mathsf{l}_t^d(\mathbf{h}^\star, x_t) \tag{5.19}$$

where $\mathsf{l}_t^s(\cdot)$ denotes the stochastic loss and $\mathsf{l}_t^d(\cdot)$ the deterministic loss. Similarly, $\mathbf{h}^d(t-1)$ and $\mathbf{h}^s(t-1)$ denote the online filter at time $t-1$ in the deterministic and stochastic settings, respectively. We claim the following.

Theorem 2 *At time t, let graph $\mathscr{G}_{t-1}$ have N_{t-1} nodes and $\mathbf{h}^s(t-1)$, $\mathbf{h}^d(t-1)$ be the filters learnt online in the stochastic and deterministic scenarios, respectively. Let the nth element of probability vector $\mathbf{p}_t$ be $[\mathbf{p}_t]_n$. Given Assumptions 4-7, the Lipschitz constant L_d, and learning rate η, the normalized static regret for the stochastic setting is upper-bounded as*

$$\begin{aligned} \frac{1}{T} R_{s,T}(\mathbf{h}^\star) \leq \frac{1}{T}\Big(\sum_{t=1}^{T} w_h^2 Y^2 (||\mathbf{p}_t||_2^2 + M_{max}) + 2R w_h Y \sqrt{||\mathbf{p}_t||_2^2 + M_{max}} + w_h^2 Y^2 \bar{\sigma}_t^2 \\ + L_d ||\mathbf{h}^s(t-1) - \mathbf{h}^d(t-1)||_2 \Big) + \frac{||\mathbf{h}^\star||_2^2}{2\eta} + \frac{\eta}{2} L_d^2 T \end{aligned} \tag{5.20}$$

where $\bar{\sigma}_t^2 = \max_{n=1:N_{t-1}} [\mathbf{p}_t]_n(1-[\mathbf{p}_t]_n)$ *and* $||\mathbf{A}_{x,t-1}\mathbf{h}||_2 \leq Y$.

Proof. See Appendix C.1. □

The regret bound in (5.20) depends on the stochastic expanding model and the incoming data as follows:

- The sum of squared norms of the probability vectors corresponding to the attachment rule, via the terms $\sum_{t=1}^{T} ||\mathbf{p}_t||_2^2$ and $\sum_{t=1}^{T}\sqrt{||\mathbf{p}_t||_2^2 + M_{max}}$. This makes the choice of attachment probability $\mathbf{p}_t$ important as it influences the online learner. For example if $\mathbf{p}_t = \mathbf{1}_{N_{t-1}}$ for all t, the sum $\sum_{t=1}^{T} ||\mathbf{p}_t||_2^2$ is of the order T^2, which means the regret bound diverges. Thus, the attachment rule should be selected such that $\sum_{t=1}^{T} ||\mathbf{p}_t||_2^2$ is of order $\mathcal{O}(T)$ or less. However, not all decaying attachment probabilities will reduce the bound reducing. It is necessary to have an inverse dependence on N_t, as is the case for the uniform distribution. This is for example the case of the uniformly at random attachment as we elaborate in Corollary 3.

- The term $\frac{w_h^2 Y^2}{T}\sum_{t=1}^{T}\bar{\sigma}_t^2$ is the sum of the maximum variance for an attachment rule $\bar{\sigma}_t^2$ over time. The maximum value of $\bar{\sigma}_t^2$ is 0.25, attained for an attachment probability of 0.5. For an attachment rule which has either high or low attachment probabilities per node, $\bar{\sigma}_t^2$ will be low, thus contributing less to the regret bound. This means a lower regret can result from stochastic attachment rules with a smaller uncertainty in attachment over the nodes.

- The average distance between the stochastic and deterministic filters over the sequence $\frac{1}{T}\sum_{t=1}^{T} ||\mathbf{h}^s(t-1) - \mathbf{h}^d(t-1)||_2$. If the filter trained with a stochastic attachment is further away from the filter updated with known attachment, the regret is higher. This can happen when the attachment rule cannot model the incoming node attachment and the filter prediction incurs a higher squared error. However, we can use this term to modify the filter update. One way to do this is to include a correction step to update the online filter after the true connection has been revealed. We will discuss this in Remark 2.

- The term $\frac{||\mathbf{h}^\star||_2^2}{2\eta T} + \frac{\eta}{2}L_d^2$ suggests similar factors which affect the deterministic regret will also affect the stochastic regret [cf. (5.14)].

We now present how this regret bound reduces for the uniformly at random attachment.

Corollary 3 *Consider a uniformly at random attachment with* $[\mathbf{p}_t]_n = \frac{1}{N_{t-1}}$. *As the sequence length grows to infinity, i.e.,* $T \to \infty$, *the regret upper bound becomes*

$$\frac{1}{T}R_{s,T}(\mathbf{h}^\star) \leq w_h^2 M_{max} Y^2 + R w_h Y(M_{max}+1) + \frac{1}{T}\sum_{t=1}^{T} L_d ||\mathbf{h}^s(t-1) - \mathbf{h}^d(t-1)||_2 + \frac{||\mathbf{h}^\star||_2^2}{2\eta T} + \frac{\eta}{2}L_d^2 \tag{5.21}$$

Proof. See Appendix C.2. □

Corollary 3 shows that the regret bound in (5.20) can be improved upon with the right choice of attachment rules. Even though the attachment rule helps, there is a chance it fails to model the true attachment process, in which case the term $\frac{1}{T}\sum_{t=1}^{T} L_d||\mathbf{h}^s(t-1)-\mathbf{h}^d(t-1)||_2$ can diverge, ultimately, not making the learner not useful in the steady state.

5.4.2. ADAPTIVE STOCHASTIC ONLINE FILTERING

Oftentimes, a single attachment rule cannot describe the connectivity of the incoming nodes and an ensemble of stochastic rules is needed. This is seen in the regret bound in Theorem 2, which depends on the distance between the stochastic and deterministic online filters. This poses the additional challenge of how to combine these rules for the growing graph scenario. To tailor the combined rule to the online setting, we consider a linear combination of different attachment models and update the parameters as we do for the filter coefficients. Specifically, consider M attachment rules parameterized by the probability vectors $\{\mathbf{p}_{m,t}\}_{m=1:M}$ and the corresponding weight vectors $\{\mathbf{w}_{m,t}\}_{m=1:M}$. Here, $[\mathbf{p}_{m,t}]_i$ denotes the probability of v_t attaching to $v_i \in \mathcal{V}_{t-1}$ under the mth rule and $[\mathbf{w}_{m,t}]_i$ the corresponding weight. Upon defining the dictionaries $\mathbf{P}_{t-1}=[\mathbf{p}_{1,t},\ldots,\mathbf{p}_{M,t}]\in\mathbb{R}^{N_{t-1}\times M}$ and $\mathbf{W}_{t-1}=[\mathbf{w}_{1,t},\ldots,\mathbf{w}_{N,t}]\in\mathbb{R}^{N_{t-1}\times M}$, we combine these models as

$$\bar{\mathbf{p}}_t=\mathbf{P}_{t-1}\mathbf{m} \quad \text{and} \quad \bar{\mathbf{w}}_t=\mathbf{W}_{t-1}\mathbf{n} \tag{5.22}$$

where the combination parameters $\mathbf{m}$ and $\mathbf{n}$ belong to the probability simplex

$$\mathscr{S}^M=\{\boldsymbol{\alpha}\in\mathbb{R}^M,\ \mathbf{1}_M^\top\boldsymbol{\alpha}=1,\ \boldsymbol{\alpha}\succeq\mathbf{0}_M\} \tag{5.23}$$

with $\mathbf{1}_M$ being the vector of M ones. For an existing node v_i, the ith row of $\mathbf{P}_t$ contains the corresponding rule-based probabilities. Equation (5.22) ensures that $\bar{\mathbf{p}}_t$ represents a composite probability vector of attachment with $[\bar{\mathbf{p}}_t]_i=\sum_{l=1}^{M} m_l[\mathbf{p}_{l,t}]_i$ representing the probability of v_t attaching to v_i. It also ensures that the weights in $\bar{\mathbf{w}}_t$ lie in $[0,w_h]$. By representing the expanding graph model via the latent vectors $\mathbf{m}$ and $\mathbf{n}$, we can analyze them in lieu of the growing nature of the problem. This eases the setting as both $\bar{\mathbf{p}}_t$ and $\bar{\mathbf{w}}_t$ grow in dimensions with t since learning these values directly becomes challenging.

Online learner. The instantaneous stochastic loss becomes

$$\mathsf{l}_t(\mathbf{h},\mathbf{m},\mathbf{n},x_t)=\frac{1}{2}((\mathbf{W}_{t-1}\mathbf{n}\circ\mathbf{P}_{t-1}\mathbf{m})^\top\mathbf{A}_{x,t-1}\mathbf{h}-x_t)^2+\frac{1}{2}(\mathbf{A}_{x,t-1}\mathbf{h})^\top\bar{\boldsymbol{\Sigma}}_t\mathbf{A}_{x,t-1}\mathbf{h}+\mu||\mathbf{h}||_2^2 \tag{5.24}$$

where $\bar{\boldsymbol{\Sigma}}_t=\mathrm{diag}((\mathbf{W}_{t-1}\mathbf{n})^{\circ 2}\circ(\mathbf{P}_{t-1}\mathbf{m})\circ(\mathbf{1}_{N_{t-1}}-\mathbf{P}_{t-1}\mathbf{m}))$ is the covariance matrix of this adaptive method. We then proceed with an online alternating gradient descent over the filter parameters $\mathbf{h}$, the composite probability parameters $\mathbf{m}$, and the composite weight parameters $\mathbf{n}$ as

$$\mathbf{h}(t)=\underset{\mathscr{H}}{\Pi}(\mathbf{h}(t-1)-\eta\nabla_h\mathsf{l}_t(\mathbf{h},\mathbf{m},\mathbf{n},x_t)|_{\mathbf{h}(t-1)}) \tag{5.25}$$

Algorithm 4 Adaptive stochastic online filtering (**Ada-OGF**)

Input. Starting graph $\mathcal{G}_0$, $\mathbf{A}_0$, $\mathbf{x}_0$, $\mathcal{T}$
Initialization. Pre-train $\mathbf{h}^s(0)$, Initialize $\mathbf{m}(0) = \mathbf{1}_M/M$, $\mathbf{n}(0) = \mathbf{1}_M/M$. Compute $\mathbf{P}_0$ and $\mathbf{W}_0$.
for t=1:T **do**
  Obtain v_t, $\bar{\mathbf{p}}_t = \mathbf{P}_{t-1}\mathbf{m}(t-1)$, $\bar{\mathbf{w}}_t = \mathbf{W}_{t-1}\mathbf{n}(t-1)$
  Prediction:$(\mathbf{W}_{t-1}\mathbf{n}(t-1) \circ \mathbf{P}_{t-1}\mathbf{m}(t-1))^\top \mathbf{A}_{x,t-1}\mathbf{h}^s(t-1)$
  Reveal loss $l_t(\mathbf{h}, \mathbf{m}, \mathbf{n}, x_t)$
  Update $\mathbf{h}(t)$ following (5.25)
  Update $\mathbf{m}(t)$ following (5.26)
  Update $\mathbf{n}(t)$ following (5.27)
  Reveal $\mathbf{a}_t$, update $\mathbf{A}_t$, $\mathbf{x}_t$, $\mathbf{P}_t$, and $\mathbf{W}_t$
end for

$$\mathbf{m}(t) = \prod_{\mathcal{S}^M} (\mathbf{m}(t-1) - \eta \nabla_m l_t(\mathbf{h}, \mathbf{m}, \mathbf{n}, x_t)|_{\mathbf{m}(t-1)}) \tag{5.26}$$

$$\mathbf{n}(t) = \prod_{\mathcal{S}^M} (\mathbf{n}(t-1) - \eta \nabla_n l_t(\mathbf{h}, \mathbf{m}, \mathbf{n}, x_t)|_{\mathbf{n}(t-1)}) \tag{5.27}$$

where $\Pi_{\mathcal{S}^M}(\cdot)$ is the projection operator onto the probability simplex $\mathcal{S}^M$ and the gradient closed-form expressions are given in Appendix C.4. After the update, the environment reveals the true attachment $\mathbf{a}_t$ and we update $\mathbf{A}_t$ and $\mathbf{x}_t$. We also update $\mathbf{P}_t$ based on the ensemble of attachment rules applied on the updated topology and the weight dictionary as $\mathbf{W}_t = [\mathbf{W}_{t-1}; \mathbf{e}_t^\top] \in \mathbb{R}^{N_t \times M}$ where $\mathbf{e}_t \in \mathbb{R}^M$ contains independent positive random variables sampled uniformly between zero and the maximum possible edge weight w_h. Algorithm 4 highlights the adaptive stochastic online learning. The computational complexity at time t for Ada-OGF is of order $\mathcal{O}(K(M_0 + N_t) + N_t M)$. See Appendix C.4 for more details. The loss function in (5.24) is jointly non-convex in $\mathbf{h}$, $\mathbf{n}$, and $\mathbf{m}$. It is marginally convex in $\mathbf{n}$ and $\mathbf{h}$ but not in $\mathbf{m}$ due to the nature of the covariance matrix. We can run multiple projected descent steps for each of the variables, but proving convergence is non-trivial. However, convergence to a local minimum of $l_t(\cdot)$ may not even be needed as we are in an online non-stationary setting where the arrival of another node leads to a new loss function. Thus, it is reasonable to take one or a few projected steps for each incoming node even without a full convergence guarantee.

Regret analysis. For the regret analysis of the stochastic adaptive online method, we claim the following.

Corollary 4 *Given the hypothesis of Theorem 2 and an adaptive stochastic online method over M attachment rules with $\{\mathbf{P}_t\}$, the normalized static regret w.r.t. the*

deterministic batch learner is upper-bounded as

$$\frac{1}{T}R_{s,T}(\mathbf{h}^\star) \leq w_h^2 Y^2 \frac{1}{T}\sum_{t=1}^{T}\left(||\mathbf{P}_{t-1}||_F^2 + M_{max}\right) + R w_h Y \frac{1}{T}\sum_{t=1}^{T}||\mathbf{P}_{t-1}||_2^2 + R w_h Y (1+M_{max}) + w_h^2 Y^2 \frac{1}{T}\sum_{t=1}^{T}\bar{P}_t + \frac{1}{T}\sum_{t=1}^{T} L_d ||\mathbf{h}^s(t-1) - \mathbf{h}^d(t-1)||_2 + \frac{||\mathbf{h}^\star||_2^2}{2\eta T} + \frac{\eta}{2}L_d^2 \tag{5.28}$$

where $\bar{P}_t = \max_{n=1:N_{t-1}} ||[\mathbf{P}_{t-1}]_{n,:}||_2$ *and* M_{max} *is the maximum number of edges formed by each incoming node.*

Proof. See Appendix C.3. □

Compared to the single heuristic attachment model, the regret in (5.28) depends on the sum of ℓ_2 norm squared of all the M attachment rules. It also depends on $\bar{P}_t$, which is the maximum norm of the vector of probabilities for all rules for each node. The bound in (5.28) holds when selecting one attachment rule at each time, i.e., $||\mathbf{m}(t)|| = 1$ for all t. However, a smaller norm of $\mathbf{m}(t)$, corresponding to considering all rules leads to a lower regret bound, potentially improving the performance. Moreover, we expect the term concerning the distance between the stochastic and determinisitc filters to reduce due to the adaptive updates, thus, reducing the bound. We shall empirically corroborate this in Section 5.5.

Remark 2 *Prediction Correction Online Graph Filtering (**PC-OGF**.) In the stochastic algorithms the bounds* (5.20) *and* (5.28) *show that the regret is influenced by the difference between deterministic filters (that know the attachment) and the stochastic filters (that do not know the attachment) via the term* $||\mathbf{h}^s(t-1) - \mathbf{h}^d(t-1)||_2$. *One way to reduce the regret is to leverage the attachments after they are revealed and correct the learned stochastic filter coefficients via a deterministic update. This corresponds to using the prediction correction framework [151]. The prediction step corresponds to performing the filter update based on the predicted output in the absence of connectivity information. The correction step performs an additional update on the prediction step by updating the filter for a loss function with the known attachment. The prediction and correction steps corresponds to one step of S-OGF and D-OGF, respectively. Algorithm 5 highlights this approach. The computational complexity of this at time t is of order* $\mathcal{O}(K(M_t + N_t + M_{\max}))$, *as it comprises one step of **S-OGF** followed by one of **D-OGF**.*

5.5. NUMERICAL EXPERIMENTS

We corroborate the proposed methods for regression tasks on both synthetic and real data-sets. We consider the following baselines and state-of-the-art alternatives.

1. **D-OGF** [Alg. 2]. This is the proposed online method for deterministic attachment. We search the filter order $K \in \{1,3,5,7,9\}$ and the learning rate η and the regularization parameter μ from $[10^{-6}, 1]$.

Algorithm 5 Prediction Correction Online Graph Filtering (**PC-OGF**)

Input. Graph $\mathcal{G}_0$, $\mathbf{A}_0$, $\mathbf{x}_0$, $\mathcal{T} = \mathcal{T} = \{v_t, x_t, \mathbf{a}_t\}_{t=1:T}$
Initialize: Pre-train $\mathbf{h}^s(0)$ over $\mathcal{G}_0$ using $\mathbf{A}_0$, $\mathbf{x}_0$.
for $t = 1 : T$ **do**
  Obtain v_t and $\mathbf{p}_t$, $\mathbf{w}_t$ following preset heuristics
  Predict $\hat{x}_t = (\mathbf{w}_t \circ \mathbf{p}_t)^\top \mathbf{A}_{x,t-1} \mathbf{h}^s(t-1)$
  Incur loss $l_t^s(\mathbf{h}, x_t)$ [cf. (5.16)]
  Update $\mathbf{h}^s(t)$ using (5.17)
  Reveal $\mathbf{a}_t$, update $\mathbf{A}_t$ and $\mathbf{x}_t$
  Update $\mathbf{h}^s(t)$ using (5.10)
end for

2. **S-OGF** [Alg. 3]. This is the proposed online method using one stochastic attachment rule. We consider a uniformly at random attachment rule for $\mathbf{p}_t$. For $\mathbf{w}_t$, we use the same weight for each possible edge, which is the median of the edge weights in $\mathcal{G}_{t-1}$. We obtain the regularization parameter μ and step-size η via grid-search over $[10^{-5}, 10^{-1}]$.

3. **Ada-OGF** [Alg. 4]. This is the proposed adaptive stochastic online method. We take $M = 5$ with attachment rules based on the following node centrality metrics: i) Degree centrality; ii) Betweenness centrality [152]; iii) Eigenvector centrality [153]; iv) Pagerank; v) Uniform.

4. **PC-OGF** [Remark 2.] This is the two-step update method. For the prediction step, we perform S-OGF with uniformly-at-random $\mathbf{p}_t$ and $\mathbf{w}_t$ as considered for S-OGF above. For the correction step, we perform one step of D-OGF. Both steps share the same learning rate $\eta \in [10^{-5}, 10^{-1}]$ and $\mu \in [10^{-5}, 10^{-1}]$.

5. **Batch.** This is the filter designed by taking into account the whole node sequence, i.e.,
$$\mathbf{h}^\star = \underset{\mathbf{h} \in \mathbb{R}^{K+1}}{\operatorname{argmin}} \sum_{t=1}^{T} (\mathbf{a}_t^\top \mathbf{A}_{x,t-1} \mathbf{h} - x_t)^2 + \mu ||\mathbf{h}||_2^2 \tag{5.29}$$
which has a closed-form least-squares expression for $\mu > 0 \in [10^{-3}, 10]$.

6. **Pre-trained.** This is a fixed filter trained over the existing graph $\mathcal{G}_0$ and used for the expanding graphs. We train the filter over 80% of the data over $\mathcal{G}_0$. The regularization parameter is chosen over $[10^{-3}, 10]$.

7. **OKL** *Online Multi-Hop Kernel Learning* [28]. We consider a Gaussian kernel with variance $\sigma^2 \in \{0.1, 1, 10\}$. The number of trainable parameters is the same as that of the filters for a fair comparison.

8. **OMHKL** *Online Multi-Hop Kernel Learning* [136]. This method considers multi-hop attachment patterns which are then fed into the random feature framework. We take the multi-hop length as the filter order. We consider one kernel for each hop with the same variance selected from $\sigma^2 \in \{0.1, 1, 10\}$. We

did not optimize over the combining coefficients for each multi-hop output. This is to keep the comparisons fair, as OMHKL has more parameters. Instead, we take the mean output, while updating the regression parameter for each multi-hop.

The hyper-parameters are chosen via a validation set. For each parameter, we perform a grid search over a specific range for each data-set, as indicated above for each approach. We use the same filter order as determined for **D-OGF** for the other online filters. We use the same filter order as determined for **D-OGF** for the other online filters. For all data sets, we divide the sequence of incoming nodes into a training and a test sequence. The first 80 percent of the incoming node sequence are taken as the training nodes. The remaining 20 percent are the test nodes. The nodes in the training sequence are used to tune the hyper-parameters, while the test set is used to evaluate the online method for the selected hyper-parameters.

5.5.1. EXPERIMENTAL SETUP

We consider a synthetic setup based on a random expanding graph model; and two real data setups based on recommender systems and COVID case predictions.

Synthetic. We start with a graph $\mathcal{G}_0$ of $N_0 = 100$ nodes and an edge formation probability of 0.2. The edge weights of $\mathbf{A}_0$ are sampled at random from the uniform distribution between zero and one. Each incoming node v_t forms five uniformly at random edges with the existing graph $\mathcal{G}_{t-1}$. Each newly-formed edge weight is the median of the edge weights in $\mathcal{G}_0$. The existing graph signal $\mathbf{x}_0$ is band-limited w.r.t. the graph Laplacian, making it low-pass over $\mathcal{G}_0$ with a bandwidth of three [8]. We generate the true signal x_t at the incoming v_t in three ways to have three different types of data that fit the different methods.

1. *Filter.* The true signal x_t is generated using a pre-trained filter of order five on $\mathcal{G}_0$. This setting is the closest to the proposed approach and is meant as a sanity check. It also helps us to investigate the differences between the deterministic and the stochastic attachments.

2. *WMean.* x_t is the weighted mean of the signals at the nodes v_t attaches to. This is a neutral setting for all methods.

3. *Kernel.* x_t is obtained from a Gaussian kernel following [28]. This prioritises kernel-based solutions and it is considered here as a controlled setting to compare our method in a non-prioritized setup.

We average the performance of all methods over 10 initial graphs $\mathcal{G}_0$ and each having $T = 1000$ incoming nodes with 800 incoming nodes for training and 200 for testing.

Cold-start recommendation. We consider the Movielens100K data-set that comprises 100,000 ratings provided by 943 users over 1152 items [154]. We build a 31 nearest neighbour starting graph of 500 random users and consider the remaining 443 users as pure cold starters for the incoming sequence. We use the cosine similarity of the rating vectors to build the adjacency matrix of this graph. We use 50 percent

of the ratings of each new user v_t to build $\mathbf{a}_t$. We evaluated all methods over 10 realizations of this setup, where, in each realization, we shuffle the order of incoming users. All methods perform online learning over 16875 and 6155 ratings in the training and test sets, respectively.

COVID case prediction. Here, we predict the number of COVID-19 infection cases for an uninfected city in an existing network of currently infected cities. We consider the data from [155] that has daily case totals for 269 cities and focus on a subset of 302 days of this data-set as in [156]. We randomly select 50 cities and build a five nearest neighbour-directed graph $\mathcal{G}_0$. The edge weight between cities v_i and v_j is $A_{i,j} = \exp(-\frac{||\mathbf{t}_i - \mathbf{t}_j||_2^2}{2\sigma^2})$, where $\mathbf{t}_i$ and $\mathbf{t}_j$ are the vector of COVID cases from day one to 250 for cities v_i and v_j, respectively. We also use this interval to calculate the attachment vector $\mathbf{a}_t$ for any incoming city node. We evaluate the performance on each of the days 255, 260, 265, 270, 275, and 280 and predict the COVID case strength for each node city in the sequence. For each day, we carried out twenty realizations where we shuffle at random the order in which the cities are added to the starting graph.

We measure the performance via the root normalized mean square error NRMSE

$$\text{NRMSE} = \frac{\sqrt{\frac{1}{T}\sum_{t=1}^{T}(\hat{x}_t - x_t)^2}}{\max\limits_t\ (x_t) - \min\limits_t\ (x_t)} \tag{5.30}$$

where $\hat{x}_t$ and x_t are the predicted and true signal at v_t, respectively. This measure gives a more realistic view of the performance since the incoming data does not follow a specific distribution and it is susceptible to outliers [157]. Additionally, we measure the normalized static regret (NReg) [cf. (5.12)] for the online methods w.r.t. the Batch solution.

5.5.2. PERFORMANCE COMPARISON

Table 5.1 comprises the NRMSEs and the standard deviations for all methods. We observe the following:

Deterministic approaches. D-OGF outperforms OKL and OMHKL across all the data-sets. The difference is more pronounced for the data generated using the *Filter* and the *WMean* method, as they are suited for filters, whereas for the *Kernel* data, the difference is smaller. For the Movielens data-set, the difference is also small. We suspect this is because we train one filter across many graph signals (each graph signal corresponds to a different item) over the same user graph, whereas the kernel method ignores the graph signals. It is possible to improve the prediction accuracy by considering item-specific graphs as showcased in [10, 148]. Next, we observe that D-OGF performs better than pre-trained throughout the experiments. This is because the online filters adapt to the incoming data stream, while the pre-trained does not. The only case we can expect a similar performance is where the incoming data is similar to the data over the existing graph.

Concerning the batch solution, we find that the deterministic online learner outperforms Batch in all data-sets apart from the COVID data-set. This shows

the limitations of batch-based solutions, i.e., an over-dependence on the observed training data, and also an inability to adapt to the sequence. For *Filter* and *WMean* data, the training and test set distributions are similar, so the difference between D-OGF and Batch can be attributed to the adaptive nature of D-OGF. In the other data-sets, the change in distribution is detrimental for the batch learner, particularly in the Movielens data.

Stochastic approaches. The S-OGF and Ada-OGF approaches have a similar performance for *Filter*, *Kernel* and Movielens data, with Ada-OGF performing better for *WMean* and Covid data. This makes sense for the synthetic data as the constructed graphs expand following a uniformly at random attachment rule, the same rule used for **S-OGF**. The standard deviation is on the lower side for Ada-OGF. Since the existing signal $\mathbf{x}_0$ is band-limited, the signal values obtained via a filtering/ mean operation with a uniformly at random attachment will also be similar. However, Ada-OGF performs better for the COVID data. This is because the incoming data in the COVID data-set is quite different from the synthetic data. It does not have properties like smoothness and thus a uniformly at random attachment cannot help in predicting the number of cases. In such a setting, a more adaptive approach will help. For Movielens data, there is no difference between the two methods, possibly due to the high number of ratings.

Deterministic vs stochastic. The deterministic methods outperform the stochastic counterparts as expected. The gap is closer for the *Kernel* and Movielens data. For Movielens this can be attributed to the subsequent filter updates that are done for different graph signals over a fixed graph. This can cause high prediction errors. The same holds also for the kernel method, as it is based on the same graph. Since the filter takes the signal into account, it might be affected more.

In the Movielens, *Kernel*, and COVID data, the pre-trained filter does not update itself and is thus at a disadvantage, compared to the online methods. For COVID data, the signal over the incoming node, i.e., the number of cases can be quite different from the signals over which the pre-trained filter is learnt, accounting for a higher error. Among the proposed online methods, the stochastic online methods perform poorly w.r.t pre-trained for *Filter* and *WMean* data. This is expected as the data distribution of the incoming data is similar to that over $\mathcal{G}_0$ for these scenarios.

The PC-OGF method performs better than the stochastic methods for all data, showing the added value of correcting for the true attachment. It even outperforms D-OGF for *Kernel* data.

5.5.3. ANALYSIS OF ONLINE METHODS

We now investigate more in detail the online methods.

Outliers. In Figs. 5.2 and 5.3 we show the violin plots of the squared errors over the test set for the Movielens and COVID data. The deterministic methods suffer more from higher outlier errors. This could be attributed to errors in estimating the attachment vectors. For Movielens data, we calculate this similarity over a subset of the items and for certain splits of the items this may lead to estimation errors for the similarity and thus also for the attachment vector. One reason why the stochastic methods are not prone to outliers could be the term in the loss functions

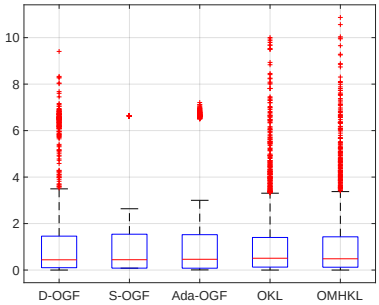


Figure 5.2.: Box plot of the squared errors for each method in the Movielens data-set.

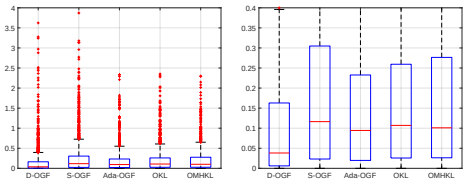


Figure 5.3.: (Left) Box plot of the squared errors across all data points over the six days. (Right) Box plot on the right zoomed in to highlight differences between all methods.

[cf. (5.16), (5.24)] that penalizes the prediction variance, ultimately, acting as a robust regularizer. Notably, for the Movielens data the errors in the stochastic online learners are fixed at certain levels. This is because the data-set has only five fixed values as ratings and because both S-OGF and Ada-OGF predict fixed values [cf. first term in (5.16)]. For the Covid data, we calculated the number of outliers in the squared error. The outlier counts are **D-OGF** = 144, **S-OGF** = 123, **Ada-OGF** = 119, **OKL** = 94, **OMHKL** = 98. This could also be due to the way the starting graph and the links of the incoming nodes are constructed. The figure on the right zooms in on the plot in the range between zero and 0.4. The D-OGF has lower NRMSE, implying the presence of many samples with low squared error. The patterns for the other methods are similar.

Filter order, p, w. Next, we investigate the role of the filter order as well as the impact of training both the attachment probabilities $\mathbf{p}_t$ and weights $\mathbf{w}_t$. Thus, we also want to compare with an alternative adaptive approach where we update only $\mathbf{p}$ while keeping $\mathbf{w}$ fixed to the true edge weights. We call this **Ada2-OGF**. We generate *Filter* data, *WMean* data, and *Kernel* data with a variance of 10. The filter orders evaluated are $K \in \{1,3,5,7,9\}$. Figure 5.4 shows the variation of RNMSE of the filter approaches with filter order K. We see that **Ada2-OGF** performs worse than **Ada-OGF** apart from the *Filter* data. This suggests that updating both $\mathbf{p}$ and

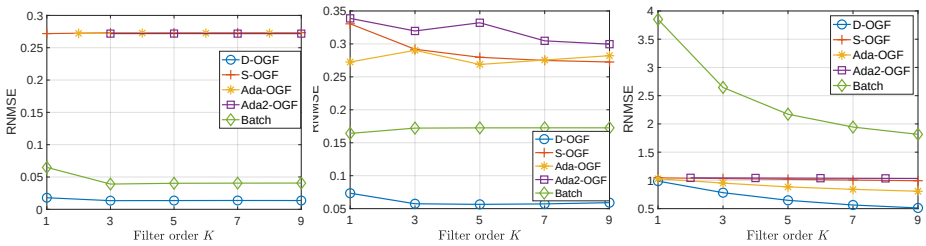


Figure 5.4.: RNMSE for different values of filter order K for (left) *Filter*, (centre) *WMean*, and (right) *Kernel* data, respectively.

w is beneficial than just updating **p**. For the filter data, we see that all the three stochastic approaches perform the same. This is because we same stochastic rule, i.e., uniformly at random attachment for data generation. **S-OGF** uses the same, while **Ada-OGF** learns it.

Learning rate. Figure 5.6 shows the normalized cumulative regret at each time of **D-OGF** w.r.t. the batch learner for different values of the learning rate η for each synthetic-dataset. Increasing η leads to a lower regret, but after one point, the regret increases. For the *Kernel* data, for example, we see that the regret increases sharply between learning rate $\eta = 3$ and $\eta = 5$. This shows that η indeed influences the online learner and its optimal value is in principle neither too high or too low. A higher value than the optimal misleads the online learner by focusing too much on the current sample. This can lead to high prediction errors for some samples, as seen in the spikes in the plots. A lower value learns about the incoming data-stream at a slower rate.

Regret. Figure 5.5 plots the normalized cumulative regret at each time for **S-OGF** and **Ada-OGF** w.r.t. the batch solution for the *Filter* (left), *WMean* (center) and *Kernel* (right) data, respectively. In all three cases, the average cumulative regret converges, implying that the cumulative error or the gap with the batch solution does not diverge. This shows that the stochastic learners, despite not having access to the connectivity at the time of making a prediction, can learn from more incoming nodes. Second, **Ada-OGF** showcases a lower regret than **S-OGF**, showing that it can learn faster from the incoming nodes by trying to predict the attachment behaviour. This is in agreement with the regret bounds in Theorem 2, Corollaries 3 and 4.

Finally, we investigate the normalized regret over the whole sequence for the online methods in Table 5.2. Since we evaluate this over the training set, we have positive values, which implies the batch solution has a lower cumulative error. However, having a positive regret during training can also lead to a lower NRMSE than the batch solution over the test set, as is the case for **D-OGF** [cf. Table I]. This is because the batch filter is fixed and cannot perform as well as in the training set if the distribution of data in the test set is different. The lower regret for **D-OGF** compared to the stochastic approaches stems from the fact that the connectivity is known and because the Batch solution also has a similar loss function. The normalized regret for **PC-OGF** is lesser than that of the stochastic

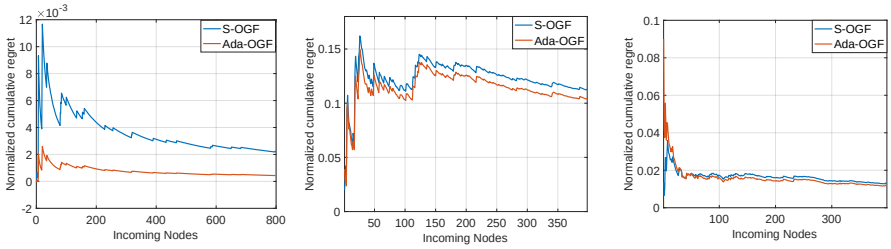


Figure 5.5.: Evolution of the normalized cumulative regret for S-OGF and Ada-OGF for the synthetic (left) *Filter*, (center) *WMean* and (right) *Kernel* data for $T = 800$, T and 400 incoming nodes, respectively.

