



Universidad
Rey Juan Carlos

TESIS DOCTORAL

Robust Network Topology Inference and Processing of Graph Signals

Autor:

SAMUEL REY ESCUDERO

Director:

ANTONIO GARCÍA MARQUÉS

**Programa de Doctorado Interuniversitario
en Multimedia y Comunicaciones**

Escuela Internacional de Doctorado

2022

Contents

Abstract	v
1 Introduction	1
1.1 Motivation and context	1
1.2 Objectives	5
1.3 Summary of contributions	6
1.4 Outline of the dissertation	9
2 Fundamentals of graph signal processing	11
2.1 Graphs, graph signals, and the GSO	11
2.2 Graph filters and filter identification	13
2.3 Models for graph signals	13
2.4 Graph inverse problems: denoising and interpolation	15
2.5 Graph learning	16
2.6 Graph neural networks	18
2.7 Graph perturbations in GSP	19
3 Non-linear denoising of graph signals	21
3.1 Introduction	21
3.2 GNNs for graph-signal denoising	22
3.3 Graph convolutional generator	24
3.3.1 Guaranteed denoising with the GCG	25
3.3.2 Numerical inspection of the deep GCG spectrum	28
3.4 Graph upsampling decoder	29
3.4.1 Graph upsampling operator from hierarchical clustering	31
3.4.2 Guaranteed denoising with the GDec	33
3.4.3 Analyzing the deep GDec	34
3.5 Numerical results	35
3.5.1 Denoising capability of graph untrained architectures	35
3.5.2 Denoising synthetic data	36
3.5.3 Denoising real-world signals	38
3.6 Conclusion	39
3.7 Appendix: Proof of Theorem 3.1	40
3.8 Appendix: Proof of Lemma 3.1	40
3.9 Appendix: Proof of Lemma 3.2	42
4 Signal interpolation of diffused sparse signals	43
4.1 Introduction	43
4.1.1 Successively aggregated graph signals	44

4.2	Aggregation Sampling of DSGS	44
4.2.1	Aggregating the sparse input	45
4.2.2	Aggregating the diffused sparse input	47
4.2.3	Blind deconvolution	47
4.2.4	Space-shift sampling of diffused sparse signals	48
4.3	Numerical experiments	49
4.4	Conclusion	51
5	Robust graph filter identification	53
5.1	Introduction	53
5.2	GF identification with imperfect graph knowledge	54
5.2.1	Modeling graph perturbations	56
5.3	Robust GF identification	57
5.3.1	Alternating minimization for robust GF identification	58
5.3.2	Leveraging stationary observations	61
5.4	Joint robust identification of multiple GFs	61
5.4.1	Joint GF identification for time series	63
5.5	Efficient implementation of the robust GF identification algorithm	65
5.6	Numerical results	68
5.6.1	Synthetic experiments	68
5.6.2	Real-world datasets	71
5.7	Concluding remarks	73
5.8	Appendix: Proof of Theorem 5.1	74
6	Robust network topology inference	77
6.1	Introduction	78
6.2	Topology inference model in the presence of hidden variables	79
6.2.1	Correlation and partial correlation networks with hidden variables	79
6.3	Topology inference from stationary signals	81
6.3.1	Topology inference with stationary observations as a convex optimization	82
6.3.2	Robust network inference	82
6.4	Joint inference from stationary signals in the presence of hidden variables	83
6.4.1	Modeling hidden variables in the joint inference problem	84
6.4.2	Convex relaxations for the joint topology inference	85
6.5	Numerical experiments	86
6.5.1	Numerical experiments based on joint inference	86
6.6	Conclusion	88
7	Concluding remarks	89
7.1	Revisiting the proposed goals	89
7.2	Future lines of research	90
	Bibliography	93

Abstract

The abundance of large and heterogeneous systems is rendering contemporary data more pervasive, intricate, and with a non-regular structure. With classical techniques facing troubles to deal with the irregular (non-Euclidean) domain where the signals are defined, a popular approach at the heart of graph signal processing (GSP) is to: (i) represent the underlying support via a graph and (ii) exploit the topology of this graph to process the signals at hand. In addition to the irregular structure of the signals, another critical limitation is that the observed data is prone to the presence of perturbations, which, in the context of GSP, may affect not only the observed signals but also the topology of the supporting graph. Ignoring the presence of perturbations, along with the couplings between the errors in the signal and the errors in their support, can drastically hinder estimation performance. While many GSP works have looked at the presence of perturbations in the signals, much fewer have looked at the presence of perturbations in the graph, and almost none at their joint effect. While this is not surprising (GSP is a relatively new field), we expect this to change in the upcoming years. Motivated by the previous discussion, the goal of this thesis is to advance toward a robust GSP paradigm where the algorithms are carefully designed to incorporate the influence of perturbations in the graph signals, the graph support, and both. To do so, we consider different types of perturbations, evaluate their disruptive impact on fundamental GSP tasks, and design robust algorithms to address them.

The first part of the thesis addresses the presence of perturbations in the graph signals, which typically lead to more tractable problems. When the observed signals are corrupted by additive noise, we introduce two untrained *nonlinear* graph neural network architectures to remove the noise from the observations, develop theoretical guarantees for their denoising capabilities in a simple setup, and provide empirical evidence in more general scenarios. Each of the architectures incorporates the information encoded by the graph in a different manner: one relying on graph convolutions, and the other employing graph upsampling operators based on hierarchical clustering. Intuitively, each architecture implements a different prior over the targeted signals. Then, we move on to a setting where perturbations appear in the form of missing values. In this case, we assume that the original signal is a *diffused sparse graph signal*, interpret the missing values as samples gathered through a *successive aggregation* sample scheme, and study the recovery (interpolation) of the original signal. Depending on the particular application, the goal is to use the *local* observations to recover the diffused signal or (the location and values of) the seeds. Different sampling configurations are investigated, including those of known and unknown locations of the sources as well as those of the diffusing filter being unknown.

The second part of the thesis deals with perturbations in the topology of the graph, which give rise to more challenging formulations. In this sense, we propose a novel approach for handling perturbations in the links of the graph and apply it to the problem of robust graph filter (GF) identification from input-output observations. Different from existing works, we formulate a non-convex optimization problem that operates in the vertex domain and jointly performs GF identification and graph denoising, and hence, on top of learning the desired GF, an estimate of the graph is obtained as a byproduct. To handle the resulting bi-convex problem, we design

an algorithm that blends techniques from alternating optimization and majorization minimization, showing its convergence to a stationary point. Then, moving on to the last type of perturbation, we investigate the problem of learning a graph from nodal observations for setups where only a subset of the nodes is observed, with the others remaining unobserved or hidden. Our schemes assume the number of observed nodes is considerably larger than the number of hidden nodes, and build on recent GSP models to relate the signals and the underlying graph. Specifically, we go beyond classical correlation and partial correlation approaches and assume that the signals are *stationary* in the sought graph, and moreover, we propose a joint network topology inference framework where several related graphs are estimated together. The underlying idea is to exploit the similarity of the different graphs to enhance the quality of the estimation. Since the resulting problems are ill-conditioned and non-convex, the block matrix structure of the proposed formulations is leveraged, and suitable convex-regularized relaxations are presented.

Although the methodology and focus of this thesis are more theoretical (defining an estimation problem, stating the considered assumptions, obtaining the estimates as solutions to rigorously formulated optimization problems, designing computationally efficient provably convergent algorithms and, whenever possible, characterizing the performance of those), the experimental results will also play an important role. To that end, we evaluate the performance of our algorithms over synthetic and real-world datasets and compare their results with state-of-the-art alternatives. These experiments reflect the impact of ignoring the presence of perturbations, show the strengths and weaknesses of the proposed methods, demonstrate that in a number of settings our methods outperform current alternatives, and assess the applicability of our schemes to real-world problems.

Chapter 1

Introduction

We begin by providing a short overview of the research environment surrounding this thesis, motivating the relevance of graph-based methods and highlighting the impact of the presence of perturbations in the data. After that, the chapter: (i) states the main objectives sought by this work; (ii) lists the resulting contributions; and (iii) presents a brief outline of the remaining chapters.

1.1 Motivation and context

In the last two decades, we have been experiencing a data deluge largely propelled by the pervasive deployment of networks of sensing devices, the massive use of online social media, and the unstoppable digitalization of our daily tasks. At the same time, as contemporary interconnected systems grow in size and importance, the data generated by such systems becomes more complex and heterogeneous, motivating the fast development of new methods and techniques to process datasets defined over irregular (non-Euclidean) domains [1–4]. Among the novel approaches that emerged to handle contemporary data, one particularly tractable and fruitful consists in modeling the underlying irregular structure by means of a *graph*, and then, interpreting the data as signals defined on the graph, which are commonly referred to as *graph signals*. This graph-based perspective has rapidly grown in popularity and it has been successfully applied to data obtained from power, communication, social, geographical, financial, or biological networks, to name a few [5–7]. Moreover, it has attracted the attention of researchers from different areas, including statistics, machine learning, and signal processing.

Precisely, interpreting signals with irregular support as graph signals and then exploiting the topology of the underlying graph to process the signals is at the core of graph signal processing (GSP), a relatively new field that is developing swiftly [8–11]. GSP is devoted to developing new models and algorithms for processing graph signals, oftentimes by generalizing classical tools originally conceived to process signals with regular support (time or space). Based on the fundamental assumption that there exists a close relation between the properties of the signals and the topology of the graph where they are supported, the key to the success of GSP is to effectively exploit the relation between the graph and the signals. To that end, a considerable proportion of the efforts in GSP are directed at analyzing how the algebraic and spectral characteristics of the graph impact

the properties of the graph signals. In this analysis, the so-called graph-shift operator (GSO) plays a fundamental role. The GSO is a sparse matrix whose sparsity pattern encodes the topology of the graph, rendering it a cornerstone element within the GSP framework [8, 12]. For example, employing the GSO as a building block enables the definition of different spectral tools such as the graph Fourier Transform (GFT) [13–15], or more general graph-signal operators such as graph filters (GFs), which may be expressed as polynomials of the GSO [12, 16–18].