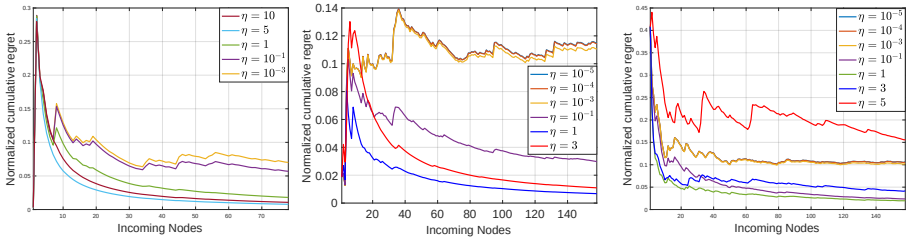


Figure 5.6.: Normalized cumulative regret evolution for different values of learning rate η for (left) *Filter*, (centre) *WMean*, and (right) *Kernel* data, respectively. The reference average error for the batch solution over the training set are 3×10^{-5}, 6.8×10^{-4}, and 1.1×10^{-2}, respectively.

approaches, showing that incorporating the attachment can counter the effect of the gap between the stochastic and deterministic filter.

5.6. CONCLUSION

We proposed online filtering over graphs that grow sequentially over time. We adapted the formulation to the deterministic scenario where the connection of the incoming nodes is known and to a stochastic scenario where this connection is known up to a random model. We performed a simple projected online gradient descent for the online filter update and provided performance bounds in terms of the static regret. In the stochastic setting, the regret is a function of the rule-specific probabilities along with their variance. Numerical results for inference tasks over synthetic and real data show that graph filters trained online learning perform collectively better than kernel methods which do not utilize the data, pre-trained filters, and even a batch filter.

For future work, we will consider the scenario where the signal also varies over the existing graph, i.e., it has a spatio-temporal nature. It is also possible to consider the scenario of joint topology and filter learning over the expanding graphs, where

we estimate the true attachment of the incoming node instead of a stochastic model along with the filter used for making the inference. Finally, to account for the robustness of the online methods, one can also perform a weighted update, where the loss at a particular time is a weighted sum of the previous samples. The complexity of the stochastic approaches grow with the size of the graph. To tackle this, distributed filter updates can be a viable approach.

Table 5.1.: Average NRMSE and standard deviation of all approaches for all data-sets.

	Synthetic Data						Real Data			
Method	Filter		WMean		Kernel		Movielens100K		COVID	
	NRMSE	Sdev	NRMSE	Sdev	NRMSE	Sdev	NRMSE	Sdev	NRMSE	Sdev
D-OGF (ours)	0.02	0.003	0.02	0.005	0.25	0.04	0.26	0.01	0.21	0.02
S-OGF (ours)	0.18	0.02	0.26	0.06	0.28	0.07	0.28	0.007	0.31	0.02
Ada-OGF (ours)	0.18	0.02	0.25	0.04	0.28	0.05	0.28	0.007	0.26	0.007
PC-OGF (ours)	0.18	0.02	0.22	0.02	0.23	0.04	0.27	0.01	0.26	0.003
Batch	0.04	0.007	0.09	0.04	1.3	0.29	6.7	0.1	0.17	0.03
Pre-trained	0.08	0.03	0.09	0.03	0.53	0.28	0.84	0.02	2.5	0.9
OKL	0.17	0.01	0.23	0.02	0.25	0.04	0.27	0.01	0.25	0.02
OMHKL	0.17	0.01	0.32	0.1	0.34	0.09	0.27	0.01	0.25	0.02

Table 5.2.: Normalized regret for the online methods for synthetic data.

Method	Filter	WMean	Kernel
D-OGF	1.6×10^{-4}	0.03	0.01
S-OGF	2.2×10^{-3}	0.84	0.08
Ada-OGF	4.1×10^{-3}	0.82	0.11
PC-OGF	1.9×10^{-4}	0.27	0.02

6

DYNAMIC GRAPH TOPOLOGY DECOMPOSITION

In Chapters 3, 4, and 5, we focused on signal processing on expanding graphs. There is a focus on processing data while adapting to the change in topology with or without uncertainty, instead of a focusing on the structural evolution itself. In the current chapter, we focus on this aspect for dynamic graphs with changing support. Existing approaches analyse dynamic graphs through low-rank tensor decomposition, without a focus on interpretability or the structural component of the evolution. Moreover, there are often missing observations, i.e., we do not know if edges exist or not due to outages, faulty sensors, or privacy requirements. Targeting these challenge, we propose a novel two-way decomposition of the adjacency tensor to represent a dynamic graph topology where we express the structural evolution as a linear combination of latent graphs whose adjacency matrices that capture the overall evolution. Additionally, we also incorporate graph signal information into this decomposition via a smoothness prior. We estimate the adjacency matrices along with their temporal scaling signatures via alternating minimization in the presence of spatio-temporal data. We show that our approach converges to a stationary point. Numerical results show that the proposed approach recovers individually and collectively expressive latent graphs that outperform the typical tensor decomposition and signal-based topology identification for reconstructing the missing network.

The rest of the chapter is organized as follows. In Section 6.2, we introduce the dynamic network decomposition model. In Section 6.3, we formulate our problem and propose an alternating optimization algorithm to solve for the dynamic network decomposition under the desired constraints. In Section 6.4, we analyze the proposed algorithm w.r.t. its convergence. In Section 6.5, we perform experiments to highlight different aspects of the proposed decomposition and compare its performance with alternatives. Section 6.6 concludes the chapter. All proofs are collected in Appendix D[1].

[1]This chapter is based on the submission: Das, B., & Buciulea, A. & Marques, A. G., & Isufi. E., Dynamic Graph Topology Decomposition, IEEE Transactions on Signal Processing and the following publication: Das, B., & Isufi. E., Tensor Graph Decomposition for Temporal Networks, IEEE ICASSP 2024

6.1. INTRODUCTION

Dynamic or temporal networks witness a structural evolution in the form of changing edges between nodes [11, 62, 63, 158]. Analyzing their evolution is important for recovering hidden patterns. Often dynamic networks co-occur alongside dynamic graph signals, which represent the evolution of signals over the nodes. Since graph signals form an association between nodes and data, dynamic graph signals can convey important information about the change in topology. Learning graphs from dynamic graph signals has been approached from several viewpoints wherein the topology is usually inferred through a batch or online learning problem trying to minimize loss functions with graph-based priors [91, 92, 138, 139, 159–163]. One can also infer multiple graphs from dynamic graph signals. In [21], the authors learn muti-view graphs which vary around a single consensus graph. The work in [22] learn multiple graphs from streaming node signals that differ from each other in terms of their sets of central nodes via eigen-centrality. The work in [23] learn a set of graphs from observed spatio-temporal variations using spatial independent component analysis [164]. The notion of having a set of known or unknown underlying graphs has been used to perform signal processing tasks in [24, 161, 165–167]. These approaches infer multiple graph solely from signals under various assumptions and settings. A drawback of topology identification from graph signals (static or dynamic) is the choice of the prior, which may not be known in advance.

Another choice for representing the structure of dynamic networks is by using tensors [70]. This is typically done by stacking their topological matrix representation along the temporal dimension to obtain a three-dimensional tensor. Exploiting the low-rank structure of their matrix and tensor representations has been shown to be useful for different applications, like community detection. One advantage of such a representation is that it allows using low-rank tensor decomposition [71–73] for compressing and representing the dynamic network, which aids in downstream tasks.

Tensor decompositions of dynamic networks have been used for different applications. For example, the works in [32, 33, 36] use low-rank tensor decomposition for community detection. The work in [34] applies tensor decomposition for graph summarization. An example of link prediction can be seen in [37] but the tensor here represents a changing heterogenous user-item graph. Another example involves anomaly detection [38]. The works in [39, 40, 168] build tensors from MRI data and use low-rank decomposition for tracking the states of dynamic networks. However these approaches face the following limitations:

- *Role of the Graph.* They treat the adjacency tensor from a purely tensor perspective ignoring the graph structure in it, ultimately, finding low rank decompositions that do not carry any particular graph-related information. This is partially because the focus lies on down-stream tasks more than capturing the topological component. However, considering the graph structure in the decomposition is important because the embeddings of the decomposition should be associated with the graph in some form. This is normally enforced via priors. There exist works which incorporate graph structure as a regularizer to decompose data tensors [169], but these do not apply to tensors comprising

the structure of dynamic graphs.

- *Use of signals.* They do not account for the graph signals in these decompositions. This data is either directly used as a tensor or incorporated in building it. Incorporating graph signals to uncover underlying topologies has been the basis for several topology identification approaches [76, 170, 171] and can help recover more accurate topological representations as well as identify more meaningful latent graphs that influence both the topological and signal evolution.

- *Availability of observations.* They consider a fully available topology, i.e., do not account for hidden observations, apart from the work on link prediction [37] which uses a direct decomposition on a heterogenous graph tensor.

6.1.1. CONTRIBUTIONS OF THIS CHAPTER

Taking inspiration from the multi-graph viewpoint of dynamic graphs along with their relation with graph signals, we propose a topology-and-signal-aware decomposition of partially observed dynamic networks. Our decomposition has at its core a set of latent graphs, whose linear combination expresses the observed structural evolution. We use the dynamic graph signals to aid the recovery of these latent graphs, especially when we do not observe the dynamic topology fully. This decomposition thus results in finding these latent graphs and their temporal signatures, responsible for scaling them.

We provide a general framework that allows representing a partially observed dynamic network via latent graphs and signatures which can be adapted to incorporate different underlying relationships. This framework is characterised by the following features:

1. It leads to interpretable latent graph decompositions.

2. It can accommodate spatiotemporal nodal signals as a prior to further aid the represent of dynamic networks, especially when the number of observations is limited.

3. It allows for an alternating optimization algorithm to recover the modal graphs and their corresponding time signatures with convergence guarantees to stationary points.

4. Shows good performance in practice with respect to reconstructing the missing network, compared to low-rank tensor decompositions and alternatives which are not topology-aware.

6.2. PROBLEM FORMULATION

Consider a dynamic undirected graph $\mathcal{G}_t = (\mathcal{V}, \mathcal{E}_t)$ with a fixed node set $\mathcal{V} = \{1, \dots, N\}$ and the time-varying edge set $\mathcal{E}_t$ that captures the evolving structure over T temporal samples. We represent the graph evolution through a three-dimensional adjacency

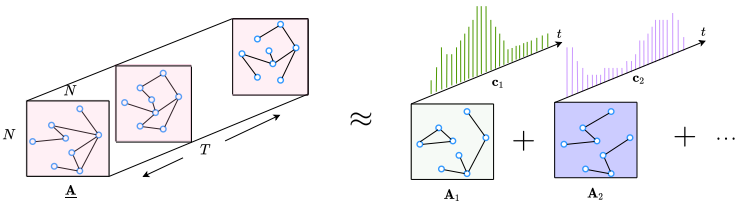


Figure 6.1.: Decomposition of the temporal topology evolution in terms of the component adjacency tensors. (left) an evolving topology over N nodes for T time instants stacked into a three-dimensional tensor. (right) two modes $\mathbf{A}_1$ and $\mathbf{A}_2$ and their corresponding temporal signatures $\mathbf{c}_1$ and $\mathbf{c}_2$ whose linear combinations approximate $\underline{\mathbf{A}}$.

tensor $\underline{\mathbf{A}} \in \mathbb{R}^{N\times N\times T}$, where the t-th frontal slice $\underline{\mathbf{A}}_{:,:,t} \in \mathbb{R}^{N\times N}$ is the $N\times N$ adjacency matrix representing the topology at time t. We have the (i,j)th entry $[\underline{\mathbf{A}}_{:,:,t}]_{ij} > 0$ if there exists an edge between nodes i and j at time t and $[\underline{\mathbf{A}}_{:,:,t}]_{ij} = 0$ otherwise.

Often we do not observe the whole topology at a given time instant. This is common in evolving social networks where some users may not reveal their connections over some time due to privacy reasons [172]. This can also occur due to communication outages in sensor networks when they operate in harsh environments [173]. We model these missing observations via a three-dimensional binary tensor mask $\underline{\mathbf{M}} \in \{0,1\}^{N\times N\times T}$ where $[\underline{\mathbf{M}}_{:,:t}]_{i,j} = [\underline{\mathbf{M}}_{:,:t}]_{j,i} = 1$ if we observe the presence or absence of an edge between nodes i and j at time t. As a consequence, we only observe a part of the structural evolution, which writes as $\underline{\mathbf{M}} \circ \underline{\mathbf{A}}$ where $\circ$ denotes the element-wise Hadamard product.

Given the observed tensor, the typical low-rank decomposition like CPD [71] will express it as a sum of outer products of vectors, with the first two outer products forming a rank one matrix. Rank one matrices do not capture the topology of graphs successfully. Differently, block term decompositions [72] can offer structurally richer decompositions, but their low rank can still be a limiting factor as adjacency matrices are not typically low rank, as is observed from priors used to estimate them [76].

To impose a richer graph structure to the decompositions, we model the true topology at each time instant in $\underline{\mathbf{A}}$ as a weighted sum of $R \ll T$ latent deterministic topologies in the form of adjacency matrices, each of size $N\times N$. The latent graphs denote static connections which may not be present at each time instant in the observed dynamic topology but can account for important hidden connections between the nodes which may not always be apparent. In addition, there may be more than one latent graph. Some edges can exist together as they imply a certain type of interaction in time. Thus, they can be assigned to one graph and different graphs can together express the topology evolution. Some latent graphs may or may not exist together, due to physical reasons. This is observed in switching dynamical models. The latent graphs, thus, collectively can account for the structural evolution of the observed topology.

More specifically, let the rth adjacency matrix be $\mathbf{A}_r \in \mathbb{R}^{N\times N}$ and consider the vector $\mathbf{c}_r \in \mathbb{R}^T$ with $[\mathbf{c}_r]_t$ denoting the scalar weight of $\mathbf{A}_r$ at time t. With this in place, we model the observed topology as

$$\underline{\mathbf{M}} \circ \underline{\mathbf{A}} = \underline{\mathbf{M}} \circ \left[\sum_{r=1}^{R} \mathbf{A}_r \diamond \mathbf{c}_r + \underline{\mathbf{E}} \right], \tag{6.1}$$

where the approximate adjacency tensor $\hat{\underline{\mathbf{A}}} = \sum_{r=1}^{R} \mathbf{A}_r \diamond \mathbf{c}_r \in \mathbb{R}^{N\times N\times T}$ is the sum of outer products $\diamond$ between the adjacency matrices $\{\mathbf{A}_r\}$ and their corresponding temporal signature vectors $\{\mathbf{c}_r\}$. The tensor $\underline{\mathbf{E}}$ represents noisy edge perturbations independent of the dynamics imposed by the latent graph adjacency matrices $\{\mathbf{A}_r\}$s. The adjacency matrix at time t $\underline{\mathbf{A}}_{:,:,t}$ is

$$\underline{\mathbf{A}}_{:,:,t} = \sum_{r=1}^{R} [\mathbf{c}_r]_t \mathbf{A}_r + \mathbf{E}_t, \tag{6.2}$$

i.e., a linear combination of the R latent adjacency matrices, each scaled by the corresponding elements in $\mathbf{c}_r$. Figure 6.1 shows the adjacency tensor $\underline{\mathbf{A}}$, along with the corresponding model approximation in terms of two latent adjacency matrices $\mathbf{A}_1$ and $\mathbf{A}_2$ and their temporal signatures $\mathbf{c}_1$ and $\mathbf{c}_2$. Similar to the low-rank decompositions, the adjacency matrices can also unravel important hidden connectivity factors, as well as be used for subsequent downstream tasks such as temporal link prediction, graph reconstruction, or anomaly detection [37, 38]. The elements of $\mathbf{c}_r$ are temporal functions whose nature can also dictate the nature of the evolution.

Besides the topology, we often have access to time-varying signals at each node. To formalize the latter, let $\underline{\mathbf{X}} \in \mathbb{R}^{N\times M\times T}$ be the signal tensor associated with the dynamic topology in $\underline{\mathbf{A}}$. The signal matrix $\underline{\mathbf{X}}_{:,:,t} \in \mathbb{R}^{N\times M}$ at time t relates to the observed topology in $\underline{\mathbf{A}}_{:,:,t}$, i.e., an M-dimensional feature vector associated with each node. Under the assumption that the evolution of signals is closely linked to the underlying topology [91, 138, 139, 159], we can use the signals in $\underline{\mathbf{X}}$ to recover the latent adjacency matrices and their temporal signatures, especially when we have missing observations in the true evolving topology in $\underline{\mathbf{A}}$. For notational convenience, we sometimes denote $\underline{\mathbf{A}}_{:,:t}$, $\underline{\mathbf{M}}_{:,:t}$, and $\underline{\mathbf{X}}_{:,:t}$ as $\mathbf{A}_t$, $\mathbf{M}_t$, and $\mathbf{X}_t$, respectively.

Problem formulation. Given the evolving topology $\underline{\mathbf{A}}$, the associated spatio-temporal signals $\underline{\mathbf{X}}$, the observation mask $\underline{\mathbf{M}}$, we want to recover the adjacency matrices of R latent graphs along with their temporal signatures $\{\mathbf{A}_r, \mathbf{c}_r\}$ by solving the general optimization problem

$$\begin{aligned} \underset{\mathbf{A},\mathbf{C}}{\text{minimize}}\ & f(\mathbf{A},\mathbf{C}) \ = l(\underline{\mathbf{A}},\underline{\mathbf{M}},\mathbf{A},\mathbf{C}) + \sum_{r=1}^{R} g(\mathbf{A}_r) + h(\mathbf{C}) + j(\underline{\mathbf{X}},\mathbf{A}) + k(\mathbf{A}) \\ \text{subject to}\ & \mathbf{A}_r \in \mathscr{S}_A,\ \mathbf{A} = [\mathbf{A}_1,\ldots,\mathbf{A}_R], \\ & \mathbf{C} \in \mathscr{S}_C,\ \mathbf{C} = [\mathbf{c}_1,\ldots,\mathbf{c}_R]. \end{aligned} \tag{6.3}$$

The function $l(\cdot)$ is the fitting term, measuring how well the latent adjacency matrices $\mathbf{A}_r$s and their temporal signatures $\mathbf{c}_r$s approximate the observed part of $\underline{\mathbf{A}}$,

functions $g(\cdot)$ and $h(\cdot)$ impose structural priors on the latent adjacency matrices $\mathbf{A}_r$s and the temporal signatures in $\mathbf{C}$, respectively. The function $j(\underline{\mathbf{X}},\mathbf{A})$ accounts for the relationship between the signal tensor $\underline{\mathbf{X}}$ and the latent adjacency matrices. The function $k(\mathbf{A})$ is a prior on the relationship between the different $\mathbf{A}_r$s, i.e., if we want the different latent adjacency matrices to be similar or dissimilar. The sets $\mathcal{S}_A$ and $\mathcal{S}_C$ impose strict constraints on the respective variables.

This formulation has a high number of degrees of freedom, with N^2 for each latent graph and T for each latent graph. This leads to a total of N^2R+TR degrees of freedom. To make the problem tractable, we need to use structural constraints as priors which we will detail in Section 6.3. Moreover, when the number of observations in $\underline{\mathbf{M}}\circ\underline{\mathbf{A}}$ is smaller than N^2R+TR, the constraint becomes even more relevant in the context of this under-determined problem.

6.3. DYNAMIC GRAPH DECOMPOSITION

Problem (6.3) is generic and can be tailored to different settings. In this section, we first make the problem more specific which involves specifying the fitting error loss function, the constraints for the latent adjacency matrices in $\mathbf{A}$ and their temporal signatures in $\mathbf{C}$. We also need to specify functions that promote desired relationships between the latent adjacency matrices $\mathbf{A}$ and ones that promote the relationship between the signals $\underline{\mathbf{X}}$, adjacency matrices $\mathbf{A}$ and temporal signatures in $\mathbf{C}$.

Fiting error. We consider for the fitting function $l(\underline{\mathbf{A}},\underline{\mathbf{M}},\mathbf{A},\mathbf{C}) = ||\underline{\mathbf{M}}\circ(\underline{\mathbf{A}}-\sum_{r=1}^{R}\mathbf{A}_r\diamond\mathbf{c}_r)||_F^2$ to measure how well the latent graphs can predict the presence and absence of the observed edges.

Constraint sets. We consider the constraint set $\mathcal{S}_A = \{\mathbf{S}\in\mathbb{R}^{N\times N}: \mathbf{S}\geq\mathbf{0}_{N\times N}, \text{tr}(\mathbf{S}) = 0, \mathbf{S}=\mathbf{S}^\top\}$, the set of non-negative, symmetric $N\times N$ matrices with zero trace. We want to estimate the adjacency matrices of the latent graphs which are non-negative and do not contain self-loops.

For the temporal signatures, we consider $\mathcal{S}_C = \{\mathbf{S}\in\mathbb{R}^{T\times R}: \mathbf{S}\geq\mathbf{0}_{T\times R}\}$, the set of non-negative $T\times R$ matrices. Positive weights in both $\mathbf{A}_r$ and $\mathbf{C}$ aid in interpretation, especially when it comes to comparing between different $\mathbf{c}_r$s. They also further restrict the degrees of freedom making Problem (6.3) more tractable.

Sparsity of latent graphs. We consider the latent graphs to be sparse. We impose sparsity on each adjacency matrix via $g(\mathbf{A}_r) = ||\mathbf{A}_r||_1$. For $\mathbf{A}_r\in\mathcal{S}_A$, this means $g(\mathbf{A}_r) = \mathbf{1}_N^\top\mathbf{A}_r\mathbf{1}_N$, where $\mathbf{1}_N$ is the column vector of N ones.

Signal smoothness. We incorporate the graph signals as a prior via the graph signal smoothness principle [83, 91]. Signal smoothness implies that nodes connected by edges have similar values and this has been commonly used as a prior for identifying topology or inference over graphs [76]. To be more specific, we assume the signals at time t, $\mathbf{X}_t$ to be smooth over the approximated topology at the same time, given by $\sum_{r=1}^{R}C_{tr}\mathbf{A}_r{}^2$. A graph signal is smooth if nodes that share edges have similar signal values. Consider the matrix $\mathbf{Z}_t\in\mathbb{R}^{N\times N}$ with its (i,j)th element $Z_{i,j} = ||[\mathbf{X}_t]_{i,:}-[\mathbf{X}_t]_{j,:}||_2^2$

[2]Notice that here we do not impose a spatiotemporal smoothness. Our model however can also accommodate that by changing the cost function $j(\cdot)$.

measuring the squared l_2 norm between the signals $[\mathbf{X}_t]_{i,:}$ and $[\mathbf{X}_t]_{i,:}$ at nodes i and j at time t. The smoothness of the graph signal $\mathbf{X}_t$ over the recovered model-driven topology is $\mathrm{QV}(\mathbf{X}_t) = \frac{1}{2}\mathrm{tr}(\hat{\underline{\mathbf{A}}}_{:,:,t}\mathbf{Z}_t)$ [91]. The signal-incorporating prior $j(\underline{\mathbf{X}},\mathbf{A})$ is the sum of the signal smoothness function for each time instant and it is calculated as

$$j(\underline{\mathbf{X}},\mathbf{A}) = \sum_{t=1}^{T} \frac{1}{2}\mathrm{tr}(\hat{\mathbf{A}}_t\mathbf{Z}_t) = \sum_{t=1}^{T}\sum_{r=1}^{R} C_{tr}\frac{\mathrm{tr}(\mathbf{A}_r\mathbf{Z}_t)}{2}. \tag{6.4}$$

This will promote new links being created from the signals in $\underline{\mathbf{X}}$ that are smooth in the original temporal network when portions of $\underline{\mathbf{A}}$ are not observed due to the mask.

Non-overlapping support. We desire that the support for the different latent graphs do not overlap so that each component contributes to the structural evolution and provides different insights. One way to promote this non-overlapping support is to penalise the $\mathbf{A}_r$s that share many edges among each other. This can be achieved by imposing a regularizer on the latent adjacency matrices the function

$$k(\mathbf{A}) = \sum_{i=1}^{R}\sum_{j=1,j\neq i}^{R} \mathrm{tr}(\mathbf{A}_i^\top\mathbf{A}_j) \tag{6.5}$$

which calculates the total pair-wise alignment of the support of the adjacency matrices. The lowest possible value of $k(\mathbf{A})$ is zero when no edge is shared between any two of the adjacency matrices.

Avoiding non-trivial solutions. The formulation in (6.3) can generate trivial solutions for some of the variables. More specifically, suppose we have a mask $\underline{\mathbf{M}}$ which hides all information across an edge for all t, i.e., $[\underline{\mathbf{M}}_{:,:,t}]_{i,j} = [\underline{\mathbf{M}}_{:,:,t}]_{j,i} = 0$ for all t. This leads to a trivial solution for the corresponding variables because of the dependence only on the functions $j(\underline{\mathbf{X}},\mathbf{A})$ and $k(\mathbf{A})$. One way to tackle this is to constrain each node in the approximation $\hat{\underline{\mathbf{A}}}_{:,:,t}$ to have a positive degree $\zeta > 0$. This is expressed through the constraint

$$\big(\sum_{r=1}^{R} C_{tr}\mathbf{A}_r\big)\mathbf{1}_N \geq \zeta\mathbf{1}_N. \tag{6.6}$$

We extend this constraint for all time instants as

$$\mathbf{A}[\mathbf{C}^\top \otimes \mathbf{1}_N] \geq \zeta\mathbf{1}_{N\times T}, \tag{6.7}$$

where $\mathbf{A} = [\mathbf{A}_1,\ldots,\mathbf{A}_r]$. This implies that each node in the reconstructed topology at time t has a degree of at least ζ.

Smooth temporal signatures. We consider each temporal signature $\mathbf{c}_r$ to change slowly over time. This helps against overfitting to the observed topology $\underline{\mathbf{M}}\circ\underline{\mathbf{A}}$, especially when fewer observations are present. A gradual change in the temporal factor would lead to smaller changes in the latent adjacency matrices. A gradual change in the temporal signature also helps in predicting the presence of an edge if it is not observed over an interval of time, given the true structural evolution is

slow. We use the squared Frobenius norm for the matrix formed by the temporal difference of the elements of each $\mathbf{c}_r$, i.e,

$$h(\mathbf{C}) = ||\mathbf{DC}||_F^2 = \sum_{r=1}^{R}\sum_{t=2}^{T}(C_{tr} - C_{t-1,r})^2 \tag{6.8}$$

where $\mathbf{D}$ is the $T \times (T-1)$ temporal difference matrix. Additionally, we consider an extra cost function for $\mathbf{C}$, namely the squared Frobenius norm. These functions aid in the convergence analysis of the optimization as we shall elaborate in Section 6.4.

With all these in place, the optimization problem (6.3) can now be written as

$$\begin{aligned}
&\underset{\{\mathbf{A}_r\},\mathbf{C}}{\text{minimize}} \quad f(\mathbf{A},\mathbf{C}) = \frac{1}{2}||\underline{\mathbf{M}} \circ (\underline{\mathbf{A}} - \sum_{r=1}^{R}\mathbf{A}_r \diamond \mathbf{c}_r)||_F^2 + \gamma\sum_{r=1}^{R}\mathbf{1}_N^\top\mathbf{A}_r\mathbf{1} + \delta\sum_{t=1}^{T}\sum_{r=1}^{R}C_{tr}\frac{\text{tr}(\mathbf{A}_r\mathbf{Z}_t)}{2} + \mu||\mathbf{DC}||_F^2 \\
&+\beta\sum_{i=1}^{R}\sum_{j=1,j\neq i}^{R}\text{tr}(\mathbf{A}_i^\top\mathbf{A}_j) + \frac{\rho}{2}||\mathbf{C}||_F^2 \\
&\text{subject to } \mathbf{A} = [\text{vec}(\mathbf{A}_1),\ldots,\text{vec}(\mathbf{A}_R)],\ \mathbf{A}_r \in \mathscr{S}_A = \{\mathbf{S} : \mathbf{S} \geq \mathbf{0}_{N\times N}, \text{tr}(\mathbf{S}) = 0, \mathbf{S} = \mathbf{S}^\top\}, \\
&\mathbf{C} \in \mathscr{S}_C = \{\mathbf{S} : \mathbf{S} \geq \mathbf{0}_{T\times R}\},\ [\mathbf{A}_1,\ldots,\mathbf{A}_r][\mathbf{C}^\top \otimes \mathbf{1}_N] \geq \zeta\mathbf{1}_{N\times T},
\end{aligned} \tag{6.9}$$

where $[\gamma,\delta,\beta,\mu,\rho]^\top$ denote the hyper-parameters.

Remark 3 *Problem* (6.9) *is a specific problem derived from the more general Problem* (6.3). *The constraints on the* $\mathbf{A}_r$*s and* $\mathbf{C}$ *can be modified to favour different scenarios. For example, a switching dynamic network can be replicated by putting an* ℓ_1 *norm penalty on each column of* $\mathbf{C}$*, i.e., each* $\mathbf{A}_r$ *is active over time windows.*

6.3.1. SOLVING THE DECOMPOSITION

The Problem (6.9) is jointly non-convex in the $\{\mathbf{A}_r\}$s and $\mathbf{C}$. However, it is separately convex in each variable when the other are fixed. This makes it easier to solve for one variable at a time via an alternating approach, i.e., we solve a sequence of convex problems in these variables until convergence. We do this by formulating $R+1$ optimization problems, where each of the first R solve for the latent adjacency matrices $\mathbf{A}_r$, whereas the last solves for the temporal signatures in $\mathbf{C}$. We solve each of them via the alternating direction method of multipliers. Thus, we perform alternating minimization between $R+1$ variables.

6.3.2. UPDATING THE $\mathbf{A}_r$S

We solve for the latent adjacency matrices $\mathbf{A}_r$s, keeping their temporal signatures $\mathbf{C}$ fixed by alternating between the different $\mathbf{A}_r$s. Consider the $N^2 \times 1$ vector $\mathbf{m}_t$ formed by vectorizing the mask at time t $\mathbf{M}_t$, i.e., $\mathbf{m}_t = \text{vec}(\mathbf{M}_t)$. Consider also the $N^2 \times T$ matrix $\mathbf{M_0}$ that collects the $\mathbf{m}_t$s along its columns, i.e.,

$$\mathbf{M_0} = [\mathbf{m}_1, \mathbf{m}_2, \ldots, \mathbf{m}_T] \in \mathbb{R}^{N^2\times T}. \tag{6.10}$$

The loss function $f(\cdot)$ w.r.t. the rth adjacency matrix $\mathbf{A}_r$ writes as

$$f(\mathbf{A}_r) = \frac{1}{2}||\mathbf{A}_r||_F^2 \sum_{t=1}^{T} C_{tr}^2 \mathbf{1}^\top \mathbf{m}_t + \text{tr}\Big(\mathbf{A}_r^\top \big[\sum_{t=1}^{T} C_{tr}\mathbf{1}^\top \mathbf{m}_t \sum_{\bar{r}=1,\bar{r}\neq r}^{R} C_{t\bar{r}}\mathbf{A}_{\bar{r}}\big]\Big)$$
$$- \text{tr}\Big(\mathbf{A}_r^\top \text{vec}^{-1}\big(\sum_{t=1}^{T} \underline{\mathbf{A}}_{:,:,t} C_{tr}\mathbf{1}^\top \mathbf{m}_t\big)\Big) + \delta \text{tr}(\mathbf{A}_r \boldsymbol{\Xi}_r) + 2\beta \text{vec}(\mathbf{A}_r)^\top \sum_{k=1,k\neq r}^{R} \text{vec}(\mathbf{A}_k) + \gamma \mathbf{1}_N^\top \mathbf{A}_r \mathbf{1}_N \tag{6.11}$$

where $\boldsymbol{\Xi}_r = \frac{1}{2}[\mathbf{Z}_1, \ldots, \mathbf{Z}_T](\mathbf{c}_r \otimes \mathbf{I}_N)$. Note the term $\sum_{t=1}^{T} C_{tr}^2 \mathbf{1}^\top \mathbf{m}_t$ that scales the Frobenius norm squared of $\mathbf{A}_r$. It is the sum across time of the product between the corresponding squared temporal coefficient C_{tr} and the number of ones in the mask $\mathbf{1}^\top \mathbf{m}_t$ which is also the number of observations available in $\underline{\mathbf{A}}$ at that time. Note also that the influence of the mask for updating the rth adjacency matrix is influenced by the corresponding temporal signature $\mathbf{c}_r$. The corresponding optimization problem thus reads as

$$\begin{aligned} &\underset{\mathbf{A}_r}{\text{minimize}}\ f(\mathbf{A}_r) \\ &\text{subject to } \mathbf{A}_r \boldsymbol{\Phi}_r + \boldsymbol{\Gamma}_r \geq \mathbf{0}_{N\times T}, \\ &\mathbf{A}_r \in \mathscr{S}_A = \{\mathbf{S} : \mathbf{S} \geq \mathbf{0}_{N\times N}, \text{tr}(\mathbf{S}) = 0, \mathbf{S} = \mathbf{S}^\top\}, \end{aligned} \tag{6.12}$$

where the inequality constraint is the same as in (6.6) but written in terms of $\mathbf{A}_r$. Additionally, we have defined the variables $\boldsymbol{\Phi}_r = \mathbf{c}_r^\top \otimes \mathbf{1}_N$ and $\boldsymbol{\Gamma}_r = \sum_{\bar{r},\bar{r}\neq r} \mathbf{A}_{\bar{r}}(\mathbf{c}_{\bar{r}}^\top \otimes \mathbf{1}_N) - \zeta \mathbf{1}_{N\times T}$. We solve Problem (6.12) via the Alternating Direction Method of Multipliers (ADMM) [174]. The formulation is

$$\begin{aligned} &\underset{\mathbf{A}_r,\mathbf{P}}{\text{minimize}}\ f(\mathbf{A}_r) \\ &\text{subject to: } \mathbf{A}_r \in \mathscr{S}_A,\ \mathbf{A}_r \boldsymbol{\Phi}_r + \boldsymbol{\Gamma}_r = \mathbf{P},\ \mathbf{P} \geq \mathbf{0}_{N\times T}, \end{aligned} \tag{6.13}$$

where the auxiliary variable $\mathbf{P}$ which is introduced as another inequality constraint, to cast the problem in the ADMM format. Then the augmented Lagrangian in (6.13) reads as

$$L(\mathbf{A}_r, \mathbf{P}, \boldsymbol{\Lambda}_r) = f(\mathbf{A}_r) + \text{tr}(\boldsymbol{\Lambda}_r(\mathbf{A}_r \boldsymbol{\Phi}_r + \boldsymbol{\Gamma}_r - \mathbf{P})) + \frac{\lambda}{2}||\mathbf{A}_r \boldsymbol{\Phi}_r + \boldsymbol{\Gamma}_r - \mathbf{P}||_F^2, \tag{6.14}$$

where $\lambda > 0$. The updates in the primal variables $\mathbf{A}_r$ at iteration $k+1$ are given as

$$\mathbf{A}_r^{k+1} = \underset{\mathscr{S}_A}{\Pi}\Big(\mathbf{A}_r^k - \lambda \nabla_{\mathbf{A}_r} L(\mathbf{A}_r^k)\Big), \tag{6.15}$$

which is a gradient step followed by a projection onto the set $\mathscr{S}_A$. The expression for the gradient $\nabla_{\mathbf{A}_r} L(\mathbf{A}_r)$ is detailed in (D.1) in Appendix D.3. For $\mathbf{P}$, we note that the Lagrangian is convex w.r.t itself and we can readily obtain a closed-form solution, followed by a projection onto the non-negative orthant. The remaining updates for the primal variable $\mathbf{P}$ and the dual variables $\boldsymbol{\Lambda}_r$ are thus

$$\mathbf{P}^{k+1} = [\frac{1}{\lambda}\boldsymbol{\Lambda}_r^{k\top} + \mathbf{A}_r^{k+1}\boldsymbol{\Phi}_r + \boldsymbol{\Gamma}_r]_+ \tag{6.16}$$

$$\mathbf{\Lambda}_r^{k+1} = \mathbf{\Lambda}_r^k + \lambda(\mathbf{A}_r^{k+1}\mathbf{\Phi}_r + \mathbf{\Gamma}_r - \mathbf{P}^{k+1})^\top, \tag{6.17}$$

where $[\cdot]_+$ is the element-wise operator which clips all negative values to zero. The two steps are repeated until a stopping criterion is met. This whole procedure is repeated alternatively for each latent adjacency matrix $\mathbf{A}_r$.