A wide range of graph-related problems have been addressed under the GSP umbrella, and even though a variety of goals and assumptions are considered for the different problems, the key idea of harnessing the relation between the graph and the signals remains a constant. A popular problem consists in modeling an arbitrary linear transformation between some input and output graph signals through a GF. This task is commonly referred to as GF design or GF identification, and the inferred GF may be interpreted as the dynamics driving a network-diffusion process of interest [16, 17, 19, 20]. The sampling and reconstruction of graph signals is also an interesting problem [21–24], with meaningful connections to semisupervised learning. Note that while sampling signals defined over regular domains is relatively straightforward (regular sampling schemes are prudent and give rise to sample signals that are regular as well), this is not the case for graph signals, which are inherently irregular. Hence, the efforts when approaching this task focus on designing sampling schemes that exploit the graph structure allowing to effectively recover the whole signal from its sampled version. The reconstruction of the signal is also known as graph signal interpolation, and it is related to solving an inverse problem that involves both the signal observations and the supporting graph [25–28]. Depending on the actual relation between the observations and the original signal, the problem has been addressed from a point of view of signal denoising, inpainting, or signal super-resolution, to name a few. Another fundamental but considerably different problem is that of network topology inference, also known as graph learning [29–34]. In contrast with previous GSP problems, in network topology inference the focus is placed on the topology of the graph, which is unknown, and therefore, the goal is to infer the graph from a set of nodal observations. Finally, in the context of deep learning, another line of research that has attracted attention is the development of non-linear architectures that exploit the relation between graphs and signals by incorporating the topology of the graph into their design. This popular family of neural networks (NNs) is known as graph neural networks (GNNs), and it encompasses a gamut of different graph-based architectures that have been applied to a wide range of problems [35–39].

All the aforementioned GSP applications use as input the observed data (graph signals) and the observed/inferred support (the graph). Unfortunately, imperfect knowledge due to the presence of noise, missing values, or outliers is pervasive in contemporary data science applications. In this sense, we will use the generic term *perturbation* to refer to any imperfection in the observed data, encompassing a variety of defects whose particularities will depend on the application at hand and the features of the data. To further illustrate the diverse nature of perturbations in a GSP context, consider the example of a network of sensors measuring some quantity of interest. The process of acquiring the measurement will introduce a certain amount of noise, and furthermore, if any sensor is damaged then its measures will be completely lost. Equally important, the information about the connections between the sensors may not be fully accurate. This example clearly illustrates that, when dealing with GSP, one needs to account for (i) perturbations in the graph signals; (ii) perturbations in the topology of the graph; and (iii) the joint effects and interactions between these two.

We start describing the presence of *perturbations in the graph signals*. Clearly, signal perturbations have been extensively investigated in signal processing, statistics, and data science, so that many of the classical results can also be leveraged in the GSP setup. From this point of view, the two key (distinctive) questions when dealing with perturbed graph signals are: (i) how does the

graph influence the perturbation? and (ii) how can the graph be exploited to design the schemes that mitigate/eliminate the perturbations? Due to the combination of practical relevance, tractability, and the existence of related works from classical signal processing, accounting for perturbations in the observed graph signals has attracted a considerable amount of attention. Accounting for this, we focus on studying two specific types of perturbations that have been thoroughly analyzed (the presence of noise in the signals and the presence of missing values), considering graph-aware processing and acquisition architectures that had not been investigated before.

- **(P1) Noise in the graph signals.** This simple case assumes that the observed signals are corrupted by additive noise, typically modeled as an independent identically distributed (i.i.d.) random variable drawn from a particular distribution. Noisy graph signals arise in a gamut of graph-related applications such as measurements in electric, social and transportation networks, or monitoring biological signals [6, 10, 25, 40]. While denoising schemes have been thoroughly investigated in classical signal processing, the noise perturbing graph signals is oftentimes related to the topology of the graph (e.g., the noise can be independent across nodes while its variance is proportional to the node degree or any other node centrality measure). Even more importantly, GSP denoising schemes must be particularized to exploit the graph when mitigating (eliminating) the noise. This process is known as *graph-based signal denoising*, and traditional approaches include minimizing the graph total variation to push the signal values at neighboring nodes to be close [25, 41], promoting a notion of signal smoothness by adding a regularization parameter based on the quadratic form of the graph Laplacian [42], or encouraging the recovery of signals with a smooth gradient [27]. More recently, non-linear solutions for denoising graph signals have been proposed, with relevant examples based on median graph filters [43], graph autoencoders [44], or graph unrolling architectures [45].
- **(P2) Missing values.** We use this term to refer to setups where only a subset of the entries of the graph signal are available. This accounts for cases where the values are missing / totally corrupted, as well as for sampling setups where the remaining entries were purposely unobserved. Practical graph scenarios that can lead to missing values include damaged sensors in a sensor network, wrong or incomplete answers when the data is gathered through online forms, or just because sampling the signal values at every node is not feasible in large networks [6, 46, 47]. A number of alternatives arise to deal with this type of perturbed signals, with naive alternatives including filling the missing values with zeros or using the mean value within the observed values in the one-hop neighborhood. To fill the missing values, a reasonable and rigorous approach is to look at the problem from the sampling perspective and design methods to perform *graph signal interpolation*. Two critical aspects in this regard are the postulation of a parsimonious model for the graph signal (bandlimited, diffused, smooth...) and the impact of the scheme collecting the samples. Several works have investigated different instances of this problem, with a strong bias towards assuming that the original signal is graph bandlimited and that the observed values proceed from observations taken at a fixed subset of nodes [21, 22, 24, 48]. Alternatively, other works have postulated that the observed values correspond to successively aggregating the values of the signal from neighboring nodes [23, 49], and designed the associated optimal interpolation schemes.

We shift focus now to the second class of imperfections studied in this work: *perturbations in the graph topology*. In this case, recall that GSP builds upon exploiting the relation between the signals and the graph, and hence, it is not surprising that methods within the GSP framework are particularly sensitive to this type of imperfections. More precisely, a fair amount of GSP methods rely on either the spectrum of the GSO or in polynomials of the GSO, and because the GSO captures

the topology of the graph, the perturbations in the observed topology translate into perturbations in the GSO. However, even when assuming additive models to represent the perturbations, which are the easiest type of models to deal with, measuring the impact of the perturbations in the topology on the spectrum of the GSO and the polynomials of the GSO is a challenging endeavor. Not only that, but when perturbations affect the number of observed nodes, even the size of the GSO will be uncertain. In conclusion, the perturbations in the topology of the graph will not only hinder the performance of GSP algorithms but, furthermore, developing robust alternatives that model the presence of perturbations is a non-trivial and ill-posed problem, which has been barely studied in the GSP literature. In this work, we focus on two types of graph perturbations: uncertainty (imperfections) in the edges of the graph, and unobserved nodes (whose effects will be analyzed in the context of graph learning). These two types of perturbations, which are both theoretically and practically relevant, are described in more detail next.

- **(P3) Uncertainty in the edges.** Here, we assume that the set of nodes is perfectly known and that imperfect information about the existence and strength of the links (graph topology) is available. The perturbations in the observed topology may encompass observing edges that do not exist in the true graph, missing/unobserved edges that exist in the true graph, noise present in the weights of the observed edges, or any combination of the previous options. These perturbations appear in a gamut of practical situations. On the one hand, when networks are given explicitly, perturbations may be due to observational noise and errors (e.g., link failures in power or wireless networks [50]). On the other hand, when in lieu of physical entities, the graphs model (statistical) pairwise relationships among the observed variables, they need to be inferred from the data [31, 32, 51]. While this type of perturbation is critical for many GSP methods, modeling the influence of the imperfect topological information and developing robust alternatives is a challenging task, and hence, there is a limited number of works approaching this problem. In the frequency domain, [52] employs a small perturbation analysis to study the impact of perturbations in the spectrum of the graph Laplacian. In the vertex domain, [53, 54] postulates a graphon-based perturbation model applied to GFs of order one. Then, in more recent approaches, [55] combines structural equation model (SEM) with total least squares (TLS) to jointly infer the GF and the perturbations in the GSO, and [56] proposes a robust GF identification alternative where the support of the graph is assumed to be known, so the perturbations are constrained to be noise in the observed edges.
- **(P4) Hidden nodes.** Finally, we consider a perturbation where some elements of the nodal set are not known, which is a problem particularly acute in the context of network topology inference. Clearly, when hidden nodes are present, one has only access to signals (measurements) from the remaining (observed) nodes. However, this should not be confused with the (missing values) perturbations introduced in **(P2)**. There, the signals at some nodes were not observed, but the existence of the node and its connections to other nodes were known. In contrast, here not only the underlying graph is unknown, but even the number of hidden nodes is oftentimes not known. A number of estimation goals arise in the context of **(P4)**: inferring the number of hidden nodes, the connections among them, or the values of the signals, to name a few. In the specific context of network topology inference under the presence of hidden nodes, the problem is more challenging because the links among observed nodes are unknown as well. As a result, the main goal is usually to estimate the links between the observed nodes (also known as learning the *observed subgraph*). More ambitious approaches aim also at estimating the links between observed and unobserved nodes. Clearly, all these problems are related and challenging (highly correlated values from two observed nodes may be explained not only by an edge between the two nodes but by a

third hidden node connected to them). Some network-inference methods have started to look at the problem of hidden nodes with examples in the context of Gaussian graphical model selection [57, 58], inference of linear Bayesian networks [59], non-linear regression [60], or brain connectivity [61], to name a few. Nonetheless, there are still many network-inference methods (including most in the context of GSP) that have not considered this type of perturbations.