6.3.3. UPDATING C

In this step, we update $\mathbf{C}$ when all the latent $\mathbf{A}_r$ are fixed. As we did with the adjacency matrices, we first write down the loss function w.r.t $\mathbf{C}$. To do so, we define

$$\mathbf{A}_{\text{vec}} = [\text{vec}(\underline{\mathbf{A}}_{:,:,1}), \text{vec}(\underline{\mathbf{A}}_{:,:,2}), \ldots, \text{vec}(\underline{\mathbf{A}}_{:,:,T})] \in \mathbb{R}^{N^2 \times T}. \tag{6.18}$$

$$\mathbf{A_0} = [\text{vec}(\mathbf{A}_1), \text{vec}(\mathbf{A}_2), \ldots, \text{vec}(\mathbf{A}_R)] \in \mathbb{R}^{N^2 \times R}. \tag{6.19}$$

The loss in $\mathbf{C}$ then writes as

$$f(\mathbf{C}) = \frac{1}{2}||\mathbf{M}_0 \circ (\mathbf{A}_{\text{vec}} - \mathbf{A_0}\mathbf{C}^\top)||_F^2 + \delta \text{tr}(\mathbf{C}^\top \bar{\mathbf{Z}}_T \mathbf{A_0}) + \frac{\mu}{2}||\mathbf{DC}||_F^2 + \frac{\rho}{2}||\mathbf{C}||_F^2, \tag{6.20}$$

where $\text{tr}(\mathbf{C}^\top \bar{\mathbf{Z}}_T \mathbf{A_0})$ is the signal smoothness prior (6.4) re-written to show the dependence on $\mathbf{C}$, and $\bar{\mathbf{Z}}_T \in \mathbb{R}^{T \times N^2}$ is a matrix with row k being the vectorized form of the matrix $\mathbf{Z}_k^\top$, i.e., $[\bar{\mathbf{Z}}_T]_{:,k} = \text{vec}(\mathbf{Z}_k^\top)^\top$. The inequality constraint involving both $\mathbf{A}$ and $\mathbf{C}$ in Problem (6.9) translates into $\mathbf{C}\Upsilon^\top - \zeta \mathbf{1}_{T \times N} \geq \mathbf{0}_{T \times N}$ where $\Upsilon = [\mathbf{A}_1, \ldots, \mathbf{A}_R](\mathbf{I}_R \otimes \mathbf{1}_N)$. Thus, the optimization problem involving $\mathbf{C}$ is

$$\begin{aligned} \underset{\mathbf{C}}{\text{minimize}} \quad & f(\mathbf{C}) \\ \text{subject to:} \quad & \mathbf{C}\Upsilon^\top - \zeta \mathbf{1}_{T \times N} \geq \mathbf{0}_{T \times N}, \\ & \mathbf{C} \in \mathscr{S}_C = \{\mathbf{S} : \mathbf{S} \geq \mathbf{0}_{T \times R}\}. \end{aligned} \tag{6.21}$$

To deal with the inequality constraint, we formulate another ADMM problem as

$$\begin{aligned} \underset{\mathbf{C}}{\text{minimize}} \quad & f(\mathbf{C}) \\ \text{subject to:} \quad & \mathbf{C} \in \mathscr{S}_C, \mathbf{C}\Upsilon^\top - \zeta \mathbf{1}_{T \times N} = \mathbf{Q}, \\ & \mathbf{C} \geq \mathbf{0}_{T \times N}. \end{aligned} \tag{6.22}$$

Then, the augmented Lagrangian becomes

$$L(\mathbf{C}, \mathbf{Q}, \mathbf{\Lambda}) = f(\mathbf{C}) + \text{tr}\left(\mathbf{\Lambda}(\mathbf{C}\Upsilon^\top - \zeta \mathbf{1}_{T \times N} - \mathbf{Q})\right) + \frac{\lambda_c}{2}||\mathbf{C}\Upsilon^\top - \zeta \mathbf{1}_{T \times N} - \mathbf{Q}||_F^2. \tag{6.23}$$

To update $\mathbf{C}$, we take a gradient step followed by a projection onto $\mathscr{S}_C$, i.e.,

$$\mathbf{C}^{k+1} = \prod_{\mathscr{S}_C}\left(\mathbf{C}^k - \lambda \nabla_{\mathbf{C}} L(\mathbf{C}^k)\right). \tag{6.24}$$

The derivative of the augmented Lagrangian w.r.t. $\mathbf{C}$ is detailed in Equation (D.4) in Appendix D.3. The other two updates are also similar to those made for each $\mathbf{A}_r$ and are respectively

$$\mathbf{Q}^{k+1} = [\frac{1}{\lambda_c}\mathbf{\Lambda}^{k\top} + (\mathbf{C}^{k+1}\mathbf{\Upsilon})^\top - \zeta \mathbf{1}_{TN})]_+, \tag{6.25}$$

$$\mathbf{\Lambda}^{k+1} = \mathbf{\Lambda}^{k} + \lambda_c(\mathbf{C}^{k+1}\Upsilon^{\top} - \zeta\mathbf{1}_{TN} - \mathbf{Q}^{k+1})^{\top}. \tag{6.26}$$

Algorithm 6 summarizes the dynamic graph decomposition.

6.4. ALGORITHM ANALYSIS

6.4.1. COMPLEXITY ANALYSIS.

In this section, we analyse the proposed DGTD algorithm first in terms of its computational complexity and then the convergence behaviour of the alternating approach.

Parameter complexity. The parameters to be estimated are $\left(\frac{N-1}{2}+2T\right)NR+(R+2N)T$: where (i) $\left(\frac{N-1}{2}+2T\right)NR$ are the parameters of the R matrices $\mathbf{A}_r$ which are symmetric with zero diagonals and the additional variables $\mathbf{P}$ and $\mathbf{\Lambda}_r$ and (ii) $(R+2N)T$ parameters of the R temporal signatures $\mathbf{C}$ and the additional variables $\mathbf{Q}$ and $\mathbf{\Lambda}$.

Computational complexity. Let there be K alternating iterations between $\mathbf{A}$ and $\mathbf{C}$. The computational complexity is of order $\mathcal{O}((N^3R+N^2TR+R^2N^2+T^2R+NTR)K)$ which can be broken down as follows. The computations for estimating $\mathbf{A}_r$ are governed by the gradient $\nabla_{\mathbf{A}_r}L(\mathbf{A}_r)$ in (6.15) and the updates of the auxiliary variables $\mathbf{P}$ in (6.16) and $\mathbf{\Lambda}_r$ in (6.17). We update the R matrices $\mathbf{A}_r$, $\mathbf{P}$, and $\mathbf{\Lambda}_r$ for K steps leading to a total computational cost of $\mathcal{O}((N^3+N^2T)R)$. Computing $\mathbf{C}$ has a cost of order $\mathcal{O}((R^2T^2+T^2R))$ and each update of the auxiliary variables $\mathbf{Q}$ and $\mathbf{\Lambda}$ has a complexity of order $\mathcal{O}(NTR)$. The matrices $\mathbf{C}$, $\mathbf{Q}$, and $\mathbf{\Lambda}$ are updated for K steps leading to a total cost of $\mathcal{O}(RT^2K+T^2K+NTK)R$.

Remark 4 *In the updates for $\mathbf{A}_r$ and $\mathbf{C}$ [ef. Equations (6.15) and (6.24)], we can also obtain a closed-form solution for each, prior to projection. However, this closed-form solution comprises an inverse that is computationally heavy with a complexity of order $\mathcal{O}(N^3)$ for each $\mathbf{A}_r$ and for each iteration in the ADMM, and $\mathcal{O}(R^3T^3)$ for each $\mathbf{C}$ update, given we need to vectorize to obtain the optimal value. Moreover, projections on to $\mathcal{S}_A$ and $\mathcal{S}_C$ after obtaining the closed-form solution can lead to undesirable projections, especially if the solution lies entirely outside the feasible set. For example, if a column in $\mathbf{C}$ has only negative elements in the optimal solution (and thus, zero after projection), it can strongly affect the corresponding $\mathbf{A}_r$ update, thereby affecting the convergence of Algorithm 6. A gradient step prior to projection is computationally less intense, offers more gradual change due to the convex nature of the functions, and works well in practice.*

6.4.2. CONVERGENCE ANALYSIS

In this section, we comment on the convergence of the proposed ADMM-based approach in Algorithm 6 to a stationary point w.r.t the $\mathbf{A}_r$s and $\mathbf{C}$. We have one alternating optimization procedure involving $R+1$ updates for the R $\mathbf{A}_r$s and one involving $\mathbf{C}$.

Assumption 8 *For all r, $\sum_{t=1}^{T} C_{tr}^2\mathbf{1}^{\top}\mathbf{m}_t > 0$, i.e., the sum of the product of the squared temporal signature coefficient with the number of available observations across time is greater than zero.*

Algorithm 6 Dynamic Graph Topology Decomposition (DGD)

Input: Tensor $\underline{\mathbf{A}} \in \mathbb{R}^{N\times N\times T}$; $\underline{\mathbf{X}} \in \mathbb{R}^{N\times M\times T}$; R, γ, β, μ, δ.
Output: Latent graph adjacency matrices $\{\mathbf{A}_r\}_{r=1}^{R}$,
temporal signatures $\mathbf{C}$
Initialization: Initialize $\{\mathbf{A}_r\}_{r=1}^{R}$, $\mathbf{C}$ as element-wise uniformly at-random over $[0,1]$
For $k = 1:K$
Block Alternating updates over $\mathbf{A}_r$ and $\mathbf{C}$
A Update:
For $r = 1:R$
Initialize $\mathbf{P}$, $\mathbf{\Lambda}$ using zero-mean
unit variance Gaussian random variables
Repeat until convergence (ADMM for $\mathbf{A}_r$)
Update $\mathbf{A}_r$ using (6.15)
Update $\mathbf{P}$ using (6.16)
Update $\mathbf{\Lambda}$ using (6.17)
end For
C Update:
Initialize $\mathbf{Q}$, $\mathbf{\Lambda}$ using zero-mean
unit-variance Gaussian random variables
Repeat until convergence (ADMM for $\mathbf{C}$)
Update $\mathbf{C}$ using (6.24)
Update $\mathbf{Q}$ using (6.25)
Update $\mathbf{\Lambda}$ using (6.26)
end For
end For

This condition holds as long as the temporal signature of the rth latent graph $\mathbf{c}_r \neq \mathbf{0}_T$ across time and we have at least one available observation at each time instant. In practice, we make several observations at each time step, i.e., $\mathbf{1}^\top \mathbf{m}_t > 0$ is easy to satisfy. A non-zero temporal signature $\mathbf{c}_r$ indicates that the corresponding latent adjacency matrix $\mathbf{A}_r$ is of importance when it comes to expressing the structural evolution.

Convergence. Since our decomposition approach is a block-wise minimization, we will consider the block-coordinate minimizer as shown in [175, 176]. Let χ be the set comprising all the latent variables $\{\mathbf{A}_1,\ldots,\mathbf{A}_R,\mathbf{C}\}$. The variables $\{\bar{\mathbf{A}}_1,\ldots,\bar{\mathbf{A}}_r,\bar{\mathbf{C}}\}$ are a block coordinate minimizer of $f(\cdot)$ if for each r, we have

$$\begin{gathered} f(\bar{\mathbf{A}}_1,\ldots,\bar{\mathbf{A}}_r,\ldots,\bar{\mathbf{A}}_R,\bar{\mathbf{C}}) \leq f(\bar{\mathbf{A}}_1,\ldots,\mathbf{A}_r,\ldots,\bar{\mathbf{A}}_R,\bar{\mathbf{C}}) \\ \text{for all } \mathbf{A}_r \in \mathscr{S}_{\mathbf{A}} \text{ and } \{\bar{\mathbf{A}}_1,\ldots,\mathbf{A}_r,\ldots,\bar{\mathbf{A}}_R,\bar{\mathbf{C}}\} \in \chi \end{gathered} \tag{6.27}$$

$$\begin{gathered} f(\bar{\mathbf{A}}_1,\ldots,\bar{\mathbf{A}}_r,\ldots,\bar{\mathbf{A}}_R,\bar{\mathbf{C}}) \leq f(\bar{\mathbf{A}}_1,\ldots,\bar{\mathbf{A}}_R,\mathbf{C}) \\ \text{for all } \mathbf{C} \in \mathscr{S}_{\mathbf{C}} \text{ and } \{\bar{\mathbf{A}}_1,\ldots,\bar{\mathbf{A}}_R,\bar{\mathbf{C}}\} \in \chi \end{gathered} \tag{6.28}$$

Proposition 7 *Given the true dynamic topology tensor* $\underline{\mathbf{A}}$*, the associated spatio-temporal graph signals* $\underline{\mathbf{X}}$*, the observation mask tensor* $\underline{\mathbf{M}}$*, and Assumption 8, the sequence* $\{\mathbf{A}_1^k,\ldots,\mathbf{A}_r^k,\mathbf{C}^k\}$ *generated by the dynamic graph decomposition (DGD) in Algorithm 6 converges to a stationary point, which is also known as the block coordinate minimizer of Problem* (6.9).

Proof. Check Appendix D.2. □

6.5. NUMERICAL RESULTS

In this section, we thoroughly corroborate the proposed approach with the experimental results on a synthetic controlled setting and real dataset. We start with a description of the data sets, evaluation metrics, and experimental details. Next, we analyze the properties of the proposed DGD method through a series of experiments. Finally, we compare it with alternative methods, focusing on the reconstruction error for the temporal network over a range of partial observations.

6.5.1. EXPERIMENTAL SETUP

We consider a synthetic experiment on stochastic block model graphs and three real datasets. The details of these datasets are summarized in Table 6.1.

1. **Synthetic dynamic network (SwDyn).** We create a dynamic network where we transition from a stochastic block model (SBM) graph with two communities to one with four communities. The observed adjacency matrix at time t is a linear combination of the two SBM graphs. Thus, we have $R=2$ latent graphs. We consider the temporal signature $\mathbf{c}_1$ of $\mathbf{A}_1$ as a vector with values from one to zero reducing linearly over the fifty time instances. As a result, $\mathbf{c}_2 = \mathbf{1}_{50} - \mathbf{c}_1$ has values from zero to one, contributing to the emergence of $\mathbf{A}_2$. For each time step t, the signal feature matrix $\mathbf{X}_t$ is computed from the Laplacian matrix of $\underline{\mathbf{A}}_{:,:,t}$ similar to [81]. We generate a total of $M=1000$ smooth graph signals associated with each one of the temporal graphs.

2. **Sea surface temperature (SeaSurf).** We consider surface temperature measurements across points on the Pacific Ocean [156]. It consists of 1728 temporal measurements spread over 600 months at $N=100$ points on the ocean. The temporal horizon was divided into $T=8$ equally-sized windows. Since there is no ground-truth graph, we estimate one for each time window. This estimation is based on half of the available temporal measurements (those corresponding to odd time instants, $\tau=\{1,3,5,...\}$), selecting the top $4N$ edges where the associated signal is smooth. The remaining half of the temporal measurements (those from even time instants, $\tau=\{2,4,6,...\}$) are used as input for the numerical experiments, resulting in $M=108$ nodal measurements for each temporal graph.

3. **US temperature (USTemp).** We use the NOAA data set from [127], which consists of 8730 temporal graph signals indicating the temperature variation

across $N = 109$ stations. We divide the time into $T = 15$ equally-sized windows. Since a ground-truth graph is not available, we generate one for each window following similar steps to the SeaSurf data set. Specifically, we estimate the graph using half of the temporal measurements (those corresponding to odd time instants, $\tau = 1,3,5,...$), selecting the top $5N$ edges where the associated signal is smooth. The remaining half of the temporal measurements (those corresponding to even time instants, $\tau = 2,4,6,...$) are used as input for the numerical experiments, resulting in $M = 291$ nodal measurements for each temporal graph.

4. **Contact.** This dataset was collected during the ACM Hypertext 2009 conference, where nodes represent $N = 113$ attendees, and edges connect them based on face-to-face proximity[3]. The dynamic network captures interactions between attendees over two and a half days, resulting in a total of 20,800 interactions. The signal at each time interaction is the sum of all prior interactions up to that point normalized by the maximum number of interactions observed until that time. After constructing the network signals, we divide them into $T = 50$ time windows, yielding $M = 416$ signals for each temporal graph. Since there is no ground-truth graph, we construct one for each time window by considering all interactions up to that point and selecting the top $3N$ edges corresponding to the nodes with the most interactions.

Table 6.1.: Properties of all data sets. The columns represent the following: Number of nodes (N), average number of edges (Av. E.) per graph in the temporal network, number of nodal samples (M), number of temporal graphs (T), temporal variation measured as the normalized squared Frobenius norm (T-Var.) between consecutive adjacency matrices along with the standard deviation (Std), and the normalized total variation for each node obtained from the signals (QV).

Data Set	N	Av. E.	M	T	T-Var.(Std)	QV
SwDyn	40	130	1000	50	0.17(0.08)	0.08
SeaSurf	100	400	108	8	0.12(0.02)	2.59
USTemp	109	545	291	15	0.47(0.09)	13.32
Contact	113	157	416	50	0.12(0.21)	2.94

Evaluation. We evaluate the performance w.r.t estimating the hidden dynamic network over the unobserved part of the true topology $\underline{\mathbf{A}}$. Consider $\underline{\mathbf{M}}_{un} \in \{0,1\}^{N\times N\times T}$ as the mask corresponding to the unobserved edges and non-edges. For our experiments, $[\underline{\mathbf{M}}_{un}]_{i,j,t} = 1 - M_{i,j,t}$. We consider a mask $\underline{\mathbf{M}}$ which masks both existing and non-existing edges with uniform probability across the tensor $\underline{\mathbf{A}}$. We are interested in two types of error metrics:

[3] http://www.sociopatterns.org/datasets/hypertext-2009-dynamic-contact-network/

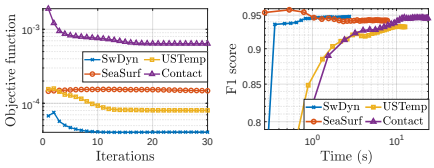


Figure 6.2.: Convergence of the proposed approach for different data sets with $R = 10$ and 90% of the temporal network as observed in terms of (a) the normalized objective function as the number of iterations increases and (b) the obtained F1 score together with the elapsed time for convergence.

1. The relative error (RE) given as

$$\mathsf{RE} = \frac{||\underline{\mathbf{M}}_{un} \circ (\hat{\underline{\mathbf{A}}} - \underline{\mathbf{A}})||_F^2}{||\underline{\mathbf{M}}_{un} \circ \underline{\mathbf{A}}||_F^2} \tag{6.29}$$

2. The F1 score measures the ability to predict the unobserved topology in $\underline{\mathbf{A}}$. It is evaluated as

$$\mathsf{F1} = \frac{2\mathsf{PrRe}}{\mathsf{Pr} + \mathsf{Re}} \tag{6.30}$$

where Pr (precision) indicates the percentage of estimated edges that are edges of the unobserved dynamic network and Re (recall) is the percentage of existing edges that were correctly estimated.

6.5.2. METHOD ANALYSIS

In this section we investigate the performance of the proposed DGD method in terms of convergence, the role of the latent factors and its ability to recover the dynamic topology.

Convergence. We assess the convergence of the proposed approach by fixing the number of latent graphs $R = 10$ and using 90% of the temporal network topology in $\underline{\mathbf{A}}$ for all data sets. Figure 6.2.a illustrates the convergence behavior, while Figure 6.2.b shows the F1 score as a function of elapsed time. Our approach converges within approximately 20 iterations across all data sets, except for USTemp. Figure 6.2.b further indicates that achieving convergence in the objective function correlates with effective recovery of the ground-truth temporal network structure. The convergence is slower in larger networks.

Effect of rank and observed topology. Next, we analyse the proposed approach across two distinct setups:

Role of R. Figure 6.3.a illustrates the variation in RE w.r.t. the number of latent graphs R. We utilize 100% of the graph signals and consider 90% of the temporal network observed. Increasing the number of latent graphs improves the approximation of the temporal network across all data-sets. For the SwDyn data set, a substantial improvement is observed when increasing R from 1 to 2, after which the error stabilizes. This suggests that two latent graphs are sufficient to capture the

dynamics of the temporal network, and further increases in R do not significantly reduce the estimation error. This makes sense because the generated topology has two latent SBM graphs. It should be noted, however, that identifying the optimal R remains a non-trivial task. For a fixed R, the relative errors for different data-sets can be attributed to their characteristics. For USTemp data, it is a combination of the variation in the true topology $\underline{\mathbf{A}}$ and the higher QV, both varying considerably over the $T = 15$ time stamps.

Role of observed data. Figure 6.3.b presents the RE as a function of the percentage of the observed temporal network for various data sets, using $R = 10$ and 100% of the graph signals. This shows that reconstructing the original tensor from a limited portion of the observed network is more challenging, resulting in higher RE. Consistent with previous findings, the estimated temporal network is significantly affected by both the proportion of the observed topology and the inherent characteristics of each network, such as temporal variations in signal smoothness and structure. This is evident from the higher estimation errors observed for USTemp and Contact data sets.

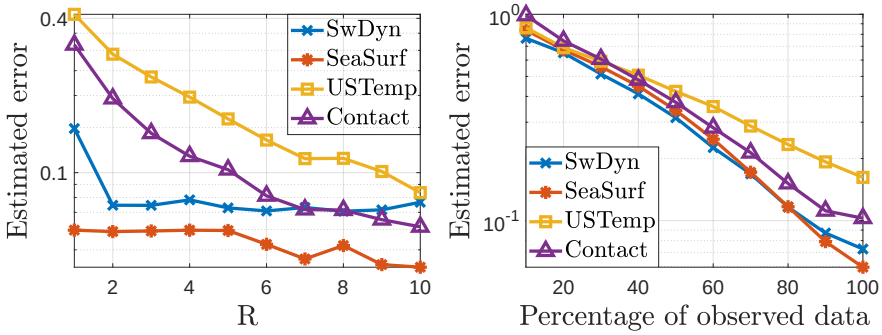


Figure 6.3.: Numerical evaluation of the proposed approach: (left) Relative error RE w.r.t. the number of modes R; (right) RE w.r.t the percentage of the observed temporal tensor $\underline{\mathbf{A}}$.

Recovered components. Next, we analyze the latent adjacency matrices $\mathbf{A}_r$ and their corresponding temporal signatures $\mathbf{c}_r$ for the USTemp and Contact data sets for $R = 3$. Figure 6.6 illustrates the temporal signatures $\mathbf{c}_1, \mathbf{c}_2$, and $\mathbf{c}_3$ for the USTemp and Constact datasets. For USTemp, each signature is larger than the others over certain time intervals and changes smoothly over the 15 time samples. We see a similar trend for the Contact data with slightly more fluctuations in time. Figures 6.4 and 6.5 illustrate the three corresponding latent graphs. Each figure comprises two rows: the first row displays the support for each latent adjacency matrix, while the second row shows the corresponding graph representation. The latent graphs are individually and collectively significant both in terms of representing the true $\underline{\mathbf{A}}$ as well as reconstructing missing observations in $\underline{\mathbf{M}}_{un} \circ \underline{\mathbf{A}}$. We also see that each latent graph (and therefore its adjacency matrix) differs from the rest. They might share some edges but there are plenty of different edges as well. For USTemp, the first and third latent graphs have a similar structure which can be due to some sort of

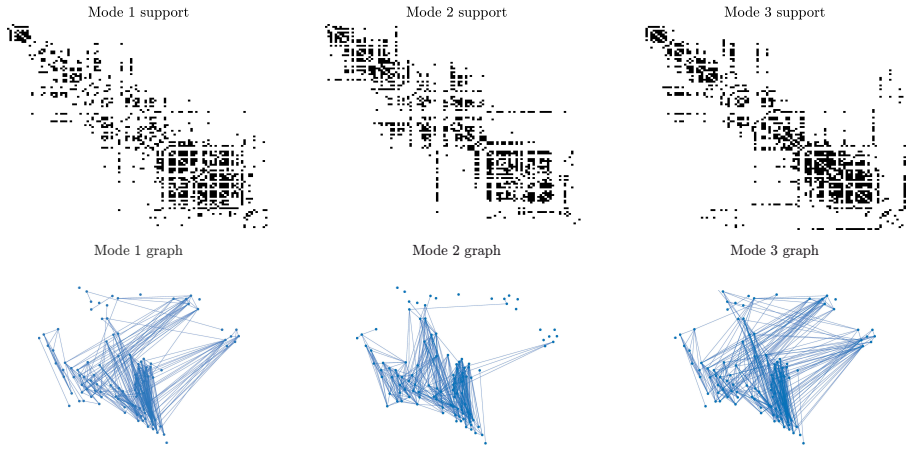


Figure 6.4.: Visualization of the components obtained from the proposed approach on USTemp data set for $R = 3$: (top row) the recovered adjacency modes; (bottom row)) shows the associated graph visualizations.

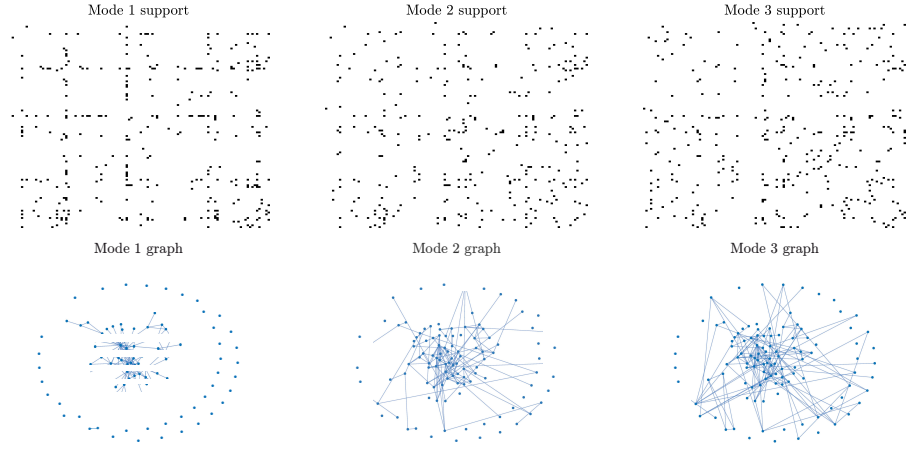


Figure 6.5.: Visualization of the components obtained from the proposed approach on Contact data for $R = 3$: (top row) the recovered adjacency modes; (bottom row)) shows the associated graph visualizations.

periodicity in the measurements.

Support recovery potential of recovered components. Next, we assess the contribution of individual estimated latent graphs towards the reconstruction of the temporal network. Specifically, we evaluate how accurately each latent graph captures the network's unobserved structure via the relative reconstruction error

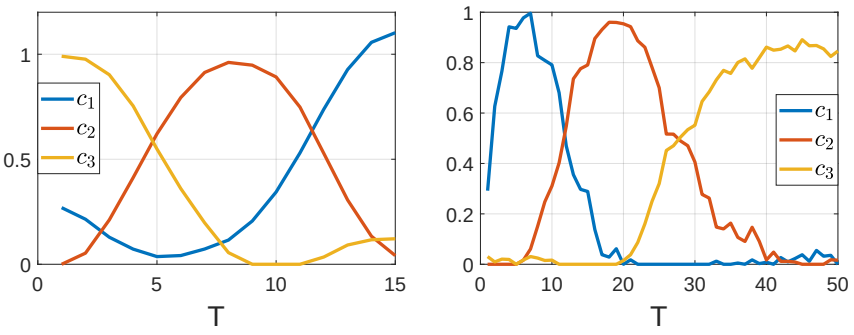


Figure 6.6.: Temporal signatures associated with each one of the modes for (left) USTemp data[cf. 6.4] and (right) Contact data [cf. 6.5].

RE. Figure 6.8 shows the RE across the different data sets. For each data set, we consider $R = 3$ and examine the reconstruction quality for each latent graph with $\mathbf{A}_1$, $\mathbf{A}_2$, $\mathbf{A}_3$as well as the reconstruction achieved by combining all latent adjacency matrices. Individual graphs alone do not adequately capture the temporal network's structure, as evidenced by higher RE scores for each latent graph by itself. However, combining the graphs significantly improves the RE scores for all data sets. This demonstrates that while each latent graph contributes to explaining a portion of the temporal network $\underline{\mathbf{A}}$ and no single latent graph is redundant. One reason for this can be seen from the corresponding temporal signatures in Figure 5. Since each latent graph has a temporal signature which has high values over a slice of the temporal window, this graph would have lower error in recovering missing observations over this time period. And since the different temporal signatures peak at different times, the corresponding graphs recover different parts of $\underline{\mathbf{M}}_{un} \circ \underline{\mathbf{A}}$ with high accuracy. This can also be explained in part by the hyper-parameter β which penalises shared edges across the latent graphs. Higher values of β can also cause edges to be separated across different $\mathbf{A}_r$s. Even if they have similar temporal signatures, they would still be able to reconstruct different aspects of $\underline{\mathbf{A}}$. Overall, this experiment highlights the importance of considering all latent graphs collectively to achieve a comprehensive reconstruction of the temporal network. The corresponding plots for F1 are shown in Appendix D.4.

6.5.3. COMPARISON

We assess the effectiveness of our approach in reconstructing the unobserved temporal network by considering various scenarios where parts of the network are unobserved. Specifically, we compare with the following approaches:

1. **Unconstrained solution (UNC).** This is the unconstrained solution to Problem (6.9) where we drop both the graph and temporal smoothness constraints on the latent adjacency matrices $\mathbf{A}_0$ and the temporal signatures $\mathbf{C}$, respectively.

The solution is obtained by minimizing

$$\underset{\mathbf{A}_0,\mathbf{C}}{\text{minimize}} \; ||\mathbf{A}_{\text{vec}} - \mathbf{A}_0\mathbf{C}^\top||_F^2 \tag{6.31}$$

where $\mathbf{A}_{\text{vec}}$ and $\mathbf{A}_0$ are defined in (6.18) and (6.19), respectively.

2. **CPD.** This is the canonical polyadic decomposition [71] which approximates $\underline{\mathbf{A}}$ via a sum of three-dimensional vector outer products. We do not impose any graph constraints on the CPD, so the components may not be interpretable like ours.

3. **BTD.** The block term decomposition (BTD) approximates $\underline{\mathbf{A}}$ as a sum of outer product of matrices [72]. Thus, we expect the BTD to represent more structure than the rank one terms in the CPD. We consider the $\{L, L, 1\}$ type BTD where each block term comprises the outer product of two matrices of rank L and a vector of size T. We use the code in [4].

4. **No signal dynamic graph decomposition (NSDGD).** This is the proposed approach without incorporating the graph signals, i.e, $\delta = 0$. This allows us to see the effect of including graph signals into the design.

5. **Smooth graph learning (SGL).** For this baseline, we use the tensor graph signals in $\underline{\mathbf{X}}$ directly to estimate the topology. We estimate the topology at each t from the corresponding $\mathbf{X}_t$ following [81].

6. **SICA.** This approach presented in [164] approximate the tensor $\underline{\mathbf{A}}$ using a spatial independent component analysis. Thus, it outputs latent graphs which are independent in a statistical sense, without relying on graph-based constraints.

For a fair comparison between the different approaches, we ensure that CPD, BTD, NSDGD, and SICA have a similar number of parameters as the proposed approach. This results in a higher number of terms in the CPD and BTD approaches, which may further complicate the ability to interpret the decomposition.

Figure 6.8 presents the RE of each approach when varying the percentage of observed topology in the temporal network. The results indicate that UNC tends to overfit to the observed part of the network and fails to reconstruct the unobserved portions of the temporal network. DGD performs better than the low-rank tensor decomposition-based approaches (CPD and BTD) for all data sets for all percentages of observed topology. The gap in performance varies, from a high (USTemp) to low (SwDyn). This corroborates our hypothesis that low-rank tensor decompositions are not very informative or representative of the structural evolution. BTD yields slightly better RE scores than CPD.

Regarding the results associated with the SICA approach, we observe a slightly improved performance in terms of error compared to low-rank tensor decomposition methods. A notable outcome, as evidenced by the results, is that the SICA approach

[4] http://dimitri.nion.free.fr/Codes/Tensor_Decompositions.html

yields higher errors compared to NSDGD and DGS when the percentage of available data is large, while the opposite trend is observed with a smaller percentage of available data. This effect stems from the assumption in the SICA approach of estimating independent components, which may be overly restrictive for capturing the structure of the network when a substantial amount of information is available, but becomes beneficial when data availability is limited.

The DGD and NSDGD demonstrate superior performance compared to other methods. These approaches employ a more sophisticated temporal network structure, by incorporating the latent adjacency matrices and temporal signatures. Notably, DGD achieves the best performance, primarily due to its utilization of additional node signals. This additional information enhances the reconstruction accuracy, particularly in scenarios where a significant portion of the temporal network is unobserved. The gap between DGD and NSDGD narrows as we observe more of the structural evolution, showcasing the need to use the signals as a prior with limited observations. However, we note that SGL outperforms all approaches for SwDyn data when we observe 10 to 40 percent of $\underline{\mathbf{A}}$, i.e, just using the signals with signal smoothness prior recovers more about the structural evolution. This makes sense as we generate smooth signals from the topology. This also suggests that a low percentage of observations goes against recovering the hidden observations when combined with signals. For the other data sets, SGL struggles to recover the structural evolution.

In summary, the proposed approach shows advantages over existing methods by leveraging additional node signals information and employing it to improve the reconstruction error of temporal network in scenarios with a large percentage of unobserved part of the temporal network.