To summarize, the presence of perturbations in GSP setups leads to having imperfect knowledge about the graph signals, the graph, or both. Clearly, if these perturbations are ignored, the performance of naive algorithms that use as input the corrupted data will be drastically hindered. The solution is to design robust schemes that model and incorporate into their formulation the presence and effects of the perturbations. While relevant and timely, this is a challenging problem, especially when the perturbations affect the graph topology. Bearing all this in mind, our goal, which is presented in detail in the next section, is to develop a set of GSP methods robust to different types of perturbations, understand their differences and similarities, and discuss how they can be combined in a meaningful and tractable way.

1.2 Objectives

As discussed in the previous section, the presence of perturbations in the observed data constitutes a relevant and ubiquitous problem. Motivated by this, the prevailing objective of this thesis is to advance towards a *robust GSP* paradigm where the algorithms are carefully designed to deal with the presence of perturbations in the graphs and the signals. To that end, we aim to analyze and understand the impact of the different types of perturbations in several fundamental GSP problems and then, design a robust formulation capable of: (i) recovering the original data from the perturbed observations; and/or (ii) approaching the desired task while taking into account the presence of perturbations in the data to minimize their disruptive influence. To render these generic (and relatively ambitious) goals more reachable, we focus our research efforts on four specific objectives. Each of them considers a specific GSP problem and addresses one of the perturbations introduced in Section 1.1.

(O1) Non-linear denoising of graph signals. Consider a setting where the observed graph signals are corrupted with noise as described in the perturbation type **(P1)**. Our goal is to design non-linear architectures to denoise the observed graph signals. Because dealing with noisy signals is a problem that has been studied to a considerable extent in the GSP literature, here we focus on the design of untrained *non-linear* architectures and on their theoretical characterization. First, we will explore different ways of incorporating the information encoded in the graph and propose novel graph-aware NN architectures to denoise graph signals. Second, we will provide theoretical guarantees for the denoising performance of the proposed architectures, and we will show that the denoising capability is directly influenced by the topology of the underlying graph.

(O2) Signal interpolation of diffused sparse graph signals. Consider a setting where the observed signals have missing values as described in the perturbation type **(P2)**. Our goal is to reconstruct the observed signal when the observations are taken at a particular node according to a successive local *aggregation sampling scheme* (AGSS). The signal to be reconstructed is assumed to be a *diffused sparse graph signal* (DSGS), a class of signals that can be modeled as a signal with zeros everywhere except in a few seeding nodes, which is then diffused through the network via a

GF. Ultimately, we will develop a reconstruction algorithm to recover both the original (diffused) signal and the seeds from the perturbed observation.

(O3) Robust GF identification. Consider a setting where the observed graph is perturbed as described in the perturbation type **(P3)**. Our goal is to estimate a GF from some input-output signal pairs while accounting for the presence of errors in the supporting graph and in the observed signals. The proposed approach needs to bypass the challenges associated with robust *spectral* graph theory and avoid the numerical instability from computing high-order polynomials. We will also address the scenario where several GFs need to be estimated, with all the GFs being defined as polynomials of the same GSO.

(O4) Robust network topology inference. Consider a network topology inference problem where a subset of the nodes remain hidden as described in the perturbation type **(P4)**. Our goal is to estimate the joint topology of the observed subgraph while taking into account the presence of the hidden nodes. The observed signals will be assumed to be stationary in the unknown graph and our main focus will be considering the case where several related graphs, all defined over the same set of nodes, need to be learned. Here, we will develop a *joint network topology inference* algorithm that exploits the graph similarity while accounting for the presence of hidden nodes. Note that leveraging the graph similarity between hidden nodes (which are not observed) is an interesting but non-trivial problem.

Clearly, objectives **(O1)** and **(O2)** are concerned with perturbations involving the graph signals and they aim to design schemes to clean/infer the graph signal of interest. In contrast, objectives **(O3)** and **(O4)** are concerned with perturbations involving the topology of the graph and they aim to solve higher order GSP tasks while mitigating (bypassing) the presence of the perturbations. On top of these four specific objectives, two transversal objectives are also considered. First, since many of the findings and challenges faced when addressing **(O1)-(O4)** are also present in related robust GSP setups, our aim is that the models and tools put forward in this thesis improve our understanding of the influence of perturbations in a general way, hoping that these robust approaches may be extended/generalized to other important GSP applications where the presence of perturbations is critical. Secondly, even though the scope of this thesis is markedly theoretical, showcasing the potential applicability of the algorithms developed herein is certainly important. Therefore, we define another transversal objective, which consists in assessing the practical value of our algorithms using real-world datasets.

1.3 Summary of contributions

This section provides the list of publications organized around the (results and contributions) of the four objectives presented in the previous section.

The graph signal denoising task **(O1)** is addressed in [62, 63]. The preliminary work in [62] proposed an underparametrized deep decoder NN capable of learning non-linear representation for graph signals, which were used for compression and denoising. Later on, [63] presented two *un-trained* and overparametrized GNNs to address the graph signal denoising problem. To incorporate the topology of the graph, the first architecture employs a fixed (non-learnable) GF to generalize the convolutional layer in [35]. The second architecture performs graph upsampling operations that, starting from a low-dimensional latent space, progressively increase the size of the input until it matches the size of the signal to denoise. Furthermore, a mathematical analysis was con-

ducted for each architecture offering bounds for their performance, improving our understanding of nonlinear architectures and the influence of incorporating the graph structure into NNs. Interestingly, the decoder architecture introduced in [63] has proven useful for other problems than signal denoising. The decoder was employed to design a graph deep decoder capable of learning the mapping between input-output signal pairs defined on different graphs [64, 65]. The key idea is that the encoder uses the input graph to map the input signal onto a latent space, and then, the decoder uses the output graph to reconstruct the output signal from the latent representation. The publications related to **(O1)** are listed below.

- [62] S. Rey, A. G. Marques, and S. Segarra, "An underparametrized deep decoder architecture for graph signals," in IEEE Intl. Wrksp. Computat. Advances Multi-Sensor Adaptive Process. (CAMSAP). IEEE, 2019, pp. 231–235.
- [63] S. Rey, S. Segarra, R. Heckel, and A. G. Marques, "Untrained graph neural networks for denoising," arXiv preprint arXiv:2109.11700, 2021 (submitted to IEEE Trans. Signal Process.).
- [64] S. Rey, V. M. Tenorio, S. Rozada, L. Martino, and A. G. Marques, "Deep encoder-decoder neural network architectures for graph output signals," in Conf. Signals, Syst., Computers (Asilomar). IEEE, 2019, pp. 225–229.
- [65] S. Rey, V. M. Tenorio, S. Rozada, L. Martino, and A. G. Marques, "Overparametrized deep encoder-decoder schemes for inputs and outputs defined over graphs," in European Signal Process. Conf. (EUSIPCO). IEEE, 2021, pp. 855–859.

The goal of graph signal interpolation **(O2)** is pursued in [66], where the observed (non-missing) values of the perturbed signals are assumed to be taken at a particular node according to an AGSS. In the AGSS, which was originally proposed for bandlimited graph signals (BGS), the nodes successively aggregate the values of the signal in their neighborhood, and moreover, the recovery of the original signals can be guaranteed even if observations are gathered at a single node. Firstly, [66] applied AGSS to the case where the observed signals are DSGS in lieu of BGS. Secondly, after defining the observational model for the perturbed signals, the paper proposed an interpolation algorithm defined in the spectral domain. Finally, existing results for support identification and blind deconvolution were generalized to deal with AGSS and DSGS. The publication related to **(O2)** is listed below.

- [66] S. Rey, F. J. I. Garcia, C. Cabrera, and A. G. Marques, "Sampling and reconstruction of diffused sparse graph signals from successive local aggregations", IEEE Signal Process. Lett., vol. 26, no. 8, pp. 1142–1146, 2019.

The robust robust GF identification problem **(O3)** is approached in [67, 68]. In those works, the proposed solution was formulated in the vertex domain, avoiding the numerical instability of computing large polynomials and, at the same time, bypassing the challenges associated with robust spectral graph theory. The robust GF identification was recast as a joint optimization problem where the GF identification objective was augmented with a graph-denoising regularizer so that, on top of the desired GF, the proposed algorithm also provided an enhanced estimate of the supporting graph. The joint formulation led to a non-convex bi-convex optimization algorithm, for which a provably-convergent efficient algorithm able to find an approximate solution was developed. Furthermore, to address scenarios where multiple GFs are present, the paper generalized the robust framework so that multiple GFs, all defined over the same graph, were jointly identified. Also related to **(O3)**, [69] introduced the neighborhood GF, a new type of GF that is numerically

stable and robust to perturbations in the observed topology. The definition of neighborhood GF, which replaced the powers of the GSO with k -hop adjacency matrices, was exploited to provide an alternative design of graph convolutional NN (GCNN) that was employed in graph signal denoising and node classification problems. The publications involved with **(O3)** are listed below.

- [67] S. Rey and A. G. Marques, "Robust graph-filter identification with graph denoising regularization", in *IEEE Int. Conf. Acoustics, Speech Signal Process. (ICASSP)*. IEEE, 2021, pp. 5300–5304.
- [68] S. Rey, V. M. Tenorio, and A. G. Marques, "Robust graph filter identification and graph denoising from signal observations," *arXiv preprint arXiv:2210.08488*, 2022 (submitted to *IEEE Trans. Signal Process.*).
- [69] V. M. Tenorio, S. Rey, F. Gama, S. Segarra, and A. G. Marques, "A robust alternative for graph convolutional neural networks via graph neighborhood filters," in *Conf. Signals, Syst., Computers (Asilomar)*. IEEE, 2021, pp. 1573–1578.