6.6. CONCLUSION

We propose a novel way to represent and interpret dynamic networks. We do this by modelling the structural evolution as composed of weighted combinations of latent graphs along with temporal signatures which scale them across time. We also utilize the presence of spatio-temporal signals as a smoothness prior and estimate the latent graphs along with their temporal signatures via alternating minimization. We show that the solution converges to a block coordinate minimizer. The recovered components can represent important underlying groups of connections. We show empirically that each individual graph contributes towards representing the dynamics, as shown by their ability to recover the missing network. Numerical results demonstrate a superior performance in terms of recovering missing parts of the network compared to topology-agnostic low-rank tensor decompositions and methods based on graph signal smoothness. Future works include an online extension of the decomposition, where we do not have access to the entire sequence of topologies at once, and thus, predict and update the components over time. Another possibility is to apply the decomposition to expanding graphs, with nodes joining the existing graph over time.

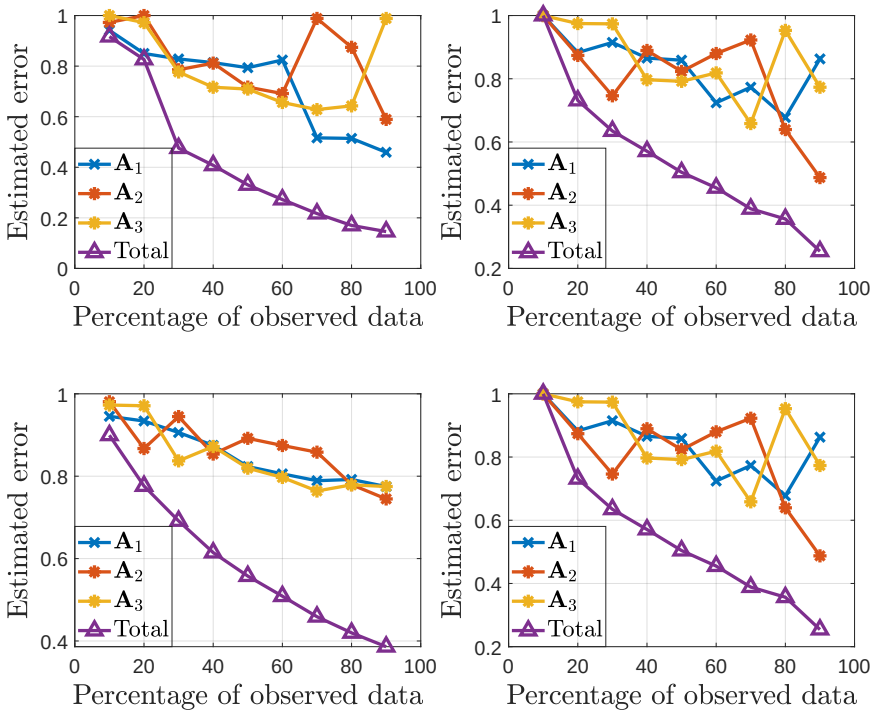


Figure 6.7.: Reconstruction potential of $R = 3$ individual modes (blue, orange and yellow) for each data set compared to that of the their model-based combination (purple). Here we show RE for (left) SwSyn, (centre-left) SeaSurf, (centre-right) USTemp, and (right) Contact data.

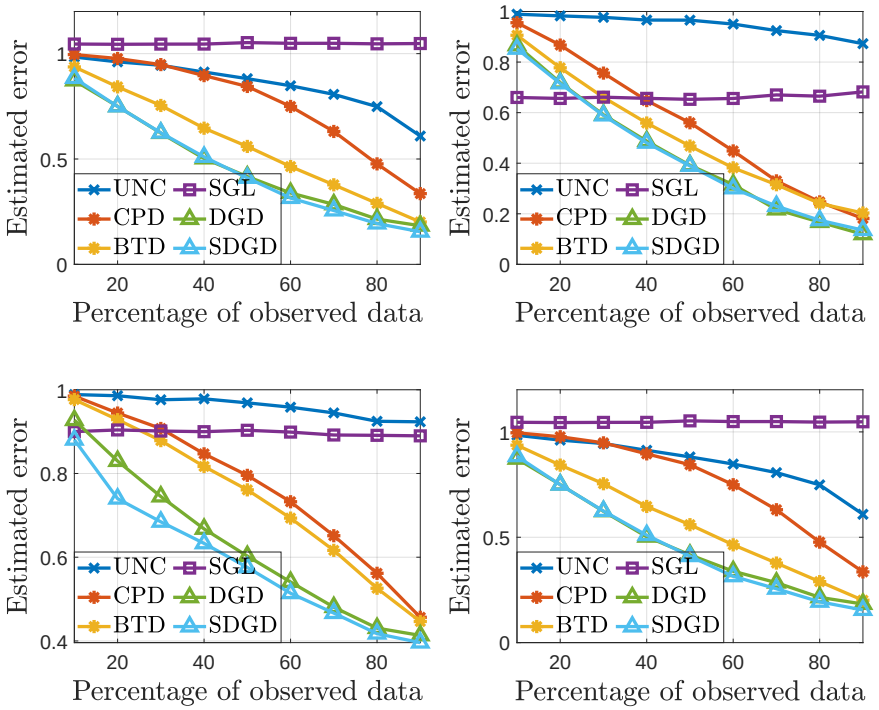


Figure 6.8.: Relative error (RE) for different approaches as a function of the percentage of observed adjacency tensor $\underline{\mathbf{A}}$ for (left) SwDyn, (centre-left) SeaSurf, (centre-right) USTemp, and (right) Contact data sets, respectively.

7

CONCLUDING REMARKS

In Chapter 1, we set out two aims for this dissertation. The first was to propose, design and analyse tools for processing signals over dynamic graphs, especially when we have new nodes in an expanding graph scenario with unknown connectivity information. The second was to propose representation of changing edge supports in dynamic graphs in a way which allows for their interpretation and recovery, opposed to traditional approaches.

7.1. ANSWERS TO RESEARCH QUESTIONS

RQ1: How do we process signals over expanding graphs, assuming both known and unknown connectivity?

We answer this question in two ways: In Chapter 3, we focus on the effects of stochastic attachment in the face of unknown connectivity; in Chapter 4 we focus exclusively on filter design.

In Chapter 3, we show that it is possible to learn parametric stochastic attachment models which describe the behaviour of incoming nodes, conditioned on an underlying signal processing task on the nodes. We learn this based on how nodes attached to the existing graph previously with replacement. Moreover, the spectral perturbation caused by the attachment of these nodes is shown to be implicitly controlled by the proposed attachment rule. In Chapter 4, we showed that a graph filter bank pair allows us to model edges away from and to the incoming nodes. We propose a general filter learning model and designed graph filters for graph signal de-noising and graph semi-supervised learning. Numerical results show that the proposed approach compares well with baselines relying on the exact topology and outperforms the current solution relying on filter transference.

In terms of filling the gap of not knowing the topology via randomness, stochastic attachment models, as they exist in network-science uses heuristic attachment rules to grow the size of the graph. This keeps the probabilities of attachment fixed once the existing graph is known. We propose a parametric stochastic attachment rule, i.e., it has a different probability for each node. It also allows for a different weight for each possible edge. Being learnable parameters, they can be trained to learn relationships between existing attachment patterns and node level mappings. The learnt attachment behaviour

may not be the true attachment, as is learnt by traditional topology inference, but it has relevance as far as the underlying task is concerned.

The stochastic attachment has implications for graph filters. If the stochastic attachment rule results in an undirected graph, solving for the attachment or filter parameters in expectation is non-trivial as it relies up to the $K$the order statistics of the stochasticity, which is difficult to obtain without imposing strong assumptions. This can be avoided by assuming edges directed either only at the incoming node, or only at the existing nodes. This allows us to not only obtain exactly the expected output, but indirectly solve optimization problems involving the same. Compared to heuristic or data-driven attachments, this new approach to attachment, being data, graph and task aware, typically performs better than purely graph based (stochastic attachment based off network-science) or inductive data-driven transfer-based approaches where a graph is trained on the existing graph and used for the incoming node, only if the connectivity is known.

RQ2: How do we design dynamic algorithms for signal processing on continually expanding graphs?

In Chapter 5, we propose a methodology to perform online graph filtering over graphs which grow sequentially over time. We use a simple online projected gradient descent to update the filter as the size of the graph grows. To account for incoming nodes without known connections, we propose a stochastic variant based on stochastic attachment models which we also adapt to further influence inference tasks.

We show that even without knowing the attachment when making a prediction, the stochastic learner can eventually learn from the data and topological evolution, while the adaptive stochastic learner can learn potentially faster.

There is, however, typically a gap between online filters and deterministic batch solutions. When the incoming node connectivity is not known, we see that this gap can be expressed in terms of the gap between the deterministic and stochastic online filters and the gap between the deterministic online and batch solutions. While the latter tends to zero as the size of the graph grows, the former typically does not. This leads to a static constant regret as the graph grows to infinity, indicating there is a price to pay by not knowing the attachment in an online setting. However, this addresses the worst-case performance upper bound.

The stochastic attachment rule certainly influences the worst-case difference in performance. If the energy of the probability vectors decreases such that their sum is sublinear over time, it results in a theoretical lower bound in performance gap. However, when one adapts the attachment rule to the growing topology and data, it is possible to learn faster from the data stream, based on our experiments. This also leads to a lower steady state regret.

We also showcased the superiority of graph filters as a tool for online graph signal processing, compared to state-of-the-art approaches. Online graph filters can perform better than batch filters due to their ability to adapt to the incoming node stream. However, when the data distribution does not change, batch solutions will fare better. Online approaches tend to fare much better than deploying a pre-trained filter over the node sequence. Compared to kernel-driven solutions, filters perform better while being more interpretable. The online filtering performance can outperform even these determin-

istic approaches in the stochastic scenario, if the attachment rule is selected or inferred appropriately.

RQ3: Given a partially observed evolving topology and a partially observed evolving process over it, are there a collection of latent graphs that drive both the evolution of this topology and of the process?

We tried to answer this question by expressing the topology evolution as a composition of adjacency matrices of the underlying latent graphs along with their time signatures, which control the importance of the latent graphs. We consider the dynamic topology is only partially observed via masking. We also use dynamic graph signals to aid the recovery of these modes and signatures by assuming the signals are smooth over the reconstructed dynamic topology. Through alternating optimization, we show that the recovered latent adjacency matrices and temporal signatures correspond to stationary points.

The latent graphs along with their time signatures are individually and collectively expressive when it comes to documenting the dynamic topology as well as reconstructing missed connections. We see that the latent graphs can play a dominant role in different time intervals over time, showing that certain sets of connections are more dominant depending on their signatures. We also see that in the presence of highly masked observations, graph signals at these nodes can aid the recovery of missing observations. Compared to the traditional low rank-based decompositions and representations of dynamic networks, we show that our approach can predict missing links (both presence and absence) with higher accuracy, thus validating our proposed representation through latent graphs as an alternative. It allows us to look at dynamic networks in a different way.

7.2. FUTURE RESEARCH DIRECTIONS

There exist several open questions arising from the findings of this thesis. In the following subsections, we outline them, with descriptions of the methodology, research questions, and how to potentially approach them.

7.2.1. ONLINE TOPOLOGY IDENTIFICATION ON GROWING GRAPHS

In Chapter 5, we focused on learning the filter $\mathbf{h}$ online as the graph grew. The attachment vector $\mathbf{a}_t$ was either known, or we used a probabilistic attachment rule to infer the signal at the incoming nodes. Instead, we can estimate the expanding topology, given spatio-temporal data over the existing and incoming nodes. Inferring the pair-wise topology from spatio-temporal data has received considerable attention for both batch [11, 81–83, 91, 177] and online settings [85, 93, 97, 178]. However, it has not been investigated for expanding graphs.

To be more specific, consider a sequence of T spatio-temporal signals $\{\mathbf{x}_t\}$. For topology identification, we assume the graph signal $\mathbf{x}_t \in \mathbb{R}^{N_t \times 1}$ at time t corresponds to an unknown graph $\mathcal{G}_t$ of N_t nodes with shift operator $\mathbf{S}_t$. To solve for $\mathbf{S}_t$, we consider a loss function $l(\mathbf{S}, \mathbf{x}_t)$. Examples include stationarity [170, 179] or a Gaussian Markov Random Field (GMRF) assumption [180] on $\mathbf{x}_t$. However, as we observe a sequence of $\mathbf{x}_t$s, we can update the topology $\mathbf{S}_t$ online, as we do in this dissertation. An example would be to

use projected gradient descent type updates, as all $\mathbf{S}_t$s obey topological constraints. The goal is now to estimate not only the connections of the new nodes from the new data but also to update the existing connections in an attempt to learn the true updated topology.

The theoretical analysis of such an online learner is also important. Different from Chapter 5, we need a dynamic regret analysis with the dynamic regret defined as

$$R_S(T) = \frac{1}{T}\sum_{t=1}^{T} l(\mathbf{S}(t), \mathbf{x}_t) - l(\mathbf{S}_t^\star, \mathbf{x}_t) \tag{7.1}$$

where $\mathbf{S}_t^\star$ is the optimal topology of size $\mathbf{N}_t \times N_t$ at time t. There are bounds for dynamic regret that can be extended to this setting [181]. It is important to study when and where these bounds hold. For example, how fast can the optimal topology change so that the online learner has asymptotically zero regret, i.e., do we get closer to estimating the true topology as the underlying graph keeps growing? Another issue is the growing dimensionality of the optimization variable. It is important to study what type of loss functions $l_t(\mathbf{S}_t, \mathbf{x}_t)$, i.e., ones based on stationarity, smoothness, or GMRF, can handle this change of dimension[1].

7.2.2. A BAYESIAN FILTERING APPROACH OVER EXPANDING GRAPHS

Most graph filter design focuses on point estimates [7], like the ones considered in this dissertation. Point estimates do not provide estimates on the uncertainty of predictions whereas Bayesian approaches are useful in this regard [183]. A Bayesian graph filter, for example, would provide uncertainty estimates of predictions at nodes, or graph signals as a whole. It might also promote transferable learning of graph filters from graphs of smaller size to graphs of larger size.

Consider a graph $\mathcal{G}$ along with a set of input-output graph signal pairs $\mathcal{D} = \{\mathbf{x}_n, \mathbf{y}_n\}$. Consider an observation model between $\mathbf{y}_n$ and $\mathbf{x}_n$ such as

$$\mathbf{y}_n = \mathbf{H}(\mathbf{S})\mathbf{x} + \mathbf{n} \tag{7.2}$$

where $\mathbf{H}(\mathbf{S})$ is the graph filter and $\mathbf{n}$ is the additive observation noise. A Bayesian approach necessitates a prior over $\mathbf{h}$. For an appropriate prior, we obtain the posterior filter $\mathbf{h}_p$ and its density $p(\mathbf{h}_p)$. This allows us to interpret the uncertainty surrounding the filter, for example, how sure the filter is about accommodating information from each resolution, along with how the filter weights across orders relate to each other. For example, when the observation likelihood and prior are both Gaussian, the posterior $\mathbf{h}_p$ is also Gaussian. Thus, the frequency response, $\mathbf{h}_p(\boldsymbol{\lambda})$ is also a multi-variate Gaussian. This means the frequency response at each eigenvalue is also Gaussian, providing spectral interpretation.

There are several ways Bayesian Filtering can be applied to expanding graphs. We outline two.

SEQUENTIALLY EXPANDING GRAPHS

Consider a sequence of expanding graphs $\mathcal{G}_0, \ldots, \mathcal{G}_T$ with input signals $\mathbf{x}_0, \ldots, \mathbf{x}_T$ and output signals $\mathbf{y}_0, \ldots, \mathbf{y}_T$ respectively. Let $p_{t-1}(\mathbf{h}_p)$ be the posterior of the filter at time $(t-1)$.

[1] Some related preliminary results can be found in [182]

At time t, when we observe $\mathcal{G}_t$ and the pair $\{\mathbf{x}_t, \mathbf{y}_t\}$, we update the posterior as in done in Kalman filtering. Thus, we keep updating our belief of $\mathbf{h}$ as the graph grows in size. We can also evaluate the performance of the filter depending on the nature of the observed inputs/ outputs and the task at hand. One example could be observing $\mathbf{x}_t$ along with the graph, predicting $\mathbf{y}_t$ and then updating the belief. This can be an alternative to updating the point estimate online, like we have in Chapter 5. The uncertainty can also help in the learning. For example, we can assign importance weights to the online updates based on the uncertainty, which might allow us to incur lesser cumulative error and improve the regret bounds in expectation.

GRAPH TRANSFER LEARNING

Let us have a graph $\mathcal{G}$ with the input-output graph signal pairs as defined in $\mathcal{D}$ in the Bayesian setting and let the posterior be $\mathbf{h}_p \sim \mathcal{N}(\boldsymbol{\mu}_p, \boldsymbol{\Sigma}_p)$ be the Gaussian posterior. We can assume the frequency response $\mathsf{h}(\lambda)$ to be a Gaussian Process [184] with prior $\mathcal{N}(\mu(\lambda), \boldsymbol{\Sigma}(\lambda, \lambda'))$ where

$$\mu(\lambda) = \sum_{k=0}^{K} \lambda^k [\boldsymbol{\mu}_{pos}]_k, \;\; \boldsymbol{\Sigma}(\lambda, \lambda')) = \sum_{i=0}^{K} \sum_{j=0}^{K} \lambda^i [\boldsymbol{\Sigma}_p]_{ij} (\lambda')^j \tag{7.3}$$

which is a polynomial kernel between the eigenvalues. If this assumption were to hold, we could in principle sample frequency responses for a different set of λs, i.e., for different graphs. This allows us to transfer the Bayesian filter we learn to graphs of larger sizes and check their performance relative to point estimate filters. This opens up new questions of analysis in the frequency domain.

For example, if we use the Gaussian Process to sample the frequency response for a larger graph, would the filter be stable relative to perturbations on the larger graph? Mathematically this means the following. For any pair $(\lambda_i), \lambda_j)$ on the larger graph and a given C, does $|\mathsf{h}(\lambda_i) - \mathsf{h}(\lambda_j)| < \frac{C|\lambda_i - \lambda_j|}{\lambda_i + \lambda_j}$ always hold, or does it hold with probability? This would require analysing the integral Lipschitz property of such Bayesian filters [185]. Another research question concerns performance guarantees. For example, can we approximate the true filter response at the larger graph, and what implications does it have for different GSP tasks?

7.2.3. DYNAMIC TOPOLOGY REPRESENTATION

In Chapter 6, we represent the dynamic topology, but we assume the entire dynamic topology to be available all at once. There are several relevant extensions, of which we outline a few.

TIME-VARYING REPRESENTATION

Suppose we do not observe the entire $\underline{\mathbf{A}}$ all at once. This implies a dynamic scenario, where we sequentially observe dynamic topologies (masked or unmasked) of size $N \times N \times T$, i.e., each over a fixed time interval T. This requires updating the modes $\mathbf{A}_r$s and their signatures $\mathbf{C}$ dynamically. Let $\{\mathbf{A}_{r,t}\}$ and $\mathbf{C}_t$ be the representations up to the tth

time-step. Let $\underline{\mathbf{A}}_{t+1}$ and $\underline{\mathbf{M}}_{t+1}$ be the observed dynamic topology tensor and observation mask at time t. We can update the variables by solving

$$\begin{aligned}\{\mathbf{A}_{r,t+1}\}, \mathbf{C}_{r,t+1} = \underset{\{\mathbf{A}_r\},\mathbf{C}}{\operatorname{argmin}} \quad & \frac{1}{2}||\underline{\mathbf{M}}_t \circ (\underline{\mathbf{A}}_t - \sum_{r=1}^{R} \mathbf{A}_r \diamond \mathbf{c}_r)||_F^2 + \gamma \sum_{r=1}^{R} \mathbf{1}_N^\top \mathbf{A}_r \mathbf{1} + \delta \sum_{t=1}^{T} \sum_{r=1}^{R} C_{rt} \frac{\operatorname{tr}(\mathbf{A}_r \mathbf{Z}_t)}{2} \\ & + \mu||\mathbf{DC}||_F^2 + \beta \sum_{i=1}^{R} \sum_{j=1}^{R} \operatorname{tr}(\mathbf{A}_i^\top \mathbf{A}_j) + \frac{\rho}{2}||\mathbf{C}||_F^2 + \eta \sum_{r=1}^{R} ||\mathbf{A}_r - \mathbf{A}_{r,t}||_2^2 + \eta||\mathbf{C} - \mathbf{C}_t||_2^2 \\ & \text{subject to } \mathbf{A} = [\operatorname{vec}(\mathbf{A}_1), \ldots, \operatorname{vec}(\mathbf{A}_R)],\ \mathbf{A}_r \succeq \mathbf{0},\ \operatorname{tr}(\mathbf{A}_r) = 0,\ \mathbf{A}_r = \mathbf{A}_r^\top,\ \mathbf{C} \succeq \mathbf{0}_{T \times R}, \\ & [\mathbf{A}_1, \ldots, \mathbf{A}_r][\mathbf{C}^\top \otimes \mathbf{1}_N] \succeq \zeta \mathbf{1}_{N \times T}\end{aligned} \tag{7.4}$$

where we desire the new representations to be similar to the previous ones by adding the cost $\eta \sum_{r=1}^{R} ||\mathbf{A}_r - \mathbf{A}_{r,t}||_2^2 + \eta||\mathbf{C} - \mathbf{C}_t||_2^2$ with $\eta > 0$ assigning the weight. Alternative costs are also possible, depending on what we want to remain similar. Regret bounds for such approaches can also be derived.

UNIQUENESS OF REPRESENTATIONS

The proposed dynamic graph decomposition can be interpreted as a non-negative matrix factorization (NMF) [186]. The non-negative nature makes it easier to interpret the values in the components. Another advantage of NMF is that it allows for analysing the uniqueness of the decomposition. A unique decomposition implies that the recovered components are unique up to scaling and permutation. For the decomposition proposed in Chapter 6, this means that the modes will be unique up to scaling and permutation, and we do not get fundamentally different modes each time we run the algorithm.

However, the uniqueness of NMF has broadly been studied for exact decomposition, i.e., where the factorization holds with equality, ie., $\underline{\mathbf{A}} = \sum_{r=1}^{R} \mathbf{A}_r \diamond \mathbf{c}_r$ [187–189]. Since we solve a constrained problem, we will not have an exact NMF. So, there is the need to look at uniqueness for non-exact matrix factorization [190]. The criteria proposed in these works can be interpreted in terms of what they mean for the observations recovered modes, signatures, and if they can be used as part of the problem formulation itself. For example, we use non-overlapping edges to make each mode expressive, but does it impact the uniqueness of the decomposition?

A

APPENDIX A

A.1. PROOF OF PROPOSITION 1

The output of an order K filter at node v_1 is [cf. (3.6)] $[\mathbf{y}_1]_{N+1} = \mathbf{a}_1^\top \sum_{k=1}^K h_k \mathbf{A}_0^k \mathbf{x}_0$. The MSE at the incoming node is $\mathbb{E}[([\mathbf{y}_1]_{N+1} - x_1)^2]$. Expanding the MSE for $K \geq 3$ leads to terms of the form $\mathbb{E}[\mathbf{a}_1^\top \mathbf{a}_1 \mathbf{a}_1^\top \mathbf{x}_0]$ which involve the third order statistics of $\mathbf{a}_1$. Computing the latter is notoriously challenging. We approximate the MSE up to second order statistics. Then, by substituting $\mathbf{A}_1$ [cf. (3.1)] into the filtering expression we get

$$[\mathbf{y}_1]_{N+1} \approx \mathbf{a}_1^\top \sum_{k=1}^K h_k \mathbf{A}^{k-1} \mathbf{x}_0 = \mathbf{a}_1^\top \mathbf{A}_x \mathbf{h} \tag{A.1}$$

where $\mathbf{A}_x = [\mathbf{x}_0, \ldots, \mathbf{A}^{K-1}\mathbf{x}_0]$ and $\mathbf{h} = [h_1, \ldots, h_K]^\top$. The MSE is approximately

$$\mathsf{MSE}(\mathbf{p}, \mathbf{w}) \approx \mathbb{E}[(\mathbf{a}_1^\top \mathbf{A}_x \mathbf{h} - x_1)^2]. \tag{A.2}$$

Adding and subtracting $(\mathbf{w} \circ \mathbf{p})^\top \mathbf{A}_x \mathbf{h}$ within the expectation, we get

$$\mathsf{MSE}(\mathbf{p}, \mathbf{w}) \approx \mathbb{E}[(\mathbf{a}_1^\top \mathbf{A}_x \mathbf{h} - (\mathbf{w} \circ \mathbf{p})^\top \mathbf{A}_x \mathbf{h} + (\mathbf{w} \circ \mathbf{p})^\top \mathbf{A}_x \mathbf{h} - x_1)^2] \tag{A.3}$$

which by expanding becomes

$$\begin{aligned} \mathsf{MSE}(\mathbf{p}, \mathbf{w}) \approx\ & \mathbb{E}[(\mathbf{a}_1^\top \mathbf{A}_x \mathbf{h} - (\mathbf{w} \circ \mathbf{p})^\top \mathbf{A}_x \mathbf{h})^2] + \mathbb{E}[(\mathbf{w} \circ \mathbf{p})^\top \mathbf{A}_x \mathbf{h} - x_1)^2] \\ & + 2\mathbb{E}[(\mathbf{a}_1^\top \mathbf{A}_x \mathbf{h} - (\mathbf{w} \circ \mathbf{p})^\top \mathbf{A}_x \mathbf{h})((\mathbf{w} \circ \mathbf{p})^\top \mathbf{A}_x \mathbf{h} - x_1)]. \end{aligned} \tag{A.4}$$

In the first term, we expand the square, take $\mathbf{A}_x \mathbf{h}$ common and take the expectation inside to get $(\mathbf{A}_x \mathbf{h})^\top \mathbf{\Sigma}_1 \mathbf{A}_x \mathbf{h} = (\mathbf{A}_x \mathbf{h})^\top \mathbb{E}[(\mathbf{a}_1 - \mathbf{w} \circ \mathbf{p})(\mathbf{a}_1 - \mathbf{w} \circ \mathbf{p})^\top] \mathbf{A}_x \mathbf{h}$. The second term is deterministic, thus, we can drop the expectation. The third term instead is zero because $\mathbb{E}[\mathbf{a}_1] = \mathbf{w} \circ \mathbf{p}$. Combining these results we get (3.7). □

A.2. PROOF OF COROLLARY 1

When node v_1 only forms directed edges landing on itself, the expanded adjacency matrix $\mathbf{A}_1$ and its lth power become

$$\mathbf{A}_1 = \begin{bmatrix} \mathbf{A} & \mathbf{0} \\ \mathbf{a}_1^\top & 0 \end{bmatrix} \text{ and } \mathbf{A}_1^k = \begin{bmatrix} \mathbf{A}^k & \mathbf{0} \\ \mathbf{a}_1^\top \mathbf{A}^{k-1} & 0 \end{bmatrix}. \tag{A.5}$$

A

Thus, the output of an order K graph filter is

$$[\mathbf{y}_1]_{N+1} = \mathbf{a}_1^\top \sum_{k=1}^{K} \mathbf{A}^{k-1}\mathbf{x}_0 = \mathbf{a}_1^\top \mathbf{A}_x \mathbf{h} \tag{A.6}$$

which is identical to (A.1). Then, the proof follows similarly as for Proposition 1. □

A.3. PROOF OF COROLLARY 2

To find the convexity condition, we analyze when the Hessian of the function in (3.11) is positive semi-definite. The gradient of (3.10) w.r.t. $\mathbf{p}$ is shown in (3.11). The Hessian w.r.t. $\mathbf{p}$ is

$$\nabla_p^2 \mathsf{C}_I(\mathbf{p},\mathbf{w}) = 2(\mathbf{w}\circ\mathbf{A}_x\mathbf{h})(\mathbf{w}\circ\mathbf{A}_x\mathbf{h})^\top - 2\mathrm{diag}((\mathbf{w}\circ\mathbf{A}_x\mathbf{h})^{\circ 2}) + 2\mu_p\mathbf{I}_N. \tag{A.7}$$

The first term $(\mathbf{w}\circ\mathbf{A}_x\mathbf{h})(\mathbf{w}\circ\mathbf{A}_x\mathbf{h})^\top$ is a rank-one matrix with one non-zero eigenvalue $2||\mathbf{w}\circ\mathbf{A}_x\mathbf{h}||^2$ and $N-1$ zero eigenvalues. The second matrix is a diagonal matrix with eigenvalues $\{-2(w_1[\mathbf{A}_x\mathbf{h}]_1)^2,\ldots,-2(w_N[\mathbf{A}_x\mathbf{h}]_N)^2\}$. The third matrix is also diagonal but with each eigenvalue $2\mu_p$. The Hessian is the sum of a rank one matrix and two diagonal matrices. Its eigenvalues are the sum of the eigenvalues of these matrices [191]. By the semi-definite convexity condition [123], each of these eigenvalues now must be greater than or equal to zero. The condition

$$\mu_p \geq \max_i (w_i[\mathbf{A}_x\mathbf{h}]_i)^2 - ||\mathbf{w}\circ\mathbf{A}_x\mathbf{h}||_2^2 \tag{A.8}$$

is sufficient in this case. Since all $w_i \leq w_h$ from the constraint set [cf. (3.10)], we get (3.13) by substituting them with the upper-bound. □

A.4. PROOF OF PROPOSITION 2

The ESSE cost is

$$\mathsf{ESSE}(\mathbf{p},\mathbf{w}) \;=\; \mathbb{E}\big[(\mathsf{QV}(\mathbf{a}) - \mathsf{QV}(\mathbf{a}_1))^2\big]. \tag{A.9}$$

Substituting for $\mathsf{QV}(\cdot)$ in (A.9), we get

$$\mathsf{ESSE}(\mathbf{p},\mathbf{w}) \;=\; \mathbb{E}[(\mathbf{a}^\top\hat{\mathbf{x}} + \mathsf{QV}(\mathbf{x}) - \mathbf{a}_1{}^\top\hat{\mathbf{x}} - \mathsf{QV}(\mathbf{x}))^2] \;=\; \mathbb{E}[(\hat{\mathbf{x}}^\top(\mathbf{a}-\mathbf{a}_1)(\mathbf{a}-\mathbf{a}_1)^\top\hat{\mathbf{x}})\big] \tag{A.10}$$

where $\hat{\mathbf{x}} = \mathbf{x}_0^{\circ 2} - 2x_1\mathbf{x}_0 - x_1^2\mathbf{1}_N$. Taking the expectation operator inside, (A.10) becomes

$$\mathsf{ESSE}(\mathbf{p},\mathbf{w}) = \hat{\mathbf{x}}^\top\big(\mathbb{E}[\mathbf{a}\mathbf{a}^\top] - (\mathbf{w}\circ\mathbf{p})\mathbf{a}_1^\top - \mathbf{a}_1(\mathbf{w}\circ\mathbf{p})^\top + \mathbf{a}_1\mathbf{a}_1^\top\big)\hat{\mathbf{x}} \tag{A.11}$$

where we utilized $\mathbb{E}[\mathbf{a}] = \mathbf{w}\circ\mathbf{p}$. The term $\mathbb{E}[\mathbf{a}\mathbf{a}^\top]$ is related to the covariance matrix of $\mathbf{a}_1$ as $\mathbf{\Sigma}_1 = \mathbb{E}[\mathbf{a}\mathbf{a}^\top] - (\mathbf{w}\circ\mathbf{p})(\mathbf{w}\circ\mathbf{p})^\top = \mathrm{diag}(\mathbf{w}^{\circ 2}\circ\mathbf{p}\circ(\mathbf{1}-\mathbf{p}))$ [cf.(2.30)]. Thus, by direct substitution we get (3.14). □

A.5. GRADIENTS

MSE gradients. Using the chain rule for $\mathbf{p}$ and $\mathbf{w}$, the partial derivatives of the first term of the MSE (3.7) for node v_n with true attachment $\mathbf{a}_n$ and signal x_1 is

$$\nabla_w((\mathbf{w}\circ\mathbf{p})^\top\mathbf{A}_x\mathbf{h}-x_1)^2 = 2((\mathbf{w}\circ\mathbf{p})^\top\mathbf{A}_x\mathbf{h}-x_1)(\mathbf{p}\circ\mathbf{A}_x\mathbf{h}) \tag{A.12}$$

$$\nabla_p((\mathbf{w}\circ\mathbf{p})^\top\mathbf{A}_x\mathbf{h}-x_1)^2 = 2((\mathbf{w}\circ\mathbf{p})^\top\mathbf{A}_x\mathbf{h}-x_1)(\mathbf{w}\circ\mathbf{A}_x\mathbf{h}). \tag{A.13}$$

The second term of the MSE (3.7) is $\sum_{i=1}^N [\mathbf{A}_x\mathbf{h}]_i^2 w_i^2 p_i(1-p_i)$. Taking the derivatives, we get

$$\begin{aligned}\nabla_w(\mathbf{A}_x\mathbf{h})^\top\boldsymbol{\Sigma}_1\mathbf{A}_x\mathbf{h} &= 2(\mathbf{A}_x\mathbf{h})^{\circ 2}\circ(\mathbf{w})\circ\mathbf{p}\circ(\mathbf{1}-\mathbf{p})\\ \nabla_p(\mathbf{A}_x\mathbf{h})^\top\boldsymbol{\Sigma}_1\mathbf{A}_x\mathbf{h} &= (\mathbf{A}_x\mathbf{h})^{\circ 2}\circ(\mathbf{w})^{\circ 2}\circ(\mathbf{1}-2\mathbf{p}).\end{aligned} \tag{A.14}$$

Thus, adding the previous two terms, the gradients for the MSE for v_n becomes

$$\nabla_w\mathsf{MSE}(\mathbf{p},\mathbf{w}) = 2((\mathbf{w}\circ\mathbf{p})^\top\mathbf{A}_x\mathbf{h}-x_1)(\mathbf{p}\circ\mathbf{A}_x\mathbf{h}) + 2(\mathbf{A}_x\mathbf{h})^{\circ 2}\circ(\mathbf{w})\circ\mathbf{p}\circ(\mathbf{1}-\mathbf{p}) \tag{A.15}$$

$$\nabla_p\mathsf{MSE}(\mathbf{p},\mathbf{w}) = 2((\mathbf{w}\circ\mathbf{p})^\top\mathbf{A}_x\mathbf{h}-x_1)(\mathbf{w}\circ\mathbf{A}_x\mathbf{h}) + (\mathbf{A}_x\mathbf{h})^{\circ 2}\circ(\mathbf{w})^{\circ 2}\circ(\mathbf{1}-2\mathbf{p}). \tag{A.16}$$

Subsequently, the gradient for the regularizer $\mu_p||\mathbf{p}-\mathbf{b}_n||^2$ is $2\mu_p(\mathbf{p}-\mathbf{b}_n)$ and for $\mu_w||\mathbf{w}-\mathbf{a}_n||^2$ is $2\mu_w(\mathbf{w}-\mathbf{a}_n)$. Putting back the latter with (A.14)-(A.16), we can get (3.11) and (3.12) with simple arithmetic. □

ESSE gradients. Here we derive the gradient for the ESSE in terms of $\mathbf{p}$ and $\mathbf{w}$. Like with the MSE, consider the case for v_n with signal x_1 and true connection $\mathbf{a}_n$. We consider the vector $\hat{\mathbf{x}}_n$. The first term of the ESSE in (3.14) writes as $\sum_{i=1}^N \hat{x}_i^2 w_i^2 p_i(1-p_i)$. Its gradient w.r.t. w_i is $2\hat{x}_i w_i p_i(1-p_i)$ and w.r.t. p_i it is $\hat{x}_i w_i^2(1-2p_i)$. This allows us to write the vector derivatives

$$\nabla_w\hat{\mathbf{x}}_n^\top\mathrm{diag}((\mathbf{w}\circ\mathbf{p})\hat{\mathbf{x}}_n = 2\mathbf{w}\circ\mathbf{p}\circ(\mathbf{1}-\mathbf{p})\circ\hat{\mathbf{x}}_n^{\circ 2} \tag{A.17}$$

$$\nabla_p\hat{\mathbf{x}}_n^\top\mathrm{diag}((\mathbf{w}\circ\mathbf{p})\hat{\mathbf{x}}_n = \mathbf{w}^{\circ 2}\circ(\mathbf{1}-2\mathbf{p})\circ\hat{\mathbf{x}}_n^{\circ 2}. \tag{A.18}$$

The second term $\hat{\mathbf{x}}_n^\top(\mathbf{w}\circ\mathbf{p})(\mathbf{w}\circ\mathbf{p})^\top\hat{\mathbf{x}}_n$ can be written as $((\mathbf{w}\circ\mathbf{p})^\top\hat{\mathbf{x}}_n)^2$ and its derivatives via the chain rule are

$$\nabla_w\hat{\mathbf{x}}_n^\top(\mathbf{w}\circ\mathbf{p})(\mathbf{w}\circ\mathbf{p})^\top\hat{\mathbf{x}}_n = 2(\mathbf{w}\circ\mathbf{p})^\top\hat{\mathbf{x}}_n(\mathbf{p}\circ\hat{\mathbf{x}}_n) \tag{A.19}$$

$$\nabla_p\hat{\mathbf{x}}_n^\top(\mathbf{w}\circ\mathbf{p})(\mathbf{w}\circ\mathbf{p})^\top\hat{\mathbf{x}}_n = 2(\mathbf{w}\circ\mathbf{p})^\top\hat{\mathbf{x}}_n(\mathbf{w}\circ\hat{\mathbf{x}}_n). \tag{A.20}$$

Similarly the third term $-2\hat{\mathbf{x}}_n^\top(\mathbf{w}\circ\mathbf{p})\mathbf{a}_n\top\hat{\mathbf{x}}_n$ can be written as $-2(\mathbf{a}_n^\top\hat{\mathbf{x}}_n)\sum_{i=1}^N\hat{x}_i w_i p_i$ and thus, its derivatives are

$$\nabla_w(-2\hat{\mathbf{x}}_n^\top(\mathbf{w}\circ\mathbf{p})\mathbf{a}_n^\top\hat{\mathbf{x}}_n) = -2(\mathbf{a}_n^\top\hat{\mathbf{x}}_n)(\mathbf{p}\circ\hat{\mathbf{x}}_n) \tag{A.21}$$

$$\nabla_p(-2\hat{\mathbf{x}}_n^\top(\mathbf{w}\circ\mathbf{p})\mathbf{a}_n\top\hat{\mathbf{x}}_n) = -2(\mathbf{a}_n^\top\hat{\mathbf{x}}_n)(\mathbf{w}\circ\hat{\mathbf{x}}_n). \tag{A.22}$$

The final term $\hat{\mathbf{x}}_n^\top\mathbf{a}_n\mathbf{a}_n^\top\hat{\mathbf{x}}_n^\top$ vanishes. Combining the above and for all the samples in $\mathcal{T}$, the ESSE gradients write as (3.16), (3.17). □

A

A.6. PROOF OF THEOREM 1

For the proof, we will need the following lemma.