Lastly, the network topology inference in the presence of hidden nodes **(O4)** is addressed in [70, 71]. Initially, [70] investigated how the presence of hidden variables impacts the classical definition of graph stationarity. Key to the proposed formulation was the consideration of a block matrix factorization approach and harnessing the low rankness and the sparsity pattern present in the blocks related to hidden variables. Then, we exploited this block matrix factorization in [71] to propose a topology inference method that, assuming that the observed signals are graph-stationary, jointly learns multiple graphs while accounting for the presence of hidden variables. To fully benefit from the joint inference formulation and successfully exploit the graph similarity among hidden nodes, the paper carefully exploited the structure inherent to the presence of latent variables with a regularization inspired by group Lasso [72]. An additional work that is closely related to the objective **(O4)** is presented in [73]. The paper presented a graph-learning algorithm that assumes that a reference graph with a density of motifs similar to that of the sought graph was known. Then, this similarity was harnessed to reveal a connection between the spectra of both graphs, which was exploited in the formulation of the inference problem and the associated algorithm. The prior information about the density of motifs of the unknown graph is local and robust, in the sense that it enables the comparison of graphs of different sizes, an issue that was non-trivial. Moreover, leveraging this prior to boost the performance of graph learning algorithms in the presence of hidden nodes arises as an interesting research problem, which is left as a future research direction. The publications related to **(O4)** are listed below.

- [70] A. Buciulea, S. Rey, C. Cabrera, and A. G. Marques, "Network reconstruction from graph-stationary signals with hidden variables", in *Conf. Signals, Syst., Computers (Asilomar)*. IEEE, 2019, pp. 56–60.
- [71] S. Rey, A. Buciulea, M. Navarro, S. Segarra, and A. G. Marques, "Joint inference of multiple graphs with hidden variables from stationary graph signals", in *IEEE Int. Conf. Acoustics, Speech Signal Process. (ICASSP)*. IEEE, 2022, pp. 5817–5821.
- [73] S. Rey, T. M. Roddenberry, S. Segarra, and A. G. Marques, "Enhanced graph-learning schemes driven by similar distributions of motifs", *arXiv preprint arXiv:2207.04747*, 2022 (submitted to *IEEE Trans. Signal Process.*).

1.4 Outline of the dissertation

The remainder of this document is organized as follows. First, Chapter 2 introduces fundamental definitions and concepts that will be employed during the thesis. Then, Chapter 3 considers the presence of noise in the signals and proposes non-linear architectures to perform graph signal denoising. Chapter 4 addresses the presence of missing values in the observed signals and introduces an interpolation method for DSGS. Regarding the perturbations in the topology, Chapter 5 addresses the problem of GF identification assuming imperfect knowledge of the observed topology, and Chapter 6 approaches the task of network topology inference while accounting for the presence of hidden nodes. Finally, Chapter 7 provides some concluding remarks and identifies some interesting future research directions.

Chapter 2

Fundamentals of graph signal processing

This chapter introduces the main concepts and tools from GSP, which are the foundations to the research carried out in this thesis. To that end, we begin by defining the basics of GSP, then introduce some fundamental tools and methods, and close the section describing a couple of more advanced GSP concepts directly related to this thesis.

2.1 Graphs, graph signals, and the GSO

A *graph* is a mathematical structure formally defined as $\mathcal{G} := (\mathcal{V}, \mathcal{E})$, where \mathcal{V} and \mathcal{E} are, respectively, the sets containing the *nodes* and *edges* conforming the graph, which are also commonly known as *vertices* and *links*. The nodes collected in \mathcal{V} are typically labeled using integers so $\mathcal{V} := \{1, 2, \dots, N\}$, with N denoting the number of nodes in the graph. Then, the edges collected in \mathcal{E} are represented by pairs of nodes (i, j) with $i, j \in \mathcal{V}$ and $(i, j) \in \mathcal{E}$ if and only if the node i is connected to node j . If none of the edges in the graph are directed, that is, if edges are agnostic to which node is the origin and which is the destiny, then the graph is called *undirected*, and hence, $(i, j) \in \mathcal{E}$ implies that $(j, i) \in \mathcal{E}$. In contrast, when the graph captures the direction of the edges it is called *directed*, and we might encounter that $(i, j) \in \mathcal{E}$ but $(j, i) \notin \mathcal{E}$. Fig. 2.1 represents an undirected graph where the nodes in \mathcal{V} are represented in blue and the edges in \mathcal{E} are represented as gray lines. Intuitively, a graph encodes pairwise relations between the nodes in \mathcal{V} , with these relations being represented by the edges. Then, an *unweighted* graph captures whether an edge exists or not but it does not provide any information about the strength (closeness, similarity,...) of the connection. On the other hand, this additional information about the distance or closeness between connected nodes is provided by *weighted* graphs. The distinction between weighted and unweighted graphs is apparent when looking at the *adjacency matrix*, a widely used matrix representation of the topology of \mathcal{G} . The adjacency matrix \mathbf{A} is a sparse $N \times N$ matrix encoding the connectivity of the graph whose entry $A_{ij} \neq 0$ if and only if $(i, j) \in \mathcal{E}$. When the graph is unweighted the entries of \mathbf{A} are binary, i.e., $\mathbf{A} \in \{0, 1\}^{N \times N}$. On the contrary, if the graph is weighted then $\mathbf{A} \in \mathbb{R}^{N \times N}$ and the non-zero entries A_{ij} capture the weight of the edge between the nodes i and j . Similarly, when the graph is undirected the matrix \mathbf{A} is symmetric. Another concept involving the connectivity of the graph is that of the neighborhood of a node. For any node i , its *neighborhood* $\mathcal{N}_i := \{j \in \mathcal{V} | (i, j) \in \mathcal{E}\}$ is the set of nodes that are connected to i . Furthermore, the *degree* of a node i is given by the number of neighbors, so it is formally defined