Lemma 2 *Consider a cost function $C(\mathbf{s})$ in some variable $\mathbf{s} \in \mathbb{R}^N$ satisfying Assumption 1. Let variable $\mathbf{s}$ be constrained to the convex set $\mathscr{S} = [s_l, s_h]^N$. Let also $\mathbf{s}^u$ and $\mathbf{s}^{u+1}$ be the uth and the $(u+1)$th iterations of a projected gradient descent approach on $\mathbf{s}$ for cost $C(\cdot)$ and let $\tilde{\mathbf{s}}^{u+1}$ be the output of the gradient update step*

$$\tilde{\mathbf{s}}^{u+1} = \mathbf{s}^u - \eta \nabla_s C(\mathbf{s}^u) \tag{A.23}$$

with $\eta > 0$. Then, for the projected vector update $\mathbf{s}^{u+1} = \prod_{\mathscr{S}}(\tilde{\mathbf{s}}^{u+1})$, the following holds:

$$||\mathbf{s}^{u+1} - \mathbf{s}^u|| \leq 2\eta ||\nabla_s C(\mathbf{s}^u)||. \tag{A.24}$$

Proof. Consider vectors $\mathbf{s}^u$, $\mathbf{s}^{u+1}$, and $\tilde{\mathbf{s}}^{u+1}$ as points in $\mathbb{R}^N$ and $||\mathbf{s}^{u+1} - \mathbf{s}^u||$, $||\mathbf{s}^{u+1} - \tilde{\mathbf{s}}^{u+1}||$, and $||\tilde{\mathbf{s}}^{u+1} - \mathbf{s}^u||$ denote the Euclidean distance, i.e, two-norm between them. The triangle inequality gives

$$||\mathbf{s}^{u+1} - \mathbf{s}^u|| \leq ||\mathbf{s}^{u+1} - \tilde{\mathbf{s}}^{u+1}|| + ||\mathbf{s}^u - \tilde{\mathbf{s}}^{u+1}||. \tag{A.25}$$

Since $\mathbf{s}^{u+1}$ is the Euclidean projection of $\tilde{\mathbf{s}}^{u+1}$, we have $||\mathbf{s}^{u+1} - \tilde{\mathbf{s}}^{u+1}|| \leq ||\mathbf{s}^u - \tilde{\mathbf{s}}^{u+1}||$ and inequality (A.25) becomes

$$||\mathbf{s}^{u+1} - \mathbf{s}^u|| \leq 2||\mathbf{s}^u - \tilde{\mathbf{s}}^{u+1}|| = 2|\eta| ||\nabla_s \mathsf{C}(\mathbf{s}^u)||.$$

Note that the lemma holds for $\mathbf{s}$ being $\mathbf{p}$ or $\mathbf{w}$ given the other is fixed, η being η_p or η_w and $\mathscr{S}$ being $[0,1]^N$ or $[w_l, w_h]^N$, respectively. □

Non-increasing cost. Let $\mathsf{C}(\mathbf{p}^{u+1}, \mathbf{w}^u)$ be the cost function evaluated at $\mathbf{p}^{u+1}$ and $\mathbf{w}^u$ [cf. step 5, Alg. 1]. Taking the Taylor expansion at this point, we get

$$\mathsf{C}(\mathbf{p}^{u+1}, \mathbf{w}^u) = \mathsf{C}(\mathbf{p}^u, \mathbf{w}^u) + \nabla_p^\top \mathsf{C}(\mathbf{p}^u)(\mathbf{p}^{u+1} - \mathbf{p}^u) + \frac{1}{2}(\mathbf{p}^{u+1} - \mathbf{p}^u)^\top \nabla_p^2 \mathsf{C}(\mathbf{p}^u, \mathbf{w}^u)(\mathbf{p}^{u+1} - \mathbf{p}^u). \tag{A.26}$$

Under Assumption 1 the Hessian is upper bounded as $\nabla_p^2 \mathsf{C}(\mathbf{p}, \mathbf{w}) \preceq L_p \mathbf{I}$, thus, we have

$$\mathsf{C}(\mathbf{p}^{u+1}, \mathbf{w}^u) \leq \mathsf{C}(\mathbf{p}^u, \mathbf{w}^u) + \nabla_p^\top \mathsf{C}(\mathbf{p}^u)(\mathbf{p}^{u+1} - \mathbf{p}^u) + \frac{L_p}{2}||\mathbf{p}^{u+1} - \mathbf{p}^u||^2. \tag{A.27}$$

We then substitute $\nabla_p \mathsf{C}(\mathbf{p}^u, \mathbf{w}^u) = -\eta_p^{-1}(\tilde{\mathbf{p}}^{u+1} - \mathbf{p}^u)$ for the gradient step to write (A.27) as

$$\mathsf{C}(\mathbf{p}^{u+1}, \mathbf{w}^u) \leq \mathsf{C}(\mathbf{p}^u, \mathbf{w}^u) - \frac{1}{\eta_p}(\tilde{\mathbf{p}}^{u+1} - \mathbf{p}^u)^\top (\mathbf{p}^{u+1} - \mathbf{p}^u) + \frac{L_p}{2}||\mathbf{p}^{u+1} - \mathbf{p}^u||^2. \tag{A.28}$$

Next, we use the cosine rule identity

$$(\tilde{\mathbf{p}}^{u+1} - \mathbf{p}^u)^\top (\mathbf{p}^{u+1} - \mathbf{p}^u) = \frac{1}{2}\left(||\tilde{\mathbf{p}}^{u+1} - \mathbf{p}^u||^2 + ||\mathbf{p}^{u+1} - \mathbf{p}^u||^2 - ||\mathbf{p}^{u+1} - \tilde{\mathbf{p}}^{u+1}||^2\right) \tag{A.29}$$

in the second term of (A.28) to get

$$\mathsf{C}(\mathbf{p}^{u+1},\mathbf{w}^{u}) \leq \mathsf{C}(\mathbf{p}^{u},\mathbf{w}^{u}) - \frac{1}{2\eta_p}||\tilde{\mathbf{p}}^{u+1}-\mathbf{p}^{u}||^2 - \frac{1}{2\eta_p}||\mathbf{p}^{u+1}-\mathbf{p}^{u}||^2 + \frac{1}{2\eta_p}||\mathbf{p}^{u+1}-\tilde{\mathbf{p}}^{u+1}||^2 + \frac{L_p}{2}||\mathbf{p}^{u+1}-\mathbf{p}^{u}||^2. \tag{A.30}$$

The second term on the r.h.s. of (A.30) is lesser than or equal to zero, so we drop it. Since $\mathbf{p}^{u+1}$ is the Euclidean projection of $\tilde{\mathbf{p}}^{u+1}$ onto the constraint set, we have $||\mathbf{p}^{u+1}-\tilde{\mathbf{p}}^{u+1}||^2 \leq ||\tilde{\mathbf{p}}^{u+1}-\mathbf{p}^{u}||^2$. Then, we write (A.30) as

$$\mathsf{C}(\mathbf{p}^{u+1},\mathbf{w}^{u}) \leq \mathsf{C}(\mathbf{p}^{u},\mathbf{w}^{u}) + (\frac{L_p}{2} - \frac{1}{2\eta_p})||\mathbf{p}^{u+1}-\mathbf{p}^{u}||^2 + \frac{1}{2\eta_p}||\tilde{\mathbf{p}}^{u+1}-\mathbf{p}^{u}||^2. \tag{A.31}$$

Using Lemma 2, we substitute $||\tilde{\mathbf{p}}^{u+1}-\mathbf{p}^{u}||^2 \leq 4\eta_p^2||\nabla_p \mathsf{C}(\mathbf{p}^{u},\mathbf{w}^{u})||^2$ in (A.31) and obtain

$$\mathsf{C}(\mathbf{p}^{u+1},\mathbf{w}^{u}) \leq \mathsf{C}(\mathbf{p}^{u},\mathbf{w}^{u}) + \alpha||\nabla_p \mathsf{C}(\mathbf{p}^{u},\mathbf{w}^{u})||^2 \tag{A.32}$$

where $\alpha = \left((\frac{L_p}{2} - \frac{1}{2\eta_p})4\eta_p^2 + \frac{\eta_p}{2}\right) = 2\eta_p^2 L_p - \frac{3\eta_p}{2}$. For $\alpha \leq 0$, the cost reduces in $\mathbf{p}$, i.e., $\mathsf{C}(\mathbf{p}^{u+1},\mathbf{w}^{u}) \leq \mathsf{C}(\mathbf{p}^{u},\mathbf{w}^{u})$, thus, the step size must satisfy $0 < \eta_p \leq \frac{3}{4L_p}$. Hence, the cost is non-increasing with each update in $\mathbf{p}$ and $\mathbf{w}^u$ fixed.

For the $\mathbf{w}$ update, we follow the same approach but we perform the Taylor expansion around the point $(\mathbf{p}^{u+1},\mathbf{w}^{u})$. Following similar derivations, it can be shown that

$$\mathsf{C}(\mathbf{p}^{u+1},\mathbf{w}^{u+1}) \leq \mathsf{C}(\mathbf{p}^{u+1},\mathbf{w}^{u}) + \beta||\nabla_w \mathsf{C}(\mathbf{p}^{u+1},\mathbf{w}^{u})||^2 \tag{A.33}$$

where $\beta = \left((\frac{L_w}{2} - \frac{1}{2\eta_w})4\eta_w^2 + \frac{\eta_w}{2}\right)$, and $\mathsf{C}(\mathbf{p}^{u+1},\mathbf{w}^{u+1}) \leq \mathsf{C}(\mathbf{p}^{u+1},\mathbf{w}^{u})$ if the step size satisfies $0 < \eta_w \leq \frac{3}{4L_w}$. Then, combining the two inequalities we have

$$\mathsf{C}(\mathbf{p}^{u+1},\mathbf{w}^{u+1}) \leq \mathsf{C}(\mathbf{p}^{u},\mathbf{w}^{u}) \tag{A.34}$$

which shows that the alternating projected gradient descent step has a non-increasing cost. □

Local minima. Let least one local minima $(\mathbf{p}^{\star},\mathbf{w}^{\star})$ exists in $\mathscr{S}$. Substituting (A.32) in (A.33) gives

$$\mathsf{C}(\mathbf{p}^{u+1},\mathbf{w}^{u+1}) \leq \mathsf{C}(\mathbf{p}^{u},\mathbf{w}^{u}) + \alpha||\nabla_p \mathsf{C}(\mathbf{p}^{u},\mathbf{w}^{u})||^2 + \beta||\nabla_w \mathsf{C}(\mathbf{p}^{u+1},\mathbf{w}^{u})||^2. \tag{A.35}$$

We denote $\nabla_p \mathsf{C}(\mathbf{p}^{u},\mathbf{w}^{u})$ and $\nabla_w \mathsf{C}(\mathbf{p}^{u+1},\mathbf{w}^{u})$ as $\nabla_p \mathsf{C}(\mathbf{p}^{u})$ and $\nabla_w \mathsf{C}(\mathbf{w}^{u})$ to further ease the notation. We denote by $\mathsf{C}(\mathbf{p}^{\star},\mathbf{w}^{\star})$ the cost at the local minima. Due to the non-increasing cost, after m iterations, we have a condition where the algorithm will be near the feasible local optima. Using then the first order Taylor expansion at this point

$$\mathsf{C}(\mathbf{p}^{u},\mathbf{w}^{u}) = \mathsf{C}(\mathbf{p}^{\star},\mathbf{w}^{\star}) - \nabla_p^{\top}\mathsf{C}(\mathbf{p}^{u})(\mathbf{p}^{\star}-\mathbf{p}^{u}) - \nabla_w^{\top}\mathsf{C}(\mathbf{w}^{u})(\mathbf{w}^{\star}-\mathbf{w}^{u}) \tag{A.36}$$

A

and substituting it in (A.35), we get

$$\mathsf{C}(\mathbf{p}^{u+1},\mathbf{w}^{u+1})-\mathsf{C}(\mathbf{p}^{\star},\mathbf{w}^{\star})\leq\nabla_p^{\top}\mathsf{C}(\mathbf{p}^{u})(\mathbf{p}^{u}-\mathbf{p}^{\star})+\alpha||\nabla_p\mathsf{C}(\mathbf{p}^{u})||^2+\nabla_w^{\top}\mathsf{C}(\mathbf{w}^{u})(\mathbf{w}^{u}-\mathbf{w}^{\star})+\beta||\nabla_w C(\mathbf{w}^{u})||^2. \tag{A.37}$$

We then substitute the cosine rule

$$\nabla_p^{\top}\mathsf{C}(\mathbf{p}^{u})(\mathbf{p}^{u}-\mathbf{p}^{\star})=\frac{\eta_p}{2}||\nabla_p\mathsf{C}(\mathbf{p}^{u})||^2+\frac{1}{2\eta_p}||\mathbf{p}^{u}-\mathbf{p}^{\star}||^2-\frac{1}{2\eta_p}||\mathbf{p}^{u}-\mathbf{p}^{\star}-\eta_p\nabla_p\mathsf{C}(\mathbf{p}^{u})||^2 \tag{A.38}$$

and its equivalent form in $\mathbf{w}$ in (A.37) to get

$$\begin{aligned}\mathsf{C}(\mathbf{p}^{u+1},\mathbf{w}^{u+1})-\mathsf{C}(\mathbf{p}^{\star},\mathbf{w}^{\star})\leq\ &(\frac{\eta_p}{2}+\alpha)||\nabla_p\mathsf{C}(\mathbf{p}^{u})||^2+(\frac{\eta_w}{2}+\beta)||\nabla_w\mathsf{C}(\mathbf{w}^{u})||^2\\&+\frac{1}{2\eta_p}(||\mathbf{p}^{u}-\mathbf{p}^{\star}||^2-||\tilde{\mathbf{p}}^{u+1}-\mathbf{p}^{\star}||^2)\\&+\frac{1}{2\eta_w}(||\mathbf{w}^{u}-\mathbf{w}^{\star}||^2-||\tilde{\mathbf{w}}^{u+1}-\mathbf{w}^{\star}||^2).\end{aligned} \tag{A.39}$$

Now, if $(\frac{\eta_p}{2}+\alpha)\leq 0$ and $(\frac{\eta_w}{2}+\beta)\leq 0$, we can ignore the first two terms in the r.h.s. of (A.39). Substituting for α and β in these conditions, we get $\eta_p\leq\frac{1}{2L_p}$ and $\eta_w\leq\frac{1}{2L_w}$. To prove convergence to the local minima $(\mathbf{p}^{\star},\mathbf{w}^{\star})$ we utilize the inequality $||\mathbf{p}^{u+1}-\mathbf{p}^{\star}||^2\leq||\tilde{\mathbf{p}}^{u+1}-\mathbf{p}^{\star}||^2$, i.e., the gradient update is closer to the optima than the projection update, which holds under the assumption of the local minima being feasible. By using this inequality for both variables, we get

$$\begin{aligned}\mathsf{C}(\mathbf{p}^{u+1},\mathbf{w}^{u+1})-\mathsf{C}(\mathbf{p}^{\star},\mathbf{w}^{\star})\leq\ &\frac{1}{2\eta_p}(||\mathbf{p}^{u}-\mathbf{p}^{\star}||^2-||\mathbf{p}^{u+1}-\mathbf{p}^{\star}||^2)\\&+\frac{1}{2\eta_w}(||\mathbf{w}^{u}-\mathbf{w}^{\star}||^2-||\mathbf{w}^{u+1}-\mathbf{w}^{\star}||^2).\end{aligned} \tag{A.40}$$

Summing from $m=0$ to M, we get a telescoping sum and can write

$$\begin{aligned}\sum_{m=0}^{M}\left(\mathsf{C}(\mathbf{p}^{u+1},\mathbf{w}^{u+1})-\mathsf{C}(\mathbf{p}^{\star},\mathbf{w}^{\star})\right)\leq\ &\frac{1}{2\eta_p}(||\mathbf{p}^{0}-\mathbf{p}^{\star}||^2-||\mathbf{p}^{u+1}-\mathbf{p}^{\star}||^2)\\&+\frac{1}{2\eta_w}(||\mathbf{w}^{0}-\mathbf{w}^{\star}||^2-||\mathbf{w}^{u+1}-\mathbf{w}^{\star}||^2).\end{aligned} \tag{A.41}$$

We divide both sides of (A.41) by $(M+1)$ and use the inequality $\mathsf{C}(\mathbf{p}^{u+1},\mathbf{w}^{u+1})\leq\frac{1}{M+1}\sum_{m=0}^{M}\mathsf{C}(\mathbf{p}^{u+1},\mathbf{w}^{u+1})$ which holds because $\mathsf{C}(\mathbf{p}^{u+1},\mathbf{w}^{u+1})$ is lesser than or equal to all the terms from $m=0,\dots M$ and get

$$\begin{aligned}\mathsf{C}(\mathbf{p}^{u+1},\mathbf{w}^{u+1})-\mathsf{C}(\mathbf{p}^{\star},\mathbf{w}^{\star})\leq\ &\frac{1}{2(M+1)\eta_p}(||\mathbf{p}^{0}-\mathbf{p}^{\star}||^2-||\mathbf{p}^{u+1}-\mathbf{p}^{\star}||^2)\\&+\frac{1}{2(M+1)\eta_w}(||\mathbf{w}^{0}-\mathbf{w}^{\star}||^2-||\mathbf{w}^{u+1}-\mathbf{w}^{\star}||^2).\end{aligned} \tag{A.42}$$

As iteration index $M\to\infty$, $\mathsf{C}(\mathbf{p}^{u+1},\mathbf{w}^{u+1})\to\mathsf{C}(\mathbf{p}^{\star},\mathbf{w}^{\star})$. Thus, convergence to a local minima is possible with rate of convergence $\mathcal{O}(1/U)$. □

A.7. PROOF OF PROPOSITION 3

The perturbation between the realization adjacency matrix $\mathbf{A}_1$ and its nominal $\bar{\mathbf{A}}_1$, $\mathbf{\Delta A}_1 = \mathbf{A}_1 - \bar{\mathbf{A}}_1$ is

$$\mathbf{\Delta A}_1 = \begin{bmatrix} \mathbf{0} & \mathbf{w} \circ (\mathsf{SV}(\mathbf{p}) - \mathbf{1}) \\ \mathbf{w} \circ (\mathsf{SV}(\mathbf{p}) - \mathbf{1})^\top & 0 \end{bmatrix}. \tag{A.43}$$

Invoking Assumption 3, we substitute $\mathbf{\Delta A}_1$ and $\mathbf{v}_{\bar{A},i}$ in (3.23) to get

$$\Delta\gamma_i = \mathbf{v}_{\bar{A},i}^\top \begin{bmatrix} \mathbf{0} & \mathbf{w} \circ (\mathsf{SV}(\mathbf{p}) - \mathbf{1}) \\ \mathbf{w} \circ (\mathsf{SV}(\mathbf{p}) - \mathbf{1})^\top & 0 \end{bmatrix} \mathbf{v}_{\bar{A},i}. \tag{A.44}$$

Then, denoting by $[\mathbf{v}_{\bar{A},i}]_{1:N}$ the vector containing the first N elements of $\mathbf{v}_{\bar{A},i}$, (A.44) can be written as

$$\Delta\gamma_i = (2[\mathbf{v}_{\bar{A},i}]_{N+1}[\mathbf{v}_{\bar{A},i}]_{1:N})^\top \mathbf{w} \circ (\mathsf{SV}(\mathbf{p}) - \mathbf{1}). \tag{A.45}$$

Now, we apply the Cauchy-Schwartz inequality on (A.45) and square both sides to get

$$\Delta^2\gamma_i \leq 4\left|\left|[\mathbf{v}_{\bar{A},i}]_{N+1}[\mathbf{v}_{\bar{A},i}]_{1:N}\right|\right|^2 \left|\left|\mathbf{w} \circ (\mathsf{SV}(\mathbf{p}) - \mathbf{1})\right|\right|^2. \tag{A.46}$$

By taking the expectation, we get

$$\mathbb{E}[\Delta^2\gamma_i] \leq \mathbb{E}\left[4\left|\left|[\mathbf{v}_{\bar{A},i}]_{N+1}[\mathbf{v}_{\bar{A},i}]_{1:N}\right|\right|^2 \left|\left|\mathbf{w} \circ (\mathsf{SV}(\mathbf{p}) - \mathbf{1})\right|\right|^2\right] \tag{A.47}$$

which writes as

$$\mathbb{E}[\Delta^2\gamma_i] \leq c_1 \mathbb{E}[\left|\left|\mathbf{w} \circ (\mathsf{SV}(\mathbf{p}) - \mathbf{1})\right|\right|^2]. \tag{A.48}$$

The expectation operates on the sum $\sum_{i=1}^N w_i^2(\mathsf{SV}(p_i) - 1)^2$. Given w_i is fixed, and utilizing $\mathbb{E}[\mathsf{SV}(p_i)^2] = p_i$, and $\mathbb{E}[\mathsf{SV}(p_i)] = p_i$, the result of the expectation is $\sum_{i=1}^N w_i^2(1 - p_i)$, which writes as $\bar{\mathbf{p}}^\top \mathbf{\Sigma}_1 \bar{\mathbf{p}}$, where $\mathbf{\Sigma}_1$ is the covariance matrix of $\mathbf{a}_1$, and $[\bar{\mathbf{p}}]_i = 1/\sqrt{p_i}$, if $p_i \neq 0$, and zero otherwise. The same steps hold for the perturbed nominal Laplacian $\bar{\mathbf{L}}_1$ and its ith eigenvector $\mathbf{v}_{\bar{L},i}$ to prove (3.25). □

B

APPENDIX B

B.1. PROOF OF LEMMA 1

Substituting $\mathbf{L}_x = \mathbf{L}(\mathbf{I}_{K+1} \otimes \mathbf{x})$ and $\mathbf{M}_x = \mathbf{M}(\mathbf{I}_{K+1} \otimes \mathbf{x})$ into $\mathbb{E}[\mathbf{L}_x^\top \mathbf{C}\mathbf{M}_x]$, we get

$$\mathbb{E}[\mathbf{L}_x^\top \mathbf{C}\mathbf{M}_x] = \mathbb{E}[(\mathbf{I}_{K+1} \otimes \mathbf{x})^\top \mathbf{L}^\top \mathbf{C}\mathbf{M}(\mathbf{I}_{K+1} \otimes \mathbf{x})]. \tag{B.1}$$

Substituting further $\mathbf{L} = [\mathbf{I}, \mathbf{A}, \ldots, \mathbf{A}^K]$, $\mathbf{M} = [\mathbf{I}, \mathbf{A}, \ldots, \mathbf{A}^K]$, the (i, j)th block of $\mathbf{L}^\top \mathbf{C}\mathbf{M}$, is

$$[\mathbf{L}^\top \mathbf{C}\mathbf{M}]_{i,j} = \mathbf{A}^{i-1} \mathbf{C} \mathbf{A}^{j-1} \text{ for } \{i, j\} = 1, \ldots, \{K+1, K+1\}. \tag{B.2}$$

Incorporating the Kronecker products involving $\mathbf{x}$, we further write the (i, j)th entry of (B.1) as

$$\mathbb{E}[\mathbf{L}_x^\top \mathbf{C}\mathbf{M}_x]_{i,j} = \mathbb{E}[\mathbf{x}^\top \mathbf{A}^{i-1} \mathbf{C} \mathbf{A}^{j-1} \mathbf{x}] \tag{B.3}$$

since the expectation acts element-wise. Since the expectation argument is a scalar, we bring in the trace operator and leverage its cyclic property $\text{tr}(\mathbf{xyZ}) = \text{tr}(\mathbf{Zxy})$ to write

$$\mathbb{E}[\mathbf{x}^\top \mathbf{A}^{i-1} \mathbf{C} \mathbf{A}^{j-1} \mathbf{x}] = \mathbb{E}[\text{tr}(\mathbf{x}\mathbf{x}^\top \mathbf{A}^{i-1} \mathbf{C} \mathbf{A}^{j-1})] \tag{B.4}$$

where the only random variable in (B.4) is $\mathbf{x}$. Substituting then $\mathbb{E}[\mathbf{x}\mathbf{x}^\top] = \mathbf{t}\mathbf{t}^\top + \sigma^2 \mathbf{I}_N$ in (B.4), we get

$$\mathbb{E}[\mathbf{L}_x^\top \mathbf{C}\mathbf{M}_x]_{i,j} = \text{tr}(\mathbf{t}\mathbf{t}^\top \mathbf{A}^{i-1} \mathbf{C} \mathbf{A}^{j-1}) + \sigma^2 \text{tr}(\mathbf{A}^{i-1} \mathbf{C} \mathbf{A}^{j-1}). \tag{B.5}$$

The first term in the r.h.s. of (B.5) is $[\mathbf{L}_t^\top \mathbf{C}\mathbf{M}_t]_{i,j} = \mathbf{t}^\top \mathbf{A}^{i-1} \mathbf{C} \mathbf{A}^{j-1} \mathbf{t}$ with $\mathbf{L}_t = \mathbf{L}_x|_{x=t}$ and $\mathbf{M}_t = \mathbf{M}_x|_{x=t}$. Instead for the second term $\sigma^2 \text{tr}(\mathbf{A}^{i-1} \mathbf{C} \mathbf{A}^{j-1})$, we leverage (B.2) and Def. 4.3.1 and note that it is the (i, j)th element of $\text{blktr}(\mathbf{L}^\top \mathbf{C}\mathbf{M}, \sigma^2 \mathbf{I}_N)$. Thus, the (i, j)th element of $\mathbb{E}[\mathbf{L}_x^\top \mathbf{C}\mathbf{M}_x]$ is the sum of the (i, j)th element of these two matrices, proving the Lemma.□

B.2. PROOF OF PROPOSITION 4

Expanding the MSE definition we get

$$\mathbb{E}[\left|\left|\mathbf{W}_1 \mathbf{h} - \mathbf{t}_1\right|\right|_{\mathbf{D}_1}^2] = \mathbf{h}^\top \mathbb{E}[\mathbf{W}_1^\top \mathbf{D}_1 \mathbf{W}_1] \mathbf{h} + 2\mathbf{h}^\top \mathbb{E}[\mathbf{W}_1^\top \mathbf{D}_1 \mathbf{t}_1] + \mathbf{t}_1^\top \mathbf{D}_1 \mathbf{t}_1. \tag{B.6}$$

The first term contains the matrix $\mathbf{\Delta} := \mathbb{E}[\mathbf{W}_1^\top \mathbf{D}\mathbf{W}_1]$. Substituting $\mathbf{W}_1$ [cf. (4.9)], we the expectation argument becomes

$$\begin{aligned}\mathbf{W}_1^\top \mathbf{D}_1 \mathbf{W}_1 &= \begin{bmatrix} \widehat{\mathbf{L}}_x^\top & \mathbf{x}_K \\ \mathbf{M}_x^\top & \widehat{\mathbf{m}}_x \end{bmatrix} \begin{bmatrix} \mathbf{D} & \mathbf{0} \\ \mathbf{0} & d_{N+1} \end{bmatrix} \begin{bmatrix} \widehat{\mathbf{L}}_x & \mathbf{M}_x \\ \mathbf{x}_K^\top & \widehat{\mathbf{m}}_x^\top \end{bmatrix} \\ &= \begin{bmatrix} \widehat{\mathbf{L}}_x^\top \mathbf{D}\widehat{\mathbf{L}}_x + d_{N+1}\mathbf{x}_K\mathbf{x}_K^\top & \widehat{\mathbf{L}}_x^\top \mathbf{D}\mathbf{M}_x + d_{N+1}\mathbf{x}_K\widehat{\mathbf{m}}_x^\top \\ \mathbf{M}_x^\top \mathbf{D}\widehat{\mathbf{L}}_x + d_{N+1}\widehat{\mathbf{m}}_x\mathbf{x}_K^\top & \mathbf{M}_x^\top \mathbf{D}\mathbf{M}_x + d_{N+1}\widehat{\mathbf{m}}_x\widehat{\mathbf{m}}_x^\top \end{bmatrix}\end{aligned} \tag{B.7}$$

which is related to the four blocks appearing in $\mathbf{\Delta}$ and where $\mathbf{D} = \text{diag}(d_1, \ldots, d_N)$.

$\mathbf{\Delta}_{11}$. The first block matrix in (B.7) is $\mathbf{\Delta}_{11} := \mathbb{E}[\widehat{\mathbf{L}}_x^\top \mathbf{D}\widehat{\mathbf{L}}_x + d_{N+1}\mathbf{x}_K\mathbf{x}_K^\top]$.

Substituting $\widehat{\mathbf{L}}_x = \mathbf{L}_x + x_1\overline{\mathbf{L}}_b$, we get

$$\mathbb{E}[\widehat{\mathbf{L}}_x^\top \mathbf{D}\widehat{\mathbf{L}}_x] = \mathbb{E}[(\mathbf{L}_x + x_1\overline{\mathbf{L}}_b)^\top \mathbf{D}(\mathbf{L}_x + x_1\overline{\mathbf{L}}_b)] \tag{B.8}$$

which is further composed of the following four terms:

- $\mathbb{E}[\mathbf{L}_x^\top \mathbf{D}\mathbf{L}_x] = \mathbf{L}_t^\top \mathbf{D}\mathbf{L}_t + \sigma^2 \text{blktr}(\mathbf{L}^\top \mathbf{D}\mathbf{L}, \mathbf{I}_N)$, which follows directly from Lemma 1.
- $\mathbb{E}[x_1\mathbf{L}_x^\top \mathbf{D}\overline{\mathbf{L}}_b] = t_1\mathbf{L}_t^\top \mathbf{D}\overline{\mathbf{L}}_{\mu^{\text{i}}}$ given the noise and the attachments are independent of each other.
- $\mathbb{E}[x_1\overline{\mathbf{L}}_b^\top \mathbf{D}\mathbf{L}_x] = t_1\overline{\mathbf{L}}_{\mu^{\text{i}}}^\top \mathbf{D}\mathbf{L}_t$ under the same independence considerations.
- $\mathbb{E}[x_1^2\overline{\mathbf{L}}_b^\top \mathbf{D}\overline{\mathbf{L}}_b]$. Under the independence between x_1 and $\mathbf{b}_1^{\text{i}}$, and by using Lemma 1 on $\overline{\mathbf{L}}_b^\top \mathbf{D}\overline{\mathbf{L}}_b$, we get

$$\mathbb{E}[x_1^2\overline{\mathbf{L}}_b^\top \mathbf{D}\overline{\mathbf{L}}_b] = (t_1^2 + \sigma^2)\big(\overline{\mathbf{L}}_{\mu^{\text{i}}}^\top \mathbf{D}\overline{\mathbf{L}}_{\mu^{\text{i}}} + \text{blktr}(\overline{\mathbf{L}}^\top \mathbf{D}\overline{\mathbf{L}}, \mathbf{\Sigma}^{\text{i}})\big) \tag{B.9}$$

  where we also used the identity $\mathbb{E}[\mathbf{b}_1^{\text{i}}\mathbf{b}_1^{\text{i}\top}] = \boldsymbol{\mu}^{\text{i}}\boldsymbol{\mu}^{\text{i}\top} + \mathbf{\Sigma}^{\text{i}}$.

In the expression of $\mathbf{\Delta}_{11}$ we also have the term $\mathbb{E}[d_{N+1}\mathbf{x}_K\mathbf{x}_K^\top] = d_{N+1}\text{diag}(t_1^2 + \sigma^2, \mathbf{0}_K)$, which holds because $\mathbf{x}_K = [x_1, \mathbf{0}_K]$. Combining these, we get expression (4.13) for $\mathbf{\Delta}_{11}$.

$\mathbf{\Delta}_{12}$. The second block matrix in (B.7) is $\mathbf{\Delta}_{12} := \mathbb{E}[\widehat{\mathbf{L}}_x^\top \mathbf{D}\mathbf{M}_x + d_{N+1}\mathbf{x}_K\widehat{\mathbf{m}}_x^\top]$. Substituting $\widehat{\mathbf{L}}_x$ and $\widehat{\mathbf{m}}_x^\top = \mathbf{a}_1^{\text{o}\top}\overline{\mathbf{M}}_x + \mathbf{x}_K^\top$, we get

$$\mathbf{\Delta}_{12} = \mathbb{E}[(\mathbf{L}_x + x_1\overline{\mathbf{L}}_b)^\top \mathbf{D}\mathbf{M}_x + d_{N+1}\mathbf{x}_K(\mathbf{a}_1^{\text{o}\top}\overline{\mathbf{M}}_x + \mathbf{x}_K^\top)] \tag{B.10}$$

which is in turn composed of the following terms:

- $\mathbb{E}[\mathbf{L}_x^\top \mathbf{D}\mathbf{M}_x] := \mathbf{L}_t^\top \mathbf{D}\mathbf{M}_t + \sigma^2 \text{blktr}(\mathbf{L}^\top \mathbf{D}\mathbf{M}, \mathbf{I}_N)$ which yields from Lemma 1.
- $\mathbb{E}[x_1\overline{\mathbf{L}}_b^\top \mathbf{D}\mathbf{M}_x] = t_1\overline{\mathbf{L}}_{\mu^{\text{i}}}^\top \mathbf{D}\mathbf{M}_t$ under the independence consideration.
- $\mathbb{E}[d_{N+1}\mathbf{x}_K\mathbf{a}_1^{\text{o}\top}\overline{\mathbf{M}}_x] = d_{N+1}\mathbf{t}_K\boldsymbol{\mu}^{\text{o}\top}\overline{\mathbf{M}}_t$ again under independence $\mathbf{t}_K = [t_1, \mathbf{0}_K]$.
- $d_{N+1}\mathbb{E}[\mathbf{x}_K\mathbf{x}_K^\top]$. Here, note that $\mathbf{x}_K = [x_1, \mathbf{0}_K]$ and $\mathbf{x}_K = [x_1, \mathbf{0}_K]$. Hence, $\mathbb{E}[\mathbf{x}_K\mathbf{x}_K^\top]$ equals $\mathbb{E}[x_1^2] = t_1^2 + \sigma^2$ in position (1,1) and zero elsewhere. Defining then matrix $\mathbf{t}_{LM} \in \mathbb{R}^{L+1\times M+1}$ with $(t_1^2 + \sigma^2)$ in location $(1,1)$ and zero elsewhere, we can write $d_{N+1}\mathbb{E}[\mathbf{x}_K\mathbf{x}_K^\top] = d_{N+1}\mathbf{t}_{LM}$.

Combining then these derivations, we get expression (4.14) for $\mathbf{\Delta}_{12}$.

$\mathbf{\Delta}_{21}$. The third block matrix in (B.7) is $\mathbf{\Delta}_{21} := \mathbb{E}[\mathbf{M}_x^\top \mathbf{D}\widehat{\mathbf{L}}_x + d_{N+1}\widehat{\mathbf{m}}_x \mathbf{x}_K^\top]$. It is easy to see that $\mathbf{\Delta}_{21} = \mathbf{\Delta}_{12}^\top$; hence, (4.14).

$\mathbf{\Delta}_{22}$. The fourth block matrix in (B.7) is $\mathbf{\Delta}_{22} := \mathbb{E}[\mathbf{M}_x^\top \mathbf{D}\mathbf{M}_x + d_{N+1}\widehat{\mathbf{m}}_x \widehat{\mathbf{m}}_x^\top]$. For the first term on the r.h.s. of the latter we have

$$\mathbb{E}[\mathbf{M}_x^\top \mathbf{D}\mathbf{M}_x] = \mathbf{M}_t^\top \mathbf{D}\mathbf{M}_t + \sigma^2 \text{blktr}(\mathbf{M}^\top \mathbf{D}\mathbf{M}, \mathbf{I}_N) \tag{B.11}$$

which yields from Lemma 1. Regarding the second term on the R.H.S, we substitute $\widehat{\mathbf{m}}_x$ and write it out as

$$\begin{aligned}\mathbb{E}[d_{N+1}\widehat{\mathbf{m}}_x \widehat{\mathbf{m}}_x^\top] &= d_{N+1}\mathbb{E}[(\overline{\mathbf{M}}_x^\top \mathbf{a}_1^{\text{o}} + \mathbf{x}_K)(\overline{\mathbf{M}}_x^\top \mathbf{a}_1^{\text{o}} + \mathbf{x}_K)^\top \\ &= d_{N+1}(\overline{\mathbf{M}}_x^\top \mathbf{a}_1^{\text{o}} \mathbf{a}_1^{\text{o}\top} \overline{\mathbf{M}}_x + \overline{\mathbf{M}}_x^\top \mathbf{a}_1^{\text{o}} \mathbf{x}_K^\top + \mathbf{x}_K \mathbf{a}_1^{\text{o}\top} \overline{\mathbf{M}}_x + \mathbf{x}_K \mathbf{x}_K^\top)].\end{aligned} \tag{B.12}$$

We proceed in the same way and elaborate on each terms within the expectation on the r.h.s. of (B.12); respectively:

- $\mathbb{E}[\overline{\mathbf{M}}_x^\top \mathbf{a}_1^{\text{o}} \mathbf{a}_1^{\text{o}\top} \overline{\mathbf{M}}_x] = \text{blktr}(\overline{\mathbf{M}}^\top \mathbf{R}^{\text{o}} \overline{\mathbf{M}}, (\mathbf{t}\mathbf{t}^\top + \sigma^2 \mathbf{I}_N))$ which holds from Lemma 1 and where $\mathbf{R}^{\text{o}} = \mathbf{\Sigma}^{\text{o}} + \boldsymbol{\mu}^{\text{o}} \boldsymbol{\mu}^{\text{o}\top}$.
- $\mathbb{E}[\overline{\mathbf{M}}_x^\top \mathbf{a}_1^{\text{o}} \mathbf{x}_K^\top] = \overline{\mathbf{M}}_t^\top \boldsymbol{\mu}^{\text{o}} \mathbf{t}_K^\top$.
- $\mathbb{E}[\mathbf{x}_K \mathbf{a}_1^{\text{o}\top} \overline{\mathbf{M}}_x] = \mathbf{t}_K \boldsymbol{\mu}^{\text{o}\top} \overline{\mathbf{M}}_t$.
- $\mathbb{E}[\mathbf{x}_K \mathbf{x}_K^\top] = \text{diag}(t_1^2 + \sigma^2, \mathbf{0}_K))$.

Combining all these terms and (B.11) yields expression (4.15) for $\mathbf{\Delta}_{22}$.

Next, we focus on the second expectation on the r.h.s. of (B.6): $\boldsymbol{\theta} := \mathbb{E}[\mathbf{W}_1^\top \mathbf{D}_1 \mathbf{t}_1]$. Substituting once again $\mathbf{W}_1$ [cf. (4.9)], we can write the expectation argument as

$$\mathbf{W}_1^\top \mathbf{D}_1 \mathbf{t}_1 = \begin{bmatrix} \widehat{\mathbf{L}}_x^\top & \mathbf{x}_K \\ \mathbf{M}_x^\top & \widehat{\mathbf{m}}_x \end{bmatrix} \begin{bmatrix} \mathbf{D} & \mathbf{0} \\ \mathbf{0} & d_{N+1} \end{bmatrix} \begin{bmatrix} \mathbf{t} \\ t_1 \end{bmatrix} = \begin{bmatrix} \widehat{\mathbf{L}}_x^\top \mathbf{D}\mathbf{t} + t_1 d_{N+1} \mathbf{x}_L \\ \mathbf{M}_x^\top \mathbf{D}\mathbf{t} + t_1 d_{N+1} \widehat{\mathbf{m}}_x \end{bmatrix}. \tag{B.13}$$

Upon substituting $\widehat{\mathbf{L}}_x$ and $\widehat{\mathbf{m}}_x$ and applying the expectation, expression (4.16) for $\boldsymbol{\theta}$ follows, completing the proof. □

B.3. PROOF OF PROPOSITION 5

Graph $\mathcal{G}_1^{\text{i}}$. The expected TV is

$$\mathbb{E}[\text{TV}(\mathbf{y}_1^{\text{i}})] = \mathbb{E}[\mathbf{y}_1^{\text{i}\top} (\mathbf{I} - \mathbf{A}_1^{\text{i}})^\top (\mathbf{I} - \mathbf{A}_1^{\text{i}}) \mathbf{y}_1^{\text{i}}] \tag{B.14}$$

which by substituting the adjacency matrix $\mathbf{A}_1^{\text{i}}$ [cf. (4.1)] becomes

$$\mathbb{E}[\text{TV}(\mathbf{y}_1^{\text{i}})] = \mathbb{E}\left[\mathbf{y}_1^{\text{i}\top} \begin{bmatrix} \mathbf{\Gamma} & -(\mathbf{I} - \mathbf{A})^\top \mathbf{b}_1^{\text{i}} \\ -\mathbf{b}_1^{\text{i}\top} (\mathbf{I} - \mathbf{A}) & \mathbf{b}_1^{\text{i}\top} \mathbf{b}_1^{\text{i}} + 1 \end{bmatrix} \mathbf{y}_1^{\text{i}}\right] \tag{B.15}$$

with $\mathbf{\Gamma} = (\mathbf{I}-\mathbf{A})^\top(\mathbf{I}-\mathbf{A})$. Substituting further $\mathbf{y}_1^{\mathrm{i}}$ [cf. (4.6)] with $\mathbf{x}_1 = \mathbf{t}_1$ we can write (B.15) as

$$\mathbb{E}[S_2(\mathbf{y}_1^{\mathrm{i}})] = \mathbf{h}^{\mathrm{i}\top}\mathbb{E}\Big[\hat{\mathbf{L}}_t^\top\mathbf{\Gamma}\hat{\mathbf{L}}_t - \mathbf{t}_K\mathbf{b}_1^{\mathrm{i}\top}(\mathbf{I}-\mathbf{A})\hat{\mathbf{L}}_t - \hat{\mathbf{L}}_t^\top(\mathbf{I}-\mathbf{A})^\top\mathbf{b}_1^{\mathrm{i}}\mathbf{t}_K^\top + (\mathbf{b}_1^{\mathrm{i}\top}\mathbf{b}_1^{\mathrm{i}}+1)\mathbf{t}_K\mathbf{t}_K^\top\Big]\mathbf{h}^{\mathrm{i}}. \quad \text{(B.16)}$$

B

We now proceed by applying the expectation to each term on the r.h.s. of (B.16). For the first term, we substitute $\hat{\mathbf{L}}_t = (\mathbf{L}_t + t_1\overline{\mathbf{L}}_{b^{\mathrm{i}}})$ with $\overline{\mathbf{L}}_{b^{\mathrm{i}}} = \overline{\mathbf{L}}|_{x=b^{\mathrm{i}}}$ and get

$$\mathbb{E}[\hat{\mathbf{L}}_t^\top\mathbf{\Gamma}\hat{\mathbf{L}}_t] = \mathbb{E}[(\mathbf{L}_t + t_1\overline{\mathbf{L}}_{b^{\mathrm{i}}})^\top\mathbf{\Gamma}(\mathbf{L}_t + t_1\overline{\mathbf{L}}_{b^{\mathrm{i}}})]. \quad \text{(B.17)}$$

This is in turn composed of the following four terms:

- $\mathbb{E}[\mathbf{L}_t^\top\mathbf{\Gamma}\mathbf{L}_t^\top] = \mathbf{L}_t^\top\mathbf{\Gamma}\mathbf{L}_t$. which is unaffected by expectation.
- $\mathbb{E}[t_1\mathbf{L}_t^\top\mathbf{\Gamma}\overline{\mathbf{L}}_{b^{\mathrm{i}}}] = t_1\mathbf{L}_t^\top\mathbf{\Gamma}\overline{\mathbf{L}}_{\boldsymbol{\mu}^{\mathrm{i}}}$ where we use $\mathbb{E}[\mathbf{b}^{\mathrm{i}}] = \boldsymbol{\mu}^{\mathrm{i}}$.
- $\mathbb{E}[t_1\mathbf{L}_{\boldsymbol{\mu}^{\mathrm{i}}}\mathbf{\Gamma}\mathbf{L}_t] = t_1\overline{\mathbf{L}}_{\boldsymbol{\mu}^{\mathrm{i}}}^\top\mathbf{\Gamma}\mathbf{L}_t$.
- $\mathbb{E}[t_1^2\overline{\mathbf{L}}_{b^{\mathrm{i}}}^\top\mathbf{\Gamma}\overline{\mathbf{L}}_{b^{\mathrm{i}}}] = t_1^2(\overline{\mathbf{L}}_{\mu^{\mathrm{i}}}^\top\mathbf{\Gamma}\overline{\mathbf{L}}_{\mu^{\mathrm{i}}} + \mathrm{blktr}(\overline{\mathbf{L}}^\top\mathbf{\Gamma}\overline{\mathbf{L}}, \mathbf{\Sigma}^{\mathrm{i}}))$ where we use Lemma 1 and $\mathbb{E}(\mathbf{b}_1^{\mathrm{i}}\mathbf{b}_1^{\mathrm{i}\top}) = \mathbf{\Sigma}^{\mathrm{i}} + \boldsymbol{\mu}^{\mathrm{i}}\boldsymbol{\mu}^{\mathrm{i}\top}$.

The second term in (B.16) $\mathbb{E}[\mathbf{t}_K\mathbf{b}_1^{\mathrm{i}\top}(\mathbf{I}-\mathbf{A})\hat{\mathbf{L}}_t]$ is composed of

- $\mathbb{E}[\mathbf{t}_K\mathbf{b}_1^{\mathrm{i}\top}\hat{\mathbf{L}}_t]$: Substituting $\hat{\mathbf{L}}_t = (\mathbf{L}_t + t_1\overline{\mathbf{L}}_{b^{\mathrm{i}}})$ we have

$$\mathbb{E}[\mathbf{t}_K\mathbf{b}_1^{\mathrm{i}\top}\hat{\mathbf{L}}_t] = \mathbf{t}_K\mathbb{E}[\mathbf{b}_1^{\mathrm{i}\top}\mathbf{L}_t] + t_1\mathbb{E}[\mathbf{b}_1^{\mathrm{i}\top}\overline{\mathbf{L}}_b]. \quad \text{(B.18)}$$

  The term $\mathbf{b}_1^{\mathrm{i}\top}\mathbf{L}_t$ has expectation $\boldsymbol{\mu}^{\mathrm{i}\top}\mathbf{L}_t$. Note however, that we also have $\mathbf{b}_1^{\mathrm{i}\top}\overline{\mathbf{L}}_b = \begin{bmatrix}\mathbf{0} & \mathbf{b}_1^{\mathrm{i}\top}\mathbf{A}\mathbf{b}_1^{\mathrm{i}} & \dots & \mathbf{b}_1^{\mathrm{i}\top}\mathbf{A}^{K-1}\mathbf{b}_1^{\mathrm{i}}\end{bmatrix}$. By using Lemma 1, we have $\mathbb{E}[\mathbf{b}_1^{\mathrm{i}\top}\overline{\mathbf{L}}_b] = \boldsymbol{\mu}^{\mathrm{i}\top}\overline{\mathbf{L}}_{\mu^{\mathrm{i}}} + \mathrm{blktr}(\overline{\mathbf{L}}, \mathbf{\Sigma}^{\mathrm{i}})$. Combining, we get

$$\mathbb{E}[\mathbf{t}_K\mathbf{b}_1^{\mathrm{i}\top}\hat{\mathbf{L}}_t] = \mathbf{t}_K(\boldsymbol{\mu}_+^{\mathrm{i}\top}\mathbf{L}_t + t_1(\boldsymbol{\mu}^{\mathrm{i}\top}\overline{\mathbf{L}}_{\mu^{\mathrm{i}}} + \mathrm{blktr}(\overline{\mathbf{L}}, \mathbf{\Sigma}^{\mathrm{i}}))). \quad \text{(B.19)}$$

- $\mathbb{E}[\mathbf{t}_K\mathbf{b}_1^{\mathrm{i}\top}\mathbf{A}\hat{\mathbf{L}}_t] = \mathbf{t}_K\mathbb{E}[\mathbf{b}_1^{\mathrm{i}\top}\mathbf{A}\mathbf{L}_t] + t_1\mathbb{E}[\mathbf{b}_1^{\mathrm{i}\top}\mathbf{A}\overline{\mathbf{L}}_b]$, which by following similar arguments yields

$$\mathbb{E}[\mathbf{t}_K\mathbf{b}_1^{\mathrm{i}\top}\mathbf{A}\hat{\mathbf{L}}_t] = \mathbf{t}_K(\boldsymbol{\mu}^{\mathrm{i}\top}\mathbf{A}\mathbf{L}_t + t_1(\boldsymbol{\mu}^{\mathrm{i}\top}\mathbf{A}\overline{\mathbf{L}}_{\mu^{\mathrm{i}}} + \mathrm{blktr}(\mathbf{A}\overline{\mathbf{L}}, \mathbf{\Sigma}^{\mathrm{i}}))). \quad \text{(B.20)}$$

Combining (B.19) and (B.20), we get

$$\begin{aligned}\mathbb{E}[\mathbf{t}_K\mathbf{b}_1^{\mathrm{i}\top}(\mathbf{I}-\mathbf{A})\hat{\mathbf{L}}_t] = \mathbf{t}_K(\boldsymbol{\mu}^{\mathrm{i}\top}\mathbf{L}_t + t_1(\boldsymbol{\mu}^{\mathrm{i}\top}\overline{\mathbf{L}}_{\mu^{\mathrm{i}}} + \mathrm{blktr}(\overline{\mathbf{L}}, \mathbf{\Sigma}^{\mathrm{i}}))) - \mathbf{t}_K\Big(\boldsymbol{\mu}^{\mathrm{i}\top}\mathbf{A}\mathbf{L}_t + t_1\boldsymbol{\mu}^{\mathrm{i}\top}\mathbf{A}\overline{\mathbf{L}}_{\mu^{\mathrm{i}}} \\ + \mathrm{blktr}(\mathbf{A}\overline{\mathbf{L}}, \mathbf{\Sigma}^{\mathrm{i}}))\Big).\end{aligned} \quad \text{(B.21)}$$

The third term is the transpose of the second one, i.e.,

$$\mathbb{E}[\hat{\mathbf{L}}_t^\top(\mathbf{I}-\mathbf{A})^\top\mathbf{b}_1^{\mathrm{i}}\mathbf{t}_K^\top] = \mathbb{E}[\mathbf{t}_K\mathbf{b}_1^{\mathrm{i}\top}(\mathbf{I}-\mathbf{A})\hat{\mathbf{L}}_t]^\top. \quad \text{(B.22)}$$

The final term is

$$\mathbb{E}[(\mathbf{b}_1^{\mathsf{i}\top}\mathbf{b}_1^{\mathsf{i}}+1)\mathbf{t}_K\mathbf{t}_K^\top] = \big(\sum_{n=1}^{N}\mathbb{E}[[\mathbf{b}_1^{\mathsf{i}}]_n^2]+1\big)\mathrm{diag}(t_1^2,\mathbf{0}) \tag{B.23}$$

where we used $\mathbf{t}_K\mathbf{t}_K^\top = \mathrm{diag}(t_1^2,\mathbf{0}_K)$. Given that $\mathbb{E}[[\mathbf{b}_1^{\mathsf{i}}]_n^2] = w_n^2 p_n$, we can write this as

$$\mathbb{E}[(\mathbf{b}_1^{\mathsf{i}\top}\mathbf{b}_1^{\mathsf{i}}+1)\mathbf{t}_K\mathbf{t}_K^\top] = (\mathbf{W}^\top\mathbf{p}^{\mathsf{i}}+1)\mathrm{diag}(t_1^2,\mathbf{0}_K). \tag{B.24}$$

Combining these together we get expression (4.21) for $\boldsymbol{\Psi}^{\mathsf{i}}$.

Graph $\mathscr{G}_1^{\mathsf{o}}$. By substituting $\mathbf{A}_1^{\mathsf{o}}$ [cf. (4.1)] and $\mathbf{y}_1^{\mathsf{o}} = [\mathbf{M}_t\mathbf{h}^{\mathsf{o}}, \widehat{\mathbf{m}}_t^\top\mathbf{h}^{\mathsf{o}}]^\top$ in $\mathbf{y}_1^{\mathsf{o}\top}(\mathbf{I}-\mathbf{A}_1^{\mathsf{o}})^\top(\mathbf{I}-\mathbf{A}_1^{\mathsf{o}})\mathbf{y}_1^{\mathsf{o}}$ the expected TV is

$$\mathbb{E}[\mathsf{TV}(\mathbf{y}_1^{\mathsf{o}})] = \mathbb{E}\Big[\begin{bmatrix}\mathbf{h}^{\mathsf{o}\top}\mathbf{M}_t^\top & \mathbf{h}^{\mathsf{o}\top}\widehat{\mathbf{m}}_t\end{bmatrix}\begin{bmatrix}\boldsymbol{\Gamma}+\mathbf{A}_1^{\mathsf{o}}\mathbf{A}_1^{\mathsf{o}\top} & -\mathbf{A}_1^{\mathsf{o}} \\ -\mathbf{A}_1^{\mathsf{o}\top} & 1\end{bmatrix}\begin{bmatrix}\mathbf{M}_t\mathbf{h}^{\mathsf{o}} \\ \widehat{\mathbf{m}}_t^\top\mathbf{h}^{\mathsf{o}}\end{bmatrix}\Big]. \tag{B.25}$$

Utilizing $\widehat{\mathbf{m}}_t = \overline{\mathbf{M}}_t^\top\mathbf{A}^{\mathsf{o}}+\mathbf{t}_K$, we get

$$\begin{aligned}\mathbb{E}[\mathsf{TV}(\mathbf{y}_1^{\mathsf{o}})] = \mathbf{h}^{\mathsf{o}\top}\mathbb{E}\Big[&\mathbf{M}_t^\top(\boldsymbol{\Gamma}+\mathbf{A}_1^{\mathsf{o}}\mathbf{A}_1^{\mathsf{o}\top})\mathbf{M}_t - \mathbf{M}_t^\top\mathbf{A}_1^{\mathsf{o}}(\mathbf{A}_1^{\mathsf{o}\top}\overline{\mathbf{M}}_t+\mathbf{t}_K^\top) - (\overline{\mathbf{M}}_t^\top\mathbf{A}^{\mathsf{o}}+\mathbf{t}_K)\mathbf{A}_1^{\mathsf{o}\top}\mathbf{M}_t \\ &+ (\overline{\mathbf{M}}_t^\top\mathbf{A}^{\mathsf{o}}+\mathbf{t}_K)(\overline{\mathbf{M}}_t^\top\mathbf{A}^{\mathsf{o}}+\mathbf{t}_K)^\top\Big]\mathbf{h}^{\mathsf{o}}.\end{aligned} \tag{B.26}$$

Equation (B.26) comprises the following four terms:

- $\mathbb{E}[\mathbf{M}_t^\top(\boldsymbol{\Gamma}+\mathbf{A}_1^{\mathsf{o}}\mathbf{A}_1^{\mathsf{o}\top})\mathbf{M}_t] = \mathbf{M}_t^\top(\boldsymbol{\Gamma}+\mathbf{R}^{\mathsf{o}})\mathbf{M}_t$ since $\mathbf{A}_1^{\mathsf{o}}$ is random here and $\mathbf{R}^{\mathsf{o}} = \mathbb{E}[\mathbf{A}_1^{\mathsf{i}}\mathbf{A}_1^{\mathsf{i}\top}]$.
- $\mathbb{E}[\mathbf{M}_t^\top\mathbf{A}_1^{\mathsf{o}}(\mathbf{A}_1^{\mathsf{o}\top}\overline{\mathbf{M}}_t+\mathbf{t}_K^\top)] = \mathbb{E}[\mathbf{M}_t^\top\mathbf{A}_1^{\mathsf{o}}\mathbf{A}_1^{\mathsf{o}\top}\overline{\mathbf{M}}_t] + \mathbb{E}[\mathbf{M}_t^\top\mathbf{A}_1^{\mathsf{o}}\mathbf{t}_K^\top] = \mathbf{M}_t^\top\mathbf{R}^{\mathsf{o}}\overline{\mathbf{M}}_t + \mathbf{M}_t^\top\boldsymbol{\mu}^{\mathsf{o}}\mathbf{t}_K^\top$.
- The third term is the transpose of the above, hence it has the expectation $\overline{\mathbf{M}}_t^\top\mathbf{R}^{\mathsf{o}}\mathbf{M}_t + \mathbf{t}_K\boldsymbol{\mu}^{\mathsf{o}\top}\mathbf{M}_t$.
- $\mathbb{E}[(\overline{\mathbf{M}}_t^\top\mathbf{A}^{\mathsf{o}}+\mathbf{t}_K)(\overline{\mathbf{M}}_t^\top\mathbf{A}^{\mathsf{o}}+\mathbf{t}_K)^\top] = \mathbb{E}[\overline{\mathbf{M}}_t^\top\mathbf{A}^{\mathsf{o}}\mathbf{A}^{\mathsf{o}\top}\overline{\mathbf{M}}_t] + \mathbb{E}[\overline{\mathbf{M}}_t^\top\mathbf{A}^{\mathsf{o}}\mathbf{t}_K^\top] + \mathbb{E}[\mathbf{t}_K\mathbf{A}^{\mathsf{o}\top}\overline{\mathbf{M}}_t] + \mathbb{E}[\mathbf{t}_K\mathbf{t}_K^\top] = \overline{\mathbf{M}}_t^\top\mathbf{R}^{\mathsf{o}}\overline{\mathbf{M}}_t + \overline{\mathbf{M}}_t^\top\boldsymbol{\mu}^{\mathsf{o}}\mathbf{t}_K^\top + \mathbf{t}_K\boldsymbol{\mu}^{\mathsf{o}\top}\overline{\mathbf{M}}_t + \mathbf{t}_K\mathbf{t}_K^\top$ by operating term-wise.

Combining these together, we get expression (4.22) for $\boldsymbol{\Psi}^{\mathsf{o}}$, completing the proof. □

C

APPENDIX C

C.1. PROOF OF THEOREM 2

The regret relative to the optimal filter $\mathbf{h}^\star$ is

$$R_{s,T}(\mathbf{h}^\star) = \sum_{t=1}^{T} l_t^s(\mathbf{h}^s(t-1), x_t) - l_t^d(\mathbf{h}^\star, x_t). \tag{C.1}$$

By adding and subtracting the terms $\sum_{t=1}^{T} l_t^d(\mathbf{h}^s(t-1), x_t)$ and $\sum_{t=1}^{T} l_t^d(\mathbf{h}^d(t-1), x_t)$ we obtain

$$\begin{aligned} R_{s,T}(\mathbf{h}^\star) = &\sum_{t=1}^{T} l_t^s(\mathbf{h}^s(t-1), x_t) - l_t^d(\mathbf{h}^s(t-1), x_t) + \sum_{t=1}^{T} l_t^d(\mathbf{h}^s(t-1), x_t) - l_t^d(\mathbf{h}^d(t-1), x_t) \\ &+ \sum_{t=1}^{T} l_t(\mathbf{h}^d(t-1), x_t) - l_t^d(\mathbf{h}^\star, x_t) \end{aligned} \tag{C.2}$$

where $l_t^d(\mathbf{h}^s(t-1), x_t)$ is the deterministic loss at time t evaluated with the filter updated in the stochastic scenario. The regret in (C.2) comprises three sums over the T-length sequence, each of which contributes to the overall regret.

The first term in (C.2), $\sum_{t=1}^{T} l_t^s(\mathbf{h}^s(t-1), x_t) - l_t^d(\mathbf{h}^s(t-1), x_t)$ measures the difference in the stochastic and the deterministic loss for the filter updated online in the stochastic setting. We substitute

$$l_t^s(\mathbf{h}^s(t-1), x_t) = ((\mathbf{w}_t \circ \mathbf{p}_t)^\top \bar{\mathbf{y}}_t - x_t)^2 + \bar{\mathbf{y}}_t^\top \mathbf{\Sigma}_t \bar{\mathbf{y}}_t + \mu ||\mathbf{h}^s(t-1)||_2^2 \tag{C.3}$$

where $\bar{\mathbf{y}}_t = \mathbf{A}_{x,t-1}\mathbf{h}^s(t-1)$ and

$$l_t^d(\mathbf{h}^s(t-1), x_t) = (\mathbf{a}_t^\top \bar{\mathbf{y}}_t - x_t)^2 + \mu ||\mathbf{h}^s(t-1)||_2^2 \tag{C.4}$$

to get the difference at time t as

$$l_t^s(\mathbf{h}^s(t-1), x_t) - l_t^d(\mathbf{h}^s(t-1), x_t) = ((\mathbf{w}_t \circ \mathbf{p}_t)^\top \bar{\mathbf{y}}_t - x_t)^2 - (\mathbf{a}_t^\top \bar{\mathbf{y}}_t - x_t)^2 + \bar{\mathbf{y}}_t^\top \mathbf{\Sigma}_t \bar{\mathbf{y}}_t. \tag{C.5}$$

After some simplification, we get

$$l_t^s(\mathbf{h}^s(t-1),x_t)-l_t^d(\mathbf{h}^s(t-1),x_t)=((\mathbf{w}_t\circ\mathbf{p}_t-\mathbf{a}_t)^\top\bar{\mathbf{y}}_t)^2+2(\mathbf{w}_t\circ\mathbf{p}_t-\mathbf{a}_t)^\top\bar{\mathbf{y}}_t(\mathbf{a}_t^\top\bar{\mathbf{y}}_t-x_t) + \bar{\mathbf{y}}_t^\top\boldsymbol{\Sigma}_t\bar{\mathbf{y}}_t. \tag{C.6}$$

The r.h.s. of equation (C.6) has three terms. For the first term we have

$$((\mathbf{w}_t\circ\mathbf{p}_t-\mathbf{a}_t)^\top\bar{\mathbf{y}}_t)^2\leq||\mathbf{w}_t\circ\mathbf{p}_t-\mathbf{a}_t||_2^2||\bar{\mathbf{y}}_t||_2^2\leq w_h^2(||\mathbf{p}_t||_2^2+M_{max})Y^2 \tag{C.7}$$

where the first inequality follows from the Cauchy-Schwartz inequality and the second inequality from Lemmas 4 and Lemma 5 in Appendix C.4.

For the second term we have

$$\begin{aligned}&2(\mathbf{w}_t\circ\mathbf{p}_t-\mathbf{a}_t)^\top\bar{\mathbf{y}}_t(\mathbf{a}_t^\top\bar{\mathbf{y}}_t-x_t)\leq 2||\mathbf{w}_t\circ\mathbf{p}_t-\mathbf{a}_t||_2||\bar{\mathbf{y}}_t||_2||\mathbf{a}_t^\top\bar{\mathbf{y}}_t-x_t||_2\\&\leq 2Rw_hY\sqrt{||\mathbf{p}_t||_2^2+M_{max}}\end{aligned} \tag{C.8}$$

where the first inequality follows from the Cauchy-Schwartz inequality and the second inequality from Assumption 7, Lemmas 5 and 4. For the third term, we have

$$\bar{\mathbf{y}}_t^\top\boldsymbol{\Sigma}_t\bar{\mathbf{y}}_t=\sum_{n=1}^{N_{t-1}}[\bar{\mathbf{y}}_t]_n^2[\mathbf{w}_t]_n^2[\mathbf{p}_t]_n(1-[\mathbf{p}_t]_n)\leq w_h^2\bar{\sigma}_t^2Y^2 \tag{C.9}$$

where the inequality follows the definition of $\bar{\sigma}_t^2$ and Lemma 5. Adding (C.7)-(C.9) we can upper-bound (C.6) as

$$\sum_{t=1}^T l_t^s(\mathbf{h}^s(t-1),x_t)-l_t^d(\mathbf{h}^s(t-1),x_t)\leq w_h^2Y^2(||\mathbf{p}_t||_2^2+M_{max})+2Rw_hY\sqrt{||\mathbf{p}_t||_2^2+M_{max}} + w_h^2\bar{\sigma}_t^2Y^2. \tag{C.10}$$

The second term in (C.2), $\sum_{t=1}^T l_t^d(\mathbf{h}^s(t-1),x_t)-l_t^d(\mathbf{h}^d(t-1),x_t)$ measures the sum of the differences in the deterministic loss between the deterministic and stochastic online filter. Since $l_t^d(\cdot,\cdot)$ is Lipschitz with constant L_d from Lemma 3, we can write

$$|l_t^d(\mathbf{h}^s(t-1),x_t)-l_t^d(\mathbf{h}^d(t-1),x_t)|\leq L_d||\mathbf{h}^s(t-1)-\mathbf{h}^d(t-1)||_2 \tag{C.11}$$

which implies $l_t(\mathbf{h}^s(t-1),x_t)-l_t(\mathbf{h}^d(t-1),x_t)\leq L_d||\mathbf{h}^s(t-1)-\mathbf{h}^d(t-1)||_2$. Summing over t, we have

$$\sum_{t=1}^T l_t^d(\mathbf{h}^s(t-1),x_t)-l_t^d(\mathbf{h}^d(t-1),x_t)\leq L_d\sum_{t=1}^T||\mathbf{h}^s(t-1)-\mathbf{h}^d(t-1)||_2. \tag{C.12}$$

The third term in (C.2), $\sum_{t=1}^T l_t^d(\mathbf{h}^d(t-1),x_t)-l_t^d(\mathbf{h}^\star,x_t)$ corresponds to the static regret in the deterministic case and has the upper bound

$$\sum_{t=1}^T\left(l_t(\mathbf{h}^d(t-1),x_t)-l_t^d(\mathbf{h}^\star,x_t)\right)\leq\frac{||\mathbf{h}^\star||_2^2}{2\eta}+\frac{\eta}{2}L_d^2T \tag{C.13}$$

as shown in Proposition 1.

By summing equations (C.10), (C.12), and (C.13), we obtain

$$R_{s,T}(\mathbf{h}^\star) \leq \sum_{t=1}^{T} w_h^2 Y^2(||\mathbf{p}_t||_2^2 + M_{max}) + 2Rw_h Y\sqrt{||\mathbf{p}_t||_2^2 + M_{max}} + w_h^2\bar{\sigma}_t^2 Y^2 \\ + L_d||\mathbf{h}^s(t-1) - \mathbf{h}^d(t-1)||_2\Big) + \frac{||\mathbf{h}^\star||_2^2}{2\eta} + \frac{\eta}{2}L_d^2 T. \tag{C.14}$$

Finally, dividing both sides by T and using Lemma 5, we complete the proof. □

C

C.2. PROOF OF COROLLARY 3

We substitute $\mathbf{p}_t = \frac{1}{N_{t-1}}\mathbf{1}_{N_{t-1}}$, in each term of the stochastic regret bound. For the first term we have

$$\sum_{t=1}^{T} w_h^2 Y^2(||\mathbf{p}_t||_2^2 + M_{max}) \leq w_h^2 Y^2 \sum_{t=1}^{T} \frac{1}{N_{t-1}} + w_h^2 M_{max} Y^2 T. \tag{C.15}$$

Substituting $N_t = N_0 + t - 1$, we have

$$\sum_{t=1}^{T} w_h^2 Y^2(||\mathbf{p}_t||_2^2 + M_{max}) \leq w_h^2 Y \sum_{t=1}^{T} \frac{1}{N_0 + t - 1} + w_h^2 M_{max} Y^2 T. \tag{C.16}$$

Next, we bound the summation $\sum_{t=1}^{T} \frac{1}{N_0+t-1}$ as [1]

$$\sum_{t=1}^{T} \frac{1}{N_0 + t - 1} \leq \text{in}_{t=1}^{T} \frac{1}{t + N_0 - 1} dt \tag{C.17}$$

which gives us

$$\sum_{t=1}^{T} \frac{1}{N_0 + t - 1} \leq \log(T + N_0 - 1) - \log(N_0). \tag{C.18}$$

Substituting (C.18) in (C.16), we have

$$\sum_{t=1}^{T} w_h^2 Y^2(||\mathbf{p}_t||_2^2 + M_{max}) \leq w_h^2 Y^2(\log(T+N_0-1) - \log(N_0)) + w_h^2 M_{max} Y^2 T. \tag{C.19}$$

On dividing by T and taking its limit to infinite, we have

$$\lim_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} w_h^2 Y^2(||\mathbf{p}_t||_2^2 + M_{max}) \leq w_h^2 M_{max} Y^2 \tag{C.20}$$

where the first term vanishes as T grows faster than $\log(T)$.

[1] For a function $f(t) > 0$, we have $\sum_{t=1}^{T} f(t)dt \leq \text{in}_{t=1}^{T} f(t)dt$.

For the second term we have

$$\begin{aligned}\sum_{t=1}^{T} 2Rw_hY\sqrt{||\mathbf{p}_t||_2^2+M_{max}} &\leq 2Rw_hY\sum_{t=1}^{T}\frac{1}{2}(||\mathbf{p}_t||_2^2+M_{max}+1)\\ &\leq Rw_hY(\sum_{t=1}^{T}||\mathbf{p}_t||_2^2+T(M_{max}+1))\end{aligned} \tag{C.21}$$

where we use the fact that the geometric mean is lesser than or equal to the arithmetic mean. Utilizing the fact that $\lim_{T\to\infty}\frac{1}{T}\sum_{t=1}^{T}||\mathbf{p}_t||_2^2$ is equal to zero (as shown above in (C.20)), we have

$$\lim_{T\to\infty}\frac{1}{T}\sum_{t=1}^{T}2Rw_hY\sqrt{||\mathbf{p}_t||_2^2+M_{max}}\leq Rw_hY(M_{max}+1). \tag{C.22}$$

For the third term $w_h^2Y^2\bar{\sigma}_t^2$, we have

$$\sum_{t=1}^{T}w_h^2\bar{\sigma}_t^2||\bar{\mathbf{y}}_t||_2^2\leq w_h^2Y\sum_{t=1}^{T}\frac{1}{N_{t-1}}(1-\frac{1}{N_{t-1}})\leq w_h^2Y\sum_{t=1}^{T}\frac{1}{N_{t-1}} \tag{C.23}$$

which holds for the uniformly at random attachment rule. Given $\lim_{T\to\infty}\frac{1}{T}\sum_{t=1}^{T}\frac{1}{N_{t-1}}=0$, the third term vanishes in the limit. Adding (C.20) and (C.22), along with the other terms from the stochastic regret bound, we have the required bound for Corollary 3. □

C.3. PROOF OF COROLLARY 4

To prove this corollary, we start with the result of Proposition 2. For the first term, we have $||\mathbf{p}_t||_2^2=||\mathbf{P}_{t-1}\mathbf{m}(t-1)||_2^2$, which is upper bounded as $||\mathbf{m}(t-1)||_2^2||\mathbf{P}_{t-1}||_F^2$. The maximum value of $||\mathbf{m}(t-1)||_2^2$ is one for $\mathbf{m}(t-1)\in\mathscr{S}_M$.

For the second term, we use the Arithmetic Mean-Geometric Mean inequality as in Corollary 3 and use again $||\mathbf{p}_t||_2^2\leq||\mathbf{P}_{t-1}||_F^2$. Similarly, for the third term we have

$$\begin{aligned}\bar{\sigma}_t^2 &= \max_{n=1:N_{t-1}}[\mathbf{p}_t]_n(1-[\mathbf{p}_t]_n)\max_{n=1:N_{t-1}}[\mathbf{P}_{t-1}\mathbf{m}(t-1)]_n(1-[\mathbf{P}_{t-1}\mathbf{m}(t-1)]_n)\\ &\leq \max_{n=1:N_{t-1}}[\mathbf{P}_{t-1}\mathbf{m}(t-1)]_n\leq\bar{P}_t||\mathbf{m}(t-1)||_2\leq P_t.\end{aligned} \tag{C.24}$$

Substituting $\bar{\sigma}_t^2$ in the regret for the stochastic setting, we complete the proof. □

C.4. RELEVANT DERIVATIONS

Lemma 3 *Under Assumption 1 and 6, the loss function $l_t(\mathbf{h},x_t)$ is Lipschitz in $\mathbf{h}$. That is, the ℓ_2 norm of the gradient of the loss at time t is upper-bounded as*

$$||\nabla_h l_t(\mathbf{h},x_t)||_2\leq L_d \tag{C.25}$$

where $L_d=(RC+2\mu H)$

Proof. We apply the Cauchy-Schawrtz inequality on the r.h.s. of (5.11) and get

$$||\nabla_h|_t(\mathbf{h}, x_t)||_2 \leq |\mathbf{a}_t^\top \mathbf{A}_{x,t-1}\mathbf{h} - x_t|||\mathbf{A}_{x,t-1}^\top \mathbf{a}_t||_2 + 2\mu||\mathbf{h}||_2 \leq RC + 2\mu H. \tag{C.26}$$

where we use Assumption 1. □

Lemma 4 *At time t, given the probability of attachment $\mathbf{p}_t$, the weight $\mathbf{w}_t$, and the attachment $\mathbf{a}_t$. Let N_{t-1} be the number of existing nodes. We have*

$$||\mathbf{w}_t \circ \mathbf{p}_t - \mathbf{a}_t||_2^2 \leq w_h^2(||\mathbf{p}_t||_2^2 + M_{max}). \tag{C.27}$$

Proof. We can write the squared norm as

$$||\mathbf{w}_t \circ \mathbf{p}_t - \mathbf{a}_t||_2^2 = \sum_{n=1}^{N_{t-1}} [\mathbf{w}_t]_n^2 [\mathbf{p}_t]_n^2 - 2[\mathbf{w}_t]_n [\mathbf{p}_t]_n [\mathbf{a}_t]_n + [\mathbf{a}_t]_n^2. \tag{C.28}$$

The second term in this summation is always negative, so we can write

$$||\mathbf{w}_t \circ \mathbf{p}_t - \mathbf{a}_t||_2^2 \leq \sum_{n=1}^{N_{t-1}} [\mathbf{w}_t]_n^2 [\mathbf{p}_t]_n^2 + [\mathbf{a}_t]_n^2. \tag{C.29}$$

Note that $\sum_{n=1}^{N_{t-1}} [\mathbf{a}_t]_n^2 \leq M_{max} w_h^2$ using Assumptions 1 and 5; thus, we have

$$||\mathbf{w}_t \circ \mathbf{p}_t - \mathbf{a}_t||_2^2 \leq w_h^2 ||\mathbf{p}_t||_2^2 + M_{max} w_h^2. \tag{C.30}$$

□

Lemma 5 *The term $||\mathbf{A}_{x,t-1}\mathbf{h}||_2$ is bounded in its ℓ_2 norm for all t, i.e., $||\mathbf{A}_{x,t-1}\mathbf{h}||_2 \leq Y$.*

Proof. From the expression of $\tilde{\mathbf{y}}_t$ in (5.5), we have

$$\begin{aligned} ||\tilde{\mathbf{y}}_t||_2 &\leq ||\sum_{k=0}^{K} h_k \mathbf{A}_{t-1}^k \mathbf{x}_t||_2 + ||\mathbf{a}_t^\top \sum_{k=1}^{K} h_k \mathbf{A}_{t-1}^{k-1} \mathbf{x}_t||_2 \\ &\leq ||\sum_{k=0}^{K} h_k \mathbf{A}_{t-1}^k \mathbf{x}_t||_2 + ||\mathbf{a}_t||_2 ||\sum_{k=1}^{K} h_k \mathbf{A}_{t-1}^{k-1} \mathbf{x}_t||_2. \end{aligned} \tag{C.31}$$

The first term on the r.h.s. is bounded for a bounded $\mathbf{h}$ and $\mathbf{A}_{t-1}$. So is the second term, following Assumptions 1 and 5. Thus both the output $\tilde{\mathbf{y}}_t$ and $\sum_{k=1}^{K} h_k \mathbf{A}_{t-1}^{k-1} \mathbf{x}_t$ are bounded. We denote the bound for $||\sum_{k=1}^{K} h_k \mathbf{A}_{t-1}^{k-1} \mathbf{x}_t||_2$ as Y. □

C.4.1. GRADIENTS

Here we provide the expressions for the gradients w.r.t $\mathbf{h}$, $\mathbf{m}$, and $\mathbf{n}$ for the adaptive stochastic online learner.

$$\nabla_h|_t^s(\mathbf{h}, \mathbf{m}, \mathbf{n}, x_t) = ((\mathbf{W}_{t-1}\mathbf{n} \circ \mathbf{P}_{t-1}\mathbf{m})^\top \mathbf{A}_{x,t-1}\mathbf{h} - x_t)\mathbf{A}_{x,t-1}^\top (\mathbf{w}_t \circ \mathbf{p}_t) + \mathbf{A}_{x,t-1}^\top \bar{\mathbf{\Sigma}}_t \mathbf{A}_{x,t-1}\mathbf{h} + 2\mu\mathbf{h}. \tag{C.32}$$

The gradient w.r.t. $\mathbf{m}$ is

$$\begin{aligned}\nabla_m|_t(\mathbf{h},\mathbf{m},\mathbf{n},x_t) &= \mathbf{P}_{t-1}^\top(\mathbf{W}_{t-1}\mathbf{n}\circ\mathbf{A}_{x,t-1}\mathbf{h})(\mathbf{W}_{t-1}\mathbf{n}\circ\mathbf{P}_{t-1}\mathbf{m})^\top\mathbf{A}_{x,t-1}\mathbf{h}-x_t)\\ &+\mathbf{P}_{t-1}^\top((\mathbf{A}_{x,t-1}\mathbf{h})^{\circ 2}\circ(\mathbf{W}_{t-1}\mathbf{n})^{\circ 2})-2\mathbf{P}_{t-1}^\top(\mathbf{P}_{t-1}\mathbf{m}\circ(\mathbf{A}_{x,t-1}\mathbf{h})^{\circ 2}\circ(\mathbf{W}_{t-1}\mathbf{n})^{\circ 2}).\end{aligned} \tag{C.33}$$

Finally, the gradient w.r.t $\mathbf{n}$ is

$$\nabla_n|_t(\mathbf{h},\mathbf{m},\mathbf{n},x_t) = \mathbf{W}_t^\top(\mathbf{P}_{t-1}\mathbf{m}\circ\mathbf{A}_{x,t-1}\mathbf{h})((\mathbf{W}_{t-1}\mathbf{n}\circ\mathbf{P}_{t-1}\mathbf{m})^\top\mathbf{A}_{x,t-1}\mathbf{h}-x_t). \tag{C.34}$$

C

□

Computational complexity. Here we provide the computational complexity of the online learners. All methods rely on computing $\mathbf{A}_{x,t-1}=[\tilde{\mathbf{x}}_t,\mathbf{A}_{t-1}\tilde{\mathbf{x}}_t,\ldots,\mathbf{A}_{t-1}^{K-1}\tilde{\mathbf{x}}_t]$ at time t. We construct $\mathbf{A}_{x,t-1}$ with a complexity of order $\mathcal{O}(M_{t-1}K)$ formed by shifting $\tilde{\mathbf{x}}_t$ $K-1$ times over $\mathcal{G}_{t-1}$. At time $t-1$, we need not calculate $\mathbf{A}_{x,t}=[\tilde{\mathbf{x}}_{t+1},\mathbf{A}_t\tilde{\mathbf{x}}_{t+1},\ldots,\mathbf{A}_t^{K-1}\tilde{\mathbf{x}}_{t+1}]$ from scratch. From the structure of $\mathbf{A}_t$ [cf. (5.1)], we only need to calculate the diffusions over the incoming edges $K-1$ times, which amounts to an additional complexity of $\mathcal{O}(M_{\max}K)$. The computational complexity of each online method is detailed next.

D-OGF. The complexity of update (5.10) is governed by the gradient, which depends on the output at time $\hat{x}_t=\mathbf{a}_t^\top\mathbf{A}_{x,t-1}\mathbf{h}$. It has a complexity of $\mathcal{O}(M_{\max}K+M_{t-1}K)$, where the complexity for $\mathbf{A}_{x,t-1}\mathbf{h}$ is $M_{t-1}K$ and that for the diffusion over the newly formed edges is $\mathcal{O}(M_{\max}K)$.

S-OGF. The only difference between S-OGF and D-OGF is that we use the expected attachment vector $(\mathbf{w}_t\circ\mathbf{p}_t)$ to calculate the output, which means the complexity incurred depends on N_{t-1} for all t.

Ada-OGF. The Ada-OGF, being a stochastic approach has already a complexity in the order of $\mathcal{O}(K(M_0+N_t))$. However, it has an extra complexity of $\mathcal{O}(N_t(M))$ due to both the $\mathbf{m}$ and $\mathbf{n}$ update.

D

APPENDIX D

D.1. CONVERGENCE PROOFS

Strong Convexity of $f(\cdot)$ in $\mathbf{A}_r$. The derivative of $f(\mathbf{A}_r)$ w.r.t $\mathbf{A}_r$ writes as

$$\begin{aligned}\nabla_{\mathbf{A}_r} f(\mathbf{A}_r) = \mathbf{A}_r(\sum_{t=1}^{T} C_{tr}^2 \mathbf{1}^\top \mathbf{m}_t) + [\sum_{t=1}^{T} C_{tr}\mathbf{1}^\top \mathbf{m}_t \sum_{\bar{r}=1,\bar{r}\neq r}^{R} C_{t\bar{r}}\mathbf{A}_{\bar{r}}] \\ - [\sum_{t=1}^{T} \underline{\mathbf{A}}_{:,:,t} C_{tr}\mathbf{1}^\top \mathbf{m}_t] + \delta\Xi_r^\top + 2\beta \sum_{k=1,k\neq r}^{R} \mathbf{A}_k + \gamma \mathbf{1}\mathbf{1}^\top.\end{aligned} \tag{D.1}$$

For strong convexity we need the Hessian $\nabla^2_{\mathbf{A}_r} f \succ 0$. First we write

$$\mathrm{d}\nabla_{\mathbf{A}_r} f = \big((\sum_{t=1}^{T} C_{tr}^2 \mathbf{1}^\top \mathbf{m}_t)\big)\mathrm{d}\mathbf{A}_0. \tag{D.2}$$

Doing vectorization on both sides gives us

$$\begin{aligned}\mathrm{vec}(\mathrm{d}\nabla_{\mathbf{A}_r} f) &= \big((\sum_{t=1}^{T} C_{tr}^2 \mathbf{1}^\top \mathbf{m}_t)\big)\mathrm{vec}(\mathrm{d}\mathbf{A}_r), \\ \nabla^2_{\mathbf{A}_r} f &= \big(\sum_{t=1}^{T} C_{tr}^2 \mathbf{1}^\top \mathbf{m}_t\big)\mathbf{I}_{N^2},\end{aligned} \tag{D.3}$$

with $\nabla^2_{\mathbf{A}_r} f \in \mathbb{R}^{N^2\times N^2}$. From Assumption 8, we have strong convexity in $\mathbf{A}_r$.

Strong Convexity of $f(\cdot)$ in $\mathbf{C}$. For $\mathbf{C}$ we have

$$\nabla_{\mathbf{C}} f(\mathbf{C}) = \mathbf{F}(-\mathbf{A}_{\mathrm{vec}}^\top \mathbf{A}_0 + \mathbf{C}\mathbf{A}_0^\top \mathbf{A}) + \delta(\bar{\mathbf{Z}}_T \mathbf{A}_0) + \mu\mathbf{D}^\top\mathbf{D} + \rho\mathbf{C}. \tag{D.4}$$

The Hessian w.r.t $\mathbf{C}$ can similarly be derived as

$$\begin{aligned}\mathrm{d}\nabla_{\mathbf{C}} f &= \mathbf{F}\mathrm{d}\mathbf{C}(\mathbf{A}_0^\top\mathbf{A}_0) + (\mathbf{D}^\top\mathbf{D} + \rho\mathbf{F})\mathrm{d}\mathbf{C} \\ \mathrm{vec}(\mathrm{d}\nabla_{\mathbf{C}} f) &= (\mathbf{A}_0^\top\mathbf{A}_0 \otimes \mathbf{F})\mathrm{vec}(\mathrm{d}\mathbf{C}) + \mathbf{I}_R \otimes (\mathbf{D}^\top\mathbf{D} + \rho\mathbf{I}_T)\mathrm{vec}(\mathrm{d}\mathbf{C}) \\ \nabla^2_{\mathbf{C}} f &= \big(\mathbf{A}_0^\top\mathbf{A}_0 \otimes \mathbf{F} + \mathbf{I}_R \otimes \mathbf{D}^\top\mathbf{D} + \rho\mathbf{I}_{RT}\big)\end{aligned} \tag{D.5}$$

where $\nabla^2_{\mathbf{C}} f \in \mathbb{R}^{TR\times TR}$. The matrix $\left(\mathbf{A}_0^\top\mathbf{A}_0 \otimes \mathbf{F} + \mathbf{I}_R \otimes \mathbf{D}^\top\mathbf{D} + \rho\mathbf{I}_{RT}\right)$ is composed of thee components. First, $\mathbf{A}_0^\top\mathbf{A}_0 \otimes \mathbf{F}$ is a positive semi-definite matrix as both the matrices $\mathbf{A}_0^\top\mathbf{A}_0$ and $\mathbf{F}$ (which is a diagonal matrix with positive diagonal elements) are positive semi-definite. By the same argument, $\mathbf{I}_R \otimes \mathbf{D}^\top\mathbf{D}$ is positive semi-definite. Since $\rho > 0$, we have $\mathbf{A}_0^\top\mathbf{A}_0 \otimes \mathbf{F}$ is a positive semi-definite matrix as both the matrices $\mathbf{A}_0^\top\mathbf{A}_0$ as positive definite, which means $\nabla^2_{\mathbf{C}} f \geq \rho\mathbf{I}_{RT}$, establishing strong convexity in $\mathbf{C}$. □

D.2. PROOF OF PROPOSITION

We prove the convergence of our alternating approach to a stationary point. We denote $\mathbf{A}_{-r}$ as the set of all $\mathbf{A}_i$s with $i \neq r$.

Convergence over $\mathbf{A}_r$s. To show that we converge to a block coordindate minimizer for this problem, we must satisfy the requirements mentioned in Theorem 2.3, Lemma 2.2, and Assumptions 1 and 2 in [176]. These are as follows.

- $f(\mathbf{A}_1,\ldots,\mathbf{A}_R,\mathbf{C})$ is block-multiconvex, i.e., it is convex in each $\mathbf{A}_r$ and $\mathbf{C}$, with the other block elements being fixed. This can be seen from Appendix D.1. In fact, we have strong convexity for each block element.
- Block multiconvexity, i.e., the set which corresponds to the constraint of each block element should be convex, given the others are fixed. We have $\sum_{r=1}^R \mathbf{A}_r\mathbf{\Phi}_r - \zeta\mathbf{1}_{N\times T} \geq \mathbf{0}_{N\times T}$ which is block multi-convex, i.e., for fixed $\mathbf{A}_{-r}$ and $\mathbf{C}$, the set satisfying $\mathbf{A}_r\mathbf{\Phi}_r + \mathbf{\Gamma}_r \geq \mathbf{0}$ is convex in $\mathbf{A}_r$. The same holds for $\mathbf{C}$ with the constraint set for $\mathbf{A}[\mathbf{C}^\top \otimes \mathbf{1}_N] \geq \zeta\mathbf{1}_{N\times T}$ being convex in $\mathbf{C}$ for fixed $\mathbf{A}_r$s.
- The minimum of $f()$ w.r.t $\mathbf{A}_r$ exists and is finite. This is verified because for all $\mathbf{A}_r$ which lies in the feasible set, the loss function $f()$ is greater than or equal to zero. Similarly, the minimum of $f()$ w.r.t $\mathbf{C}$ is also finite, following the same reasoning, i.e., element-wise positivity.
- The set map w.r.t each $\mathbf{A}_r$ changes continuously [cf. [176]]. The constraint $\mathbf{A}_r\mathbf{\Phi}_r + \mathbf{\Gamma}_r \geq \mathbf{0}$ which changes for every update is continuous in each $\mathbf{A}_r$. The constraint set $\mathbf{A}[\mathbf{C}^\top \otimes \mathbf{1}_N] \geq \zeta\mathbf{1}_{N\times T}$ changes in a continuous manner for each update in $\mathbf{C}$, given the changes in $\mathbf{A}_r$. This is easily verified, given our alternating update scheme.

Therefore, the proposed alternating approach converges to a block coordinate minimizer in the latent adjacency matrices $\mathbf{A}_r$s and their temporal signatures in $\mathbf{C}$. □

D.3. GRADIENTS FOR ADMM UPDATES

The following is the gradient expression used in (6.15):

$$\begin{aligned}\nabla_{\mathbf{A}_r} L(\mathbf{A}_r) = \mathbf{A}_r\Big(\sum_{t=1}^T C_{tr}^2\mathbf{1}^\top\mathbf{m}_t\Big) + \Big[\sum_{t=1}^T C_{tr}\mathbf{1}^\top\mathbf{m}_t \sum_{\bar{r}=1,\bar{r}\neq r}^R C_{t\bar{r}}\mathbf{A}_{\bar{r}}\Big] \\ - \Big[\sum_{t=1}^T \underline{\mathbf{A}}_{:,:,t} C_{tr}\,\mathrm{tr}(\mathrm{diag}(\mathbf{M}_t))\Big] + \delta\mathbf{\Xi}_r^\top + 2\beta\sum_{k=1,k\neq r}^R \mathbf{A}_k + \gamma\mathbf{1}\mathbf{1}^\top \\ + \eta\mathbf{A}_r + \mathbf{\Lambda}_r^\top\mathbf{\Phi}_r^\top + \lambda(\mathbf{A}_r\mathbf{\Phi}_r + \mathbf{\Gamma}_r - \mathbf{P})\mathbf{\Phi}_r^\top.\end{aligned} \tag{D.6}$$

The gradient of the augmented Lagrangian w.r.t. $\mathbf{C}$ as used in (6.24) is

$$\nabla_{\mathbf{C}} L(\mathbf{C}) = \mathbf{F}(-\mathbf{A}_{\text{vec}}^{\top}\mathbf{A} + \mathbf{C}\mathbf{A}^{\top}\mathbf{A}) + \delta(\bar{\mathbf{Z}}_T\mathbf{A}) + \mu\mathbf{D}^{\top}\mathbf{D}\mathbf{C} + \rho\mathbf{C} + \mathbf{\Lambda}^{\top}\Upsilon + \lambda_c(\mathbf{C}\Upsilon^{\top} - \zeta\mathbf{1}_{T,N} - \mathbf{Q})\Upsilon, \tag{D.7}$$

where $\mathbf{F} = \text{diag}(\mathbf{f}) = \text{diag}(\mathbf{1}_{N^2}^{\top}\mathbf{M})$, i.e., the diagonal elements of $\mathbf{F}$ measure the number of active observations at that time.

D.4. COMPONENT-WISE F1 SCORES

The following are the F1 scores corresponding to the RE scores as shown in Figure 6.7 in Section 6.5:

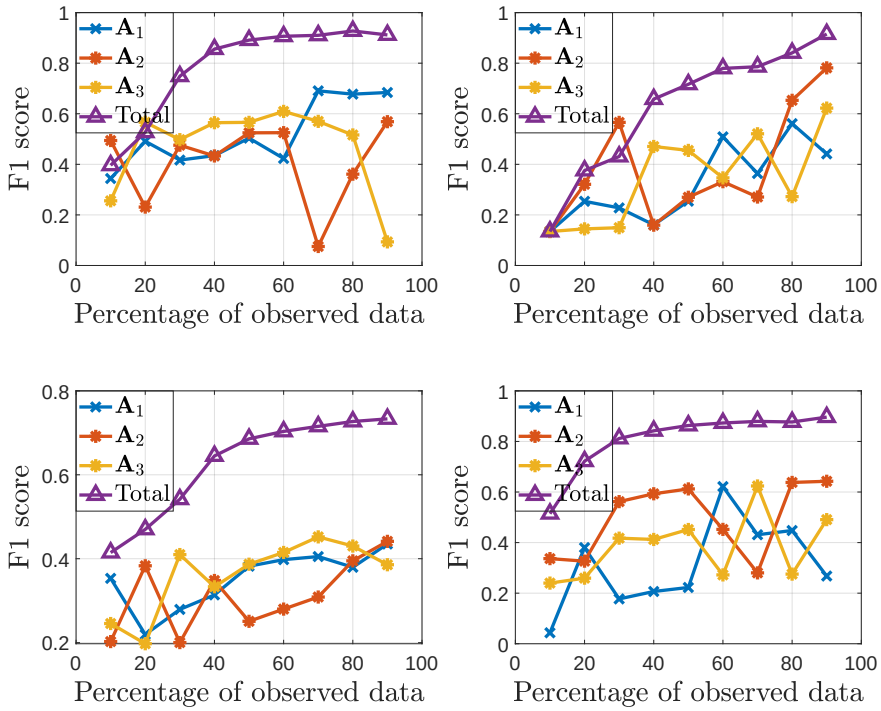


Figure D.1.: F1 of each of the $R = 3$ individual modes (blue, orange and yellow) for each data set compared to that of their model-based combination (purple).

BIBLIOGRAPHY

[1] S.-H. Yook, Z. N. Oltvai and A.-L. Barabási. 'Functional and topological characterization of protein interaction networks'. In: *Proteomics* 4.4 (2004), pp. 928–942.

[2] C. C. Aggarwal. 'An introduction to social network data analytics'. In: *Social network data analytics*. Springer, 2011, pp. 1–15.

[3] K. Nagel. 'Traffic networks'. In: *Handbook of graphs and networks: From the genome to the internet* (2002), pp. 248–272.

[4] S. A. Greenberg. 'How citation distortions create unfounded authority: analysis of a citation network'. In: *Bmj* 339 (2009).

[5] A. Ortega, P. Frossard, J. Kovaevi, J. M. Moura and P. Vandergheynst. 'Graph signal processing: Overview, challenges, and applications'. In: *Proceedings of the IEEE* 106.5 (2018), pp. 808–828.

[6] 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'. In: *IEEE Signal Process. Mag.* 30.3 (2013), pp. 83–98.

[7] E. Isufi, F. Gama, D. I. Shuman and S. Segarra. 'Graph filters for signal processing and machine learning on graphs'. In: *IEEE Trans. Signal Process.* (2024).

[8] A. Ortega, P. Frossard, J. Kovaevi, J. M. F. Moura and P. Vandergheynst. 'Graph Signal Processing: Overview, Challenges, and Applications'. In: *Proc. IEEE* 106.5 (May 2018), pp. 808–828.

[9] X. He, K. Deng, X. Wang, Y. Li, Y. Zhang and M. Wang. 'Lightgcn: Simplifying and powering graph convolution network for recommendation'. In: *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*. 2020, pp. 639–648.

[10] W. Huang, A. G. Marques and A. R. Ribeiro. 'Rating Prediction via Graph Signal Processing'. In: *IEEE Trans. Signal Process.* 66.19 (Oct. 2018), pp. 5066–5081.

[11] P. Holme and J. Saramäki. 'Temporal networks'. In: *Phy. Rep.* 519.3 (2012), pp. 97–125.

[12] X. Li, M. Zhang, S. Wu, Z. Liu, L. Wang and S. Y. Philip. 'Dynamic graph collaborative filtering'. In: *IEEE Intl. Conf. Dat. Min.* IEEE. 2020, pp. 322–331.

[13] P. Erdos. 'On the evolution of random graphs'. In: *Bullet. Inst. Inter. Stat.* 38 (1961), pp. 343–347.

[14] A. L. Barabási and R. Albert. 'Emergence of Scaling in Random Networks'. en. In: *Science* 286.5439 (Oct. 1999). Publisher: American Association for the Advancement of Science. (Visited on 10/11/2020).

[15] M. Newman. *Networks*. Oxford university press, 2018.

[16] X. N. Lam, T. Vu, T. D. Le and A. D. Duong. 'Addressing cold-start problem in recommendation systems'. In: *Proceedings of the 2nd international conference on Ubiquitous information management and communication*. 2008, pp. 208–211.

[17] A. I. Schein, A. Popescul, L. H. Ungar and D. M. Pennock. 'Methods and metrics for cold-start recommendations'. In: *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*. SIGIR '02. New York, NY, USA, Aug. 2002, pp. 253–260.

[18] S. Liu, I. Ounis, C. Macdonald and Z. Meng. 'A Heterogeneous Graph Neural Model for Cold-start Recommendation'. In: *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '20. New York, NY, USA: Association for Computing Machinery, July 2020, pp. 2029–2032. (Visited on 08/10/2020).

[19] A.-L. Barabási *et al. Network science*. Cambridge university press, 2016.

[20] P. Di Lorenzo, P. Banelli, E. Isufi, S. Barbarossa and G. Leus. 'Adaptive graph signal processing: Algorithms and optimal sampling strategies'. In: *IEEE Trans. Signal Process.* 66.13 (2018), pp. 3584–3598.

[21] A. Karaaslanli and S. Aviyente. 'Multiview Graph Learning with Consensus Graph'. In: *arXiv preprint arXiv:2401.13769* (2024).

[22] Y. He and H.-T. Wai. 'Online inference for mixture model of streaming graph signals with sparse excitation'. In: *IEEE Trans. Signal Process.* 70 (2022), pp. 6419–6433.

[23] B. Park, D. S. Kim and H. J. Park. 'Graph independent component analysis reveals repertoires of intrinsic network components in the human brain'. In: *PloS one* 9.1 (2014), e82873.

[24] M. Pedersen, A. Zalesky, A. Omidvarnia and G. D. Jackson. 'Multilayer network switching rate predicts brain performance'. In: *Proc. Nat. Aca. Sci.* 115.52 (2018), pp. 13376–13381.

[25] F. Gama, E. Isufi, A. Ribeiro and G. Leus. 'Controllability of bandlimited graph processes over random time varying graphs'. In: *IEEE Trans. Signal Process.* 67.24 (2019), pp. 6440–6454.

[26] E. Isufi, A. Loukas, A. Simonetto and G. Leus. 'Filtering random graph processes over random time-varying graphs'. In: *IEEE Trans. Signal Process.* 65.16 (2017), pp. 4406–4421.

[27] G. Bianconi and A.-L. Barabási. 'Competition and multiscaling in evolving networks'. en. In: *EPL* 54.4 (May 2001), p. 436. (Visited on 10/11/2020).

[28] Y. Shen, G. Leus and G. B. Giannakis. 'Online Graph-Adaptive Learning With Scalability and Privacy'. In: *IEEE Trans. Signal Process.* 67.9 (May 2019), pp. 2471–2483.

[29] Z. Yang, W. Cohen and R. Salakhudinov. 'Revisiting semi-supervised learning with graph embeddings'. In: *International conference on machine learning*. PMLR. 2016, pp. 40–48.

D

[30] F. Dornaika, R. Dahbi, A. Bosaghzadeh and Y. Ruichek. 'Efficient dynamic graph construction for inductive semi-supervised learning'. In: *Neural Networks* 94 (2017), pp. 192–203.

[31] S. Chen, F. Cerda, P. Rizzo, J. Bielak, J. H. Garrett and J. Kovaevi. 'Semi-Supervised Multiresolution Classification Using Adaptive Graph Filtering With Application to Indirect Bridge Structural Health Monitoring'. In: *IEEE Trans. Signal Process.* 62.11 (June 2014). Conference Name: IEEE Transactions on Signal Processing, pp. 2879–2893.

[32] L. Gauvin, A. Panisson and C. Cattuto. 'Detecting the community structure and activity patterns of temporal networks: a non-negative tensor factorization approach'. In: *PloS one* 9.1 (2014), e86028.

[33] A. Gorovits, E. Gujral, E. E. Papalexakis and P. Bogdanov. 'Larc: Learning activity-regularized overlapping communities across time'. In: *Intl. Conf. Knowledge Discovery Data Mining*. 2018, pp. 1465–1474.

[34] S. Fernandes, H. Fanaee-T and J. Gama. 'Dynamic graph summarization: a tensor decomposition approach'. In: *Springer Dat. Min. Know. Discovery* 32 (2018), pp. 1397–1420.

[35] Q. Wang. 'Large-scale Dynamic Network Representation via Tensor Ring Decomposition'. In: *arXiv preprint arXiv:2304.08798* (2023).

[36] E. Gujral, R. Pasricha and E. Papalexakis. 'Beyond rank-1: Discovering rich community structure in multi-aspect graphs'. In: *Proc. Web Conf. 2020*. 2020, pp. 452–462.

[37] D. M. Dunlavy, T. G. Kolda and E. Acar. 'Temporal link prediction using matrix and tensor factorizations'. In: *ACM Trans. Know. Disc. Data* 5.2 (2011), pp. 1–27.

[38] A. Sapienza, A. Panisson, J. T. K. Wu, L. Gauvin and C. Cattuto. 'Anomaly detection in temporal graph data: An iterative tensor decomposition and masking approach'. In: *Intl. Work. Adv. Anal. Learn. Temp. Data*. 2015.

[39] A. G. Mahyari, D. M. Zoltowski, E. M. Bernat and S. Aviyente. 'A tensor decomposition-based approach for detecting dynamic network states from EEG'. In: *IEEE Trans. Biom. Engg.* 64.1 (2016), pp. 225–237.

[40] A. Ozdemir, E. M. Bernat and S. Aviyente. 'Recursive tensor subspace tracking for dynamic brain network analysis'. In: *IEEE Trans. Signal Inf. Process. Netw.* 3.4 (2017), pp. 669–682.

[41] L. De Lathauwer. 'Decompositions of a higher-order tensor in block termsPart I: Lemmas for partitioned matrices'. In: *SIAM Journal on Matrix Analysis and Applications* 30.3 (2008), pp. 1022–1032.

[42] J. C. Mitchell. 'Social networks'. In: *Annual review of anthropology* 3.1 (1974), pp. 279–299.

[43] M. Tubaishat and S. Madria. 'Sensor networks: an overview'. In: *IEEE potentials* 22.2 (2003), pp. 20–23.

[44] D. Zhao and A. Strotmann. *Analysis and visualization of citation networks*. Morgan & Claypool Publishers, 2015.

[45] F. R. Chung. *Spectral graph theory*. Vol. 92. American Mathematical Soc., 1997.

[46] A. Sandryhaila and J. M. F. Moura. 'Discrete signal processing on graphs: Frequency analysis'. In: *IEEE Trans. Signal Process.* 62.12 (2014), pp. 3042–3054.

[47] B. Xie, M. Wang and D. Tao. 'Toward the optimization of normalized graph Laplacian'. In: *IEEE Trans. Neural Netw.* 22.4 (2011), pp. 660–666.

[48] F. Bauer. 'Normalized graph Laplacians for directed graphs'. In: *Linear Algebra and its Applications* 436.11 (2012), pp. 4193–4222.

[49] J. Lyklema, C. Evertsz and L. Pietronero. 'The Laplacian random walk'. In: *Europh. Let.* 2.2 (1986), p. 77.

[50] X. Liu, G. Cheung, X. Wu and D. Zhao. 'Random walk graph Laplacian-based smoothness prior for soft decoding of JPEG images'. In: *IEEE Trans. Imag. Proc.* 26.2 (2016), pp. 509–524.

[51] G. Mateos, S. Segarra, A. G. Marques and A. Ribeiro. 'Connecting the dots: Identifying network structure via graph signal processing'. In: *IEEE Signal Process. Mag.* 36.3 (2019), pp. 16–43.

[52] S. Chen, A. Sandryhaila, J. M. Moura and J. Kovacevic. 'Signal denoising on graphs via graph filtering'. In: *Global Conf. Signal and Info. Process. (GlobalSIP)*. IEEE. 2014, pp. 872–876.

[53] Y. Ma, X. Liu, T. Zhao, Y. Liu, J. Tang and N. Shah. 'A unified view on graph neural networks as graph signal denoising'. In: *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 2021, pp. 1202–1211.

[54] S. Chen, Y. C. Eldar and L. Zhao. 'Graph unrolling networks: Interpretable neural networks for graph signal denoising'. In: *IEEE Trans. Signal Process.* 69 (2021), pp. 3699–3713.

[55] M. Onuki, S. Ono, M. Yamagishi and Y. Tanaka. 'Graph signal denoising via trilateral filter on graph spectral domain'. In: *IEEE Trans. Signal Inf. Process. Netw.* 2.2 (2016), pp. 137–148.

[56] S. K. Narang, A. Gadde and A. Ortega. 'Signal processing techniques for interpolation in graph structured data'. In: *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE. 2013, pp. 5445–5449.

[57] D. Berberidis, A. N. Nikolakopoulos and G. B. Giannakis. 'Adaptive diffusions for scalable learning over graphs'. In: *IEEE Trans. Signal Process.* 67.5 (2018), pp. 1307–1321.

[58] S. Chen, F. Cerda, P. Rizzo, J. Bielak, J. H. Garrett and J. Kovaevi. 'Semi-supervised multiresolution classification using adaptive graph filtering with application to indirect bridge structural health monitoring'. In: *IEEE Trans. Signal Process.* 62.11 (2014), pp. 2879–2893.

D

[59] S. Segarra, A. G. Marques, G. Leus and A. Ribeiro. 'Interpolation of graph signals using shift-invariant graph filters'. In: *European Signal Process. Conf. (EUSIPCO)*. IEEE. 2015, pp. 210–214.

[60] Q. Li, X.-M. Wu, H. Liu, X. Zhang and Z. Guan. 'Label efficient semi-supervised learning via graph filtering'. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 9582–9591.

[61] L. Xiang, Q. Yuan, S. Zhao, L. Chen, X. Zhang, Q. Yang and J. Sun. 'Temporal recommendation on graphs via long-and short-term preference fusion'. In: *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2010, pp. 723–732.

[62] X. Zhang, C. Moore and M. E. J. Newman. 'Random graph models for dynamic networks'. In: *Eur. Phy. Journ. B* 90 (2017), pp. 1–14.

[63] J. Lindquist, J. Ma, P. Van den Driessche and F. H. Willeboordse. 'Network evolution by different rewiring schemes'. In: *Phy. D: Nonl. Phen.* 238.4 (2009), pp. 370–378.

[64] Y. B. Xie, T. Zhou and B. H. Wang. 'Scale-free networks without growth'. In: *Phy. A: Stat. Mech. Appl.* 387.7 (2008), pp. 1683–1688.

[65] E. Isufi, M. Pocchiari and A. Hanjalic. 'Accuracy-diversity trade-off in recommender systems via graph convolutions'. In: *Inf. Proc. Manag.* 58.2 (2021), p. 102459.

[66] A. I. Schein, A. Popescul, L. H. Ungar and D. M. Pennock. 'Methods and metrics for cold-start recommendations'. In: *Proc. ACM SIGIR Conf. Res. Dev. Inf. Ret.* 2002, pp. 253–260.

[67] A.-L. Barabási and R. Albert. 'Emergence of scaling in random networks'. In: *science* 286.5439 (1999), pp. 509–512.

[68] R. A. Rossi and N. K. Ahmed. 'The Network Data Repository with Interactive Graph Analytics and Visualization'. In: (2015).

[69] J. Fournet and A. Barrat. 'Contact patterns among high school students'. In: *PloS one* 9.9 (2014), e107878.

[70] A. Cichocki, N. Lee, I. Oseledets, A. H. Phan, Q. Zhao, D. P. Mandic *et al.* 'Tensor networks for dimensionality reduction and large-scale optimization: Part 1 low-rank tensor decompositions'. In: *Found. Tren. Mach. Learn.* 9.4-5 (2016), pp. 249–429.

[71] T. G. Kolda and B. W. Bader. 'Tensor decompositions and applications'. In: *SIAM rev.* 51.3 (2009), pp. 455–500.

[72] L. De Lathauwer and D. Nion. 'Decompositions of a higher-order tensor in block termsPart III: Alternating least squares algorithms'. In: *SIAM Jour. Matr. Anal. Appl.* 30.3 (2008), pp. 1067–1083.

[73] L. De Lathauwer, B. De Moor and J. Vandewalle. 'A multilinear singular value decomposition'. In: *SIAM Jour. Matr. Anal. Appl.* 21.4 (2000), pp. 1253–1278.

[74] I. V. Oseledets. 'Tensor-train decomposition'. In: *SIAM Journal on Scientific Computing* 33.5 (2011), pp. 2295–2317.

[75] Q. Zhao, G. Zhou, S. Xie, L. Zhang and A. Cichocki. 'Tensor ring decomposition'. In: *arXiv preprint arXiv:1606.05535* (2016).

[76] G. Mateos, S. Segarra, A. G. Marques and A. Ribeiro. 'Connecting the Dots: Identifying Network Structure via Graph Signal Processing'. In: *IEEE Signal Process. Mag.* 36.3 (May 2019), pp. 16–43.

[77] V. Martínez, F. Berzal and J.-C. Cubero. 'A survey of link prediction in complex networks'. In: *ACM computing surveys (CSUR)* 49.4 (2016), pp. 1–33.

[78] L. A. Adamic and N. Glance. 'The political blogosphere and the 2004 US election: divided they blog'. In: *Proceedings of the 3rd international workshop on Link discovery*. 2005, pp. 36–43.

[79] V. N. Zadorozhnyi and E. B. Yudin. 'Growing network: Models following nonlinear preferential attachment rule'. en. In: *Physica A: Statistical Mechanics and its Applications* 428 (June 2015), pp. 111–132. (Visited on 10/11/2020).

[80] A. Venkitaraman, S. Chatterjee and B. Wahlberg. 'Recursive Prediction of Graph Signals With Incoming Nodes'. In: *IEEE Intl. Conf. Acoustics, Speech and Signal Process. (ICASSP)*. May 2020, pp. 5565–5569.

[81] V. Kalofolias. 'How to learn a graph from smooth signals'. In: *Art. Intel. Stat.* PMLR. 2016, pp. 920–929.

[82] X. Dong, D. Thanou, M. Rabbat and P. Frossard. 'Learning graphs from data: A signal representation perspective'. In: *IEEE Signal Process. Mag.* 36.3 (2019), pp. 44–63.

[83] X. Dong, D. Thanou, P. Frossard and P. Vandergheynst. 'Learning Laplacian matrix in smooth graph signal representations'. In: *IEEE Trans. Signal Process.* 64.23 (2016), pp. 6160–6173.

[84] S. Segarra, A. G. Marques, G. Mateos and A. Ribeiro. 'Network topology inference from spectral templates'. In: *IEEE Trans. Signal Inf. Process. Netw.* 3.3 (2017), pp. 467–483.

[85] R. Shafipour, S. Segarra, A. Marques and G. Mateos. 'Identifying the topology of undirected networks from diffused non-stationary graph signals'. In: *IEEE Open Journal of Signal Processing* (2020).

[86] D. Thanou, X. Dong, D. Kressner and P. Frossard. 'Learning heat diffusion graphs'. In: *IEEE Trans. Signal Inf. Process. Netw.* 3.3 (2017), pp. 484–499.

[87] M. Coutino, E. Isufi, T. Maehara and G. Leus. 'State-space network topology identification from partial observations'. In: *IEEE Trans. Signal Inf. Process. Netw.* 6 (2020), pp. 211–225.

[88] H. E. Egilmez, E. Pavez and A. Ortega. 'Graph learning from data under Laplacian and structural constraints'. In: *IEEE Journal of Selected Topics in Signal Processing* 11.6 (2017), pp. 825–841.

D

[89] P. Ravikumar, M. J. Wainwright, G. Raskutti, B. Yu *et al.* 'High-dimensional covariance estimation by minimizing 1-penalized log-determinant divergence'. In: *Electronic Journal of Statistics* 5 (2011), pp. 935–980.

[90] N. Meinshausen, P. Bühlmann *et al.* 'High-dimensional graphs and variable selection with the lasso'. In: *The annals of statistics* 34.3 (2006), pp. 1436–1462.

[91] V. Kalofolias, A. Loukas, D. Thanou and P. Frossard. 'Learning time varying graphs'. In: *IEEE Intl. Conf. Acoustics, Speech and Signal Process. (ICASSP)*. Ieee. 2017, pp. 2826–2830.

[92] D. Hallac, Y. Park, S. Boyd and J. Leskovec. 'Network inference via the time-varying graphical lasso'. In: *Intl. Conf. Knowledge Discovery Data Mining*. 2017, pp. 205–213.

[93] S. Vlaski, H. P. Mareti, R. Nassif, P. Frossard and A. H. Sayed. 'Online graph learning from sequential data'. In: *IEEE Data Science Workshop (DSW)*. 2018, pp. 190–194.

[94] M. Moscu, R. Borsoi and C. Richard. 'Online Graph Topology Inference with Kernels For Brain Connectivity Estimation'. In: *IEEE Intl. Conf. Acoustics, Speech and Signal Process. (ICASSP)*. 2020, pp. 1200–1204.

[95] R. Shafipour and G. Mateos. 'Online proximal gradient for learning graphs from streaming signals'. In: *European Signal Process. Conf. (EUSIPCO)*. 2021, pp. 865–869.

[96] B. Zaman, L. M. L. Ramos, D. Romero and B. Beferull-Lozano. 'Online topology identification from vector autoregressive time series'. In: *IEEE Trans. Signal Process.* 69 (2020), pp. 210–225.

[97] A. Natali, M. Coutino, E. Isufi and G. Leus. 'Online time-varying topology identification via prediction-correction algorithms'. In: *IEEE Intl. Conf. Acoustics, Speech and Signal Process. (ICASSP)*. 2021, pp. 5400–5404.

[98] M. McPherson, L. Smith-Lovin and J. M. Cook. 'Birds of a feather: Homophily in social networks'. In: *Annual review of sociology* 27.1 (2001), pp. 415–444.

[99] L. Lü and T. Zhou. 'Link prediction in complex networks: A survey'. In: *Physica A: statistical mechanics and its applications* 390.6 (2011), pp. 1150–1170.

[100] A. Clauset, C. Moore and M. E. Newman. 'Hierarchical structure and the prediction of missing links in networks'. In: *Nature* 453.7191 (2008), pp. 98–101.

[101] R. Guimerà and M. Sales-Pardo. 'Missing and spurious interactions and the reconstruction of complex networks'. In: *Proceedings of the National Academy of Sciences* 106.52 (2009), pp. 22073–22078.

[102] F. Tan, Y. Xia and B. Zhu. 'Link prediction in complex networks: a mutual information perspective'. In: 9.9 (2014).

[103] T. Zhou, L. Lü and Y.-C. Zhang. 'Predicting missing links via local information'. In: *The European Physical Journal B* 71.4 (2009), pp. 623–630.

[104] W. Liu and L. Lü. 'Link prediction based on local random walk'. In: *Europh. Let.* 89.5 (2010), p. 58007.

[105] E. A. Leicht, P. Holme and M. E. Newman. 'Vertex similarity in networks'. In: *Physical Review E* 73.2 (2006), p. 026120.

[106] V. Matta, A. Santos and A. H. Sayed. 'Graph learning with partial observations: Role of degree concentration'. In: *IEEE International Symposium on Information Theory (ISIT)*. 2019, pp. 1312–1316.

[107] M. Cirillo, V. Matta and A. H. Sayed. 'Learning Bollobás-Riordan Graphs Under Partial Observability'. In: *IEEE Intl. Conf. Acoustics, Speech and Signal Process. (ICASSP)*. 2021.

[108] A. Sandryhaila and J. M. F. Moura. 'Discrete signal processing on graphs'. In: *IEEE Trans. Signal Process.* 61.7 (2013), pp. 1644–1656.

[109] Y. Chen and M. J. Wainwright. 'Fast low-rank estimation by projected gradient descent: General statistical and algorithmic guarantees'. In: *arXiv preprint arXiv:1509.03025* (2015).

[110] H. Gupta, K. H. Jin, H. Q. Nguyen, M. T. McCann and M. Unser. 'CNN-based projected gradient descent for consistent CT image reconstruction'. In: *IEEE Trans. Medical Imag.* 37.6 (2018), pp. 1440–1453.

[111] E. Ceci and S. Barbarossa. 'Graph Signal Processing in the Presence of Topology Uncertainties'. In: *IEEE Trans. Signal Process.* 68 (2020), pp. 1558–1573.

[112] E. Ceci and S. Barbarossa. 'Small perturbation analysis of network topologies'. In: *IEEE Intl. Conf. Acoustics, Speech and Signal Process. (ICASSP)*. 2018.

[113] F. Hua, C. Richard, J. Chen, H. Wang, P. Borgnat and P. Gonçalves. 'Learning combination of graph filters for graph signal modeling'. In: *IEEE Signal Processing Letters* 26.12 (2019), pp. 1912–1916.

[114] A. Sandryhaila and J. M. F. Moura. 'Discrete Signal Processing on Graphs'. In: *IEEE Trans. Signal Process.* 61.7 (Apr. 2013). Conference Name: IEEE Transactions on Signal Processing, pp. 1644–1656.

[115] D. Thanou, D. Shuman and P. Frossard. 'Learning parametric dictionaries for signals on graphs'. In: *IEEE Trans. Signal Process.* 62.15 (2014), pp. 3849–3862.

[116] F. Gama, E. Isufi, G. Leus and A. Ribeiro. 'Graphs, convolutions, and neural networks: From graph filters to graph neural networks'. In: *IEEE Signal Process. Mag.* 37.6 (2020), pp. 128–138.

[117] J. Cervino, L. Ruiz and A. Ribeiro. 'Increase and Conquer: Training Graph Neural Networks on Growing Graphs'. In: *arXiv preprint arXiv:2106.03693* (2021).

[118] L. Lovász. *Large networks and graph limits*. Vol. 60. American Mathematical Soc., 2012.

[119] B. Dai, S. Ding and G. Wahba. 'Multivariate bernoulli distribution'. In: *Bernoulli* 19.4 (2013), pp. 1465–1483.

[120] V. N. Vapnik. 'An overview of statistical learning theory'. In: *IEEE Trans. Neural Netw.* 10.5 (1999), pp. 988–999.

[121] A. Rakotomamonjy, F. Bach, S. Canu and Y. Grandvalet. 'SimpleMKL'. In: *Journal of Machine Learning Research* 9 (2008), pp. 2491–2521.

D

[122] G. Turin. 'An introduction to matched filters'. In: *IEEE Trans. Info. Theory* 6.3 (1960), pp. 311–329.

[123] S. P. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[124] X. Zhu, J. Lafferty and R. Rosenfeld. 'Semi-supervised learning with graphs'. In: (2005).

[125] A. J. Smola and R. Kondor. 'Kernels and regularization on graphs'. In: *Learning theory and kernel machines*. Springer, 2003, pp. 144–158.

[126] D. Zhou and B. Schölkopf. 'A regularization framework for learning from graph data'. In: *ICML 2004 Workshop on Statistical Relational Learning and Its Connections to Other Fields (SRL 2004)*. 2004, pp. 132–137.

[127] A. Arguez, I. Durre, S. Applequist, R. S. Vose, M. F. Squires, X. Yin, R. R. Heim and T. W. Owen. 'NOAA's 1981–2010 US climate normals: an overview'. In: *Bul. Amer. Meteor. Soc.* 93.11 (2012), pp. 1687–1697.

[128] J. Mei and J. M. F. Moura. 'Signal processing on graphs: Causal modeling of unstructured data'. In: *IEEE Trans. Signal Process.* 65.8 (2016), pp. 2077–2092.

[129] E. Isufi, A. Loukas, N. Perraudin and G. Leus. 'Forecasting time series with varma recursions on graphs'. In: *IEEE Trans. Signal Process.* 67.18 (2019), pp. 4870–4885.

[130] N. Perraudin, J. Paratte, D. Shuman, L. Martin, V. Kalofolias, P. Vandergheynst and D. K. Hammond. 'GSPBOX: A toolbox for signal processing on graphs'. In: (2014).

[131] Z. Wang, Y. Tan and M. Zhang. 'Graph-based recommendation on social networks'. In: *2010 12th International Asia-Pacific Web Conference*. IEEE. 2010, pp. 116–122.

[132] N. Silva, D. Carvalho, A. C. Pereira, F. Mourão and L. Rocha. 'The pure cold-start problem: A deep study about how to conquer first-time users in recommendations domains'. In: *Information Systems* 80 (2019), pp. 1–12.

[133] F. Orabona. 'A modern introduction to online learning'. In: *arXiv preprint arXiv:1912.13213* (2019).

[134] E. Hazan *et al.* 'Introduction to online convex optimization'. In: *Found. Trend. Opt.* 2.3-4 (2016), pp. 157–325.

[135] A. Rahimi and B. Recht. 'Random features for large-scale kernel machines'. In: *Adv Neu. Inf. Proc. Sys.* 20 (2007).

[136] Z. Zong and Y. Shen. 'Online Multi-Hop Information Based Kernel Learning Over Graphs'. In: *IEEE Intl. Conf. Acoustics, Speech and Signal Process. (ICASSP)*. IEEE. 2021, pp. 2980–2984.

[137] R. Money, J. Krishnan and B. Beferull-Lozano. 'Online non-linear topology identification from graph-connected time series'. In: *2021 IEEE Data Science and Learning Workshop (DSLW)*. IEEE. 2021, pp. 1–6.

[138] R. Money, J. Krishnan and B. Beferull-Lozano. 'Sparse online learning with kernels using random features for estimating nonlinear dynamic graphs'. In: *IEEE Trans. Signal Process.* (2023).

[139] R. Shafipour and G. Mateos. 'Online topology inference from streaming stationary graph signals with partial connectivity information'. In: *Algorithms* 13.9 (2020), p. 228.

[140] L. Jian, J. Li and H. Liu. 'Toward online node classification on streaming networks'. In: *Data Mining and Knowledge Discovery* 32.1 (2018), pp. 231–257.

[141] R. Nassif, C. Richard, J. Chen and A. H. Sayed. 'A graph diffusion LMS strategy for adaptive graph signal processing'. In: *Conf. Signals, Syst., Computers (Asilomar)*. IEEE. 2017, pp. 1973–1976.

[142] F. Hua, R. Nassif, C. Richard, H. Wang and A. H. Sayed. 'Online distributed learning over graphs with multitask graph-filter models'. In: *IEEE Trans. Signal Inf. Process. Netw.* 6 (2020), pp. 63–77.

[143] R. Nassif, C. Richard, J. Chen and A. H. Sayed. 'Distributed diffusion adaptation over graph signals'. In: *IEEE Intl. Conf. Acoustics, Speech and Signal Process. (ICASSP)*. IEEE. 2018, pp. 4129–4133.

[144] V. R. Elias, V. C. Gogineni, W. A. Martins and S. Werner. 'Adaptive graph filters in reproducing kernel Hilbert spaces: Design and performance analysis'. In: *IEEE Trans. Signal Inf. Process. Netw.* 7 (2020), pp. 62–74.

[145] B. Das and E. Isufi. 'Graph Filtering over Expanding Graphs'. In: *IEEE Data Science Learning Workshop Processing (DSLW)*. May 2022.

[146] X. Liu, P. C. Hsieh, N. Duffield, R. Chen, M. Xie and X. Wen. 'Streaming network embedding through local actions'. In: *arXiv preprint arXiv:1811.05932* (2018).

[147] B. Das and E. Isufi. 'Learning expanding graphs for signal interpolation'. In: *IEEE Intl. Conf. Acoustics, Speech and Signal Process. (ICASSP)*. IEEE. 2022, pp. 5917–5921.

[148] B. Das, A. Hanjalic and E. Isufi. 'Task-aware connectivity learning for incoming nodes over growing graphs'. In: *IEEE Trans. Signal Inf. Process. Netw.* 8 (2022), pp. 894–906.

[149] S. Vlaski, S. Kar, A. H. Sayed and J. M. Moura. 'Networked Signal and Information Processing: Learning by multiagent systems'. In: *IEEE Signal Process. Mag.* 40.5 (2023), pp. 92–105.

[150] S. Shalev-Shwartz *et al.* 'Online learning and online convex optimization'. In: *Foundations and Trendső in Machine Learning* 4.2 (2012), pp. 107–194.

[151] A. Simonetto and E. DallAnese. 'Prediction-correction algorithms for time-varying constrained optimization'. In: *IEEE Trans. Signal Process.* 65.20 (2017), pp. 5481–5494.

[152] M. E. Newman. 'A measure of betweenness centrality based on random walks'. In: *Social networks* 27.1 (2005), pp. 39–54.

[153] B. Ruhnau. 'Eigenvector-centralitya node-centrality?' In: *Social networks* 22.4 (2000), pp. 357–365.

[154] F. M. Harper and J. A. Konstan. 'The movielens datasets: History and context'. In: *Acm Trans. Inter. Int. Sys.* 5.4 (2015), pp. 1–19.

[155] E. Dong, H. Du and L. Gardner. 'An interactive web-based dashboard to track COVID-19 in real time'. In: *The Lancet infectious diseases* 20.5 (2020), pp. 533–534.

[156] J. H. Giraldo, A. Mahmood, B. Garcia-Garcia, D. Thanou and T. Bouwmans. 'Reconstruction of Time-Varying Graph Signals via Sobolev Smoothness'. In: *IEEE Trans. Signal Inf. Process. Netw.* 8 (2022), pp. 201–214.

[157] M. V. Shcherbakov, A. Brebels, N. L. Shcherbakova, A. P. Tyukov, T. A. Janovsky, V. A. Kamaev *et al.* 'A survey of forecast error measures'. In: *World applied sciences journal* 24.24 (2013), pp. 171–176.

[158] A. Casteigts, P. Flocchini, W. Quattrociocchi and N. Santoro. 'Time-varying graphs and dynamic networks'. In: *Intl. Journ. Par., Emerg. Distr. Sys.* 27.5 (2012), pp. 387–408.

[159] A. Natali, E. Isufi, M. Coutino and G. Leus. 'Learning time-varying graphs from online data'. In: *IEEE Open J. Sig. Process.* 3 (2022), pp. 212–228.

[160] Y. Shen, B. Baingana and G. B. Giannakis. 'Tensor decompositions for identifying directed graph topologies and tracking dynamic networks'. In: *IEEE Trans. Signal Process.* 65.14 (2017), pp. 3675–3687.

[161] B. Baingana and G. B. Giannakis. 'Tracking switched dynamic network topologies from information cascades'. In: *IEEE Trans. Signal Process.* 65.4 (2016), pp. 985–997.

[162] A. Buciulea, M. Navarro, S. Rey, S. Segarra and A. G. Marques. 'Online Network Inference from Graph-Stationary Signals with Hidden Nodes'. In: *arXiv preprint arXiv:2409.08760* (2024).

[163] A. Javaheri, A. Amini, F. Marvasti and D. P. Palomar. 'Joint Signal Recovery and Graph Learning from Incomplete Time-Series'. In: *arXiv preprint arXiv:2312.16940* (2023).

[164] A. Hyvärinen and E. Oja. 'Independent component analysis: algorithms and applications'. In: *Elsevier Neu. Networks* 13.4-5 (2000), pp. 411–430.

[165] Q. Lu, V. N. Ioannidis and G. B. Giannakis. 'Graph-adaptive semi-supervised tracking of dynamic processes over switching network modes'. In: *IEEE Trans. Signal Process.* 68 (2020), pp. 2586–2597.

[166] Y. Liu, S. Magliacane, M. Kofinas and E. Gavves. 'Graph switching dynamical systems'. In: *Intl. Conf. Machine Learn. (ICML)*. PMLR. 2023, pp. 21867–21883.

[167] E. N. Davison, K. J. Schlesinger, D. S. Bassett, M. E. Lynall, M. B. Miller, S. T. Grafton and J. M. Carlson. 'Brain network adaptability across task states'. In: *PLoS comp. bio.* 11.1 (2015), e1004029.

[168] A. G. Mahyari and S. Aviyente. 'Identification of dynamic functional brain network states through tensor decomposition'. In: *IEEE Intl. Conf. Acoustics, Speech and Signal Process. (ICASSP)*. IEEE. 2014, pp. 2099–2103.

[169] Y. Qiu, G. Zhou, Y. Wang, Y. Zhang and S. Xie. 'A generalized graph regularized non-negative tucker decomposition framework for tensor data representation'. In: *IEEE Tran. Cyb.* 52.1 (2020), pp. 594–607.

[170] A. Buciulea, S. Rey and A. G. Marques. 'Learning graphs from smooth and graph-stationary signals with hidden variables'. In: *IEEE Trans. Signal Inf. Process. Netw.* 8 (2022), pp. 273–287.

[171] M. Navarro, S. Rey, A. Buciulea, A. G. Marques and S. Segarra. 'Joint network topology inference in the presence of hidden nodes'. In: *IEEE Transactions on Signal Processing* (2024).

[172] D. J. Houghton and A. N. Joinson. 'Privacy, social network sites, and social relations'. In: *Hum. Serv. Netw. Soc.* Routledge, 2014, pp. 77–97.

[173] V. C. Gungor, B. Lu and G. P. Hancke. 'Opportunities and challenges of wireless sensor networks in smart grid'. In: *IEEE transactions on industrial electronics* 57.10 (2010), pp. 3557–3564.

[174] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein *et al.* 'Distributed optimization and statistical learning via the alternating direction method of multipliers'. In: *Found. Tren. Mach. Learn.* 3.1 (2011), pp. 1–122.

[175] A. Buciulea, J. Ying, A. G. Marques and D. P. Palomar. 'Polynomial Graphical Lasso: Learning Edges from Gaussian Graph-Stationary Signals'. In: *arXiv preprint arXiv:2404.02621* (2024).

[176] Y. Xu and W. Yin. 'A block coordinate descent method for regularized multiconvex optimization with applications to nonnegative tensor factorization and completion'. In: *SIAM Jour. Imag. Sci.* 6.3 (2013), pp. 1758–1789.

[177] S. P. Chepuri, S. Liu, G. Leus and A. O. Hero. 'Learning sparse graphs under smoothness prior'. In: *IEEE Intl. Conf. Acoustics, Speech and Signal Process. (ICASSP)*. IEEE. 2017, pp. 6508–6512.

[178] R. Shafipour, A. Hashemi, G. Mateos and H. Vikalo. 'Online topology inference from streaming stationary graph signals'. In: *2019 IEEE Data Science Workshop (DSW)*. IEEE. 2019, pp. 140–144.

[179] S. Segarra, Y. Wang, C. Uhler and A. G. Marques. 'Joint inference of networks from stationary graph signals'. In: *Conf. Signals, Syst., Computers (Asilomar)*. IEEE. 2017, pp. 975–979.

[180] J. Friedman, T. Hastie and R. Tibshirani. 'Sparse inverse covariance estimation with the graphical lasso'. In: *Biostatistics* 9.3 (2008), pp. 432–441.

[181] A. Mokhtari, S. Shahrampour, A. Jadbabaie and A. Ribeiro. 'Online optimization in dynamic environments: Improved regret rates for strongly convex problems'. In: *2016 IEEE 55th Conference on Decision and Control (CDC)*. IEEE. 2016, pp. 7195–7201.

[182] S. Rey, B. Das and E. Isufi. 'Online Learning Of Expanding Graphs'. In: *arXiv preprint arXiv:2409.08660* (2024).

[183] S. Theodoridis. *Machine learning: a Bayesian and optimization perspective*. Academic press, 2015.

[184] C. Williams and C. Rasmussen. 'Gaussian processes for regression'. In: *Adv Neu. Inf. Proc. Sys.* 8 (1995).

D

[185] F. Gama, J. Bruna and A. Ribeiro. 'Stability properties of graph neural networks'. In: *IEEE Trans. Signal Process.* 68 (2020), pp. 5680–5695.

[186] Y.-X. Wang and Y.-J. Zhang. 'Nonnegative matrix factorization: A comprehensive review'. In: *IEEE Tran. Know. Dat. Eng.* 25.6 (2012), pp. 1336–1353.

[187] K. Huang, N. D. Sidiropoulos and A. Swami. 'Non-negative matrix factorization revisited: Uniqueness and algorithm for symmetric decomposition'. In: *IEEE Trans. Signal Process.* 62.1 (2013), pp. 211–224.

[188] H. Laurberg, M. G. Christensen, M. D. Plumbley, L. K. Hansen and S. H. Jensen. 'Theorems on positive data: On the uniqueness of NMF'. In: *Comp. Intel. Neuros.* 2008.1 (2008), p. 764206.

[189] N. Gillis. 'Sparse and unique nonnegative matrix factorization through data preprocessing'. In: *J. Mach. Learn. Res.* 13.1 (2012), pp. 3349–3386.

[190] C. Bhattacharya, N. Goyal, R. Kannan and J. Pani. 'Non-negative matrix factorization under heavy noise'. In: *Intl. Conf. Machine Learn. (ICML)*. PMLR. 2016, pp. 1426–1434.

[191] S. Boyd and L. Vandenberghe. *Introduction to applied linear algebra: vectors, matrices, and least squares.* Cambridge university press, 2018.

ACKNOWLEDGEMENTS

First and foremost, I would like to thank Elvin. You were always very prompt with your feedback. I have learnt a lot about writing from you. The hundreds of comments and suggestions have certainly influenced how I write and review, in general. I see the value in it more over time. I have learned things from you which I hope will help me further. I appreciate your push for collaborating with others, which is something I should have led by myself more actively. Also thank you for your patience as I feel I have been very lazy and stubborn at times with wrapping up existing work. Thank you for keeping the standards high, which is something I hope to uphold in the times ahead.

I would like to thank Alan for the feedback he provided especially on the higher-level picture. It is important to have such a vision in mind, and it is another aspect I have learned to appreciate over the years.

I would also like to really thank Geert, Michel, Antonio, Gonzalo, and Cédric (in no particular order) for agreeing to be a part of the committee.

For someone who spent a sizeable part of this journey without the presence of colleagues, getting to know and bond with those in MMC towards the end has had a profound impact. Thank you (again in no particular order) Mohammad, Andrea, Maosheng, Tianqi, Dimme, Jorge, Wenyi, Marijn, Vimal, Chengen. It was a blessing to have you guys as colleagues. I value it immensely. It was also a pleasure to have met and worked with Andrei, Madeline, and Samuel.

Outside of the university environment, I am thankful to have maintained and/or forged some of my more long-standing friendships. I would like to mention Udhav, Rajesh, Yash, Tamaghna, Akash, and Aranyak. Four or more continuous years in any persons life can throw up a few things here and there, and it is comforting to know that there are souls you can rely on. Also thankful to my parents who they brought me up in a very open environment, never enforced their world views on me, and always let me choose what I wanted to do. This has allowed me to remain independent in some way.

Also thankful to some of the math teachers I have had through school. I dont think without them I would have eventually ended up in this field.

I would also like to acknowledge the many Youtube channels and content creators from whom I have learned a lot and were often a constant presence in the background. I am nothing without you guys.

CURRICULUM VITÆ

Bishwadeep DAS

26-09-1992	Born in Kolkata, India.

EDUCATION

1998–2009	South Point High School
2009–2011	Birla High School
2012–2016	Bachelors in Electronics & Communication Engineering Motilal Nehru National Institute of Technology Prayagraj
2017–2019	Masters in Electrical Engineering Technische Universiteit Delft
2020–2024	PhD. Technische Universiteit Delft *Thesis:* Signal processing over dynamic graphs *Promotor:* Prof. dr. A. Hanjalic *Copromotor:* Dr. E. Isufi