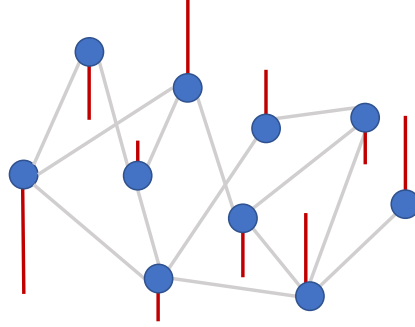


Figure 2.1: Depiction of a graph signal \mathbf{x} and the underlying graph \mathcal{G} . Nodes are represented in blue, the edges are the connections in gray, and the height of the red vertical bars represents the values of \mathbf{x} at each node.

as $d_i := |\mathcal{N}_i| = [\mathbf{A}\mathbf{1}]_i$, where $\mathbf{1}$ denotes the vector of all ones. In other words, d_i can be computed by adding the entries of the i -th row of \mathbf{A} . If graphs are directed, one must account for defining incoming (outcoming) neighborhoods as well as incoming (outcoming) degrees.

We move on to the definition of graph signals, which represent the other fundamental piece within the GSP framework and constitute the subject of study in most GSP problems. Formally, a *graph signal* can be modeled as a function from the node set to the real field¹ $x : \mathcal{V} \rightarrow \mathbb{R}$ or, equivalently, as an N -dimensional real-valued vector $\mathbf{x} = [x_1, \dots, x_N]^\top \in \mathbb{R}^N$, with x_i denoting the value of the signal at the node i . An example of a graph signal is given in Fig. 2.1, where the height of the vertical bars represents the value of the signal at each node. Then, because the graph signal \mathbf{x} is defined on \mathcal{G} , the core assumption of GSP is that either the values or the properties of \mathbf{x} depend on the topology of \mathcal{G} [12]. For instance, consider a graph that encodes similarity. If the value of A_{ij} is high, then one will expect the signal values x_i and x_j to be akin to each other. This rationale helps to explain the advantages of leveraging the topology of the graph when processing graph signals. Some relevant examples of graph signal models that reflect the influence of the graph topology on \mathbf{x} are provided in Section 2.3.

The last key element in the GSP framework is the so-called *graph-shift operator* (GSO), a square matrix that captures the topology of the underlying graph \mathcal{G} [12]. The GSO is denoted by $\mathbf{S} \in \mathbb{R}^{N \times N}$ and its entry S_{ij} is allowed to be non-zero if and only if $i = j$ or $(i, j) \in \mathcal{E}$. Intuitively, \mathbf{S} can be understood as a topology-aware local operator that can be applied to process graph signals. There exist several options for selecting the GSO, with typical choices including the adjacency matrix \mathbf{A} , the graph combinatorial Laplacian $\mathbf{L} := \text{diag}(\mathbf{A}\mathbf{1}) - \mathbf{A}$, and its normalized variants [8, 12]. Note that $\text{diag}(\cdot)$ denotes the diagonal operator that transforms a vector into a diagonal matrix. In this sense, using \mathbf{S} instead of a specific choice for the GSO is particularly useful since it provides a higher level of abstraction and results in algorithms that may be applied to a wider range of scenarios. When \mathcal{G} is assumed to be undirected, it follows that \mathbf{S} is symmetric and it can be diagonalized as $\mathbf{S} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^\top$, where the orthonormal matrix $\mathbf{V} \in \mathbb{R}^{N \times N}$ collects the eigenvectors of \mathbf{S} , and the diagonal matrix $\mathbf{\Lambda} = \text{diag}(\boldsymbol{\lambda})$ collects the eigenvalues $\boldsymbol{\lambda} \in \mathbb{R}^N$. On the other hand, when \mathcal{G} represents a directed graph we will assume that \mathbf{S} is still diagonalizable and its decomposition will be given by $\mathbf{S} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^{-1}$. Note that, in the directed case, the eigenvalues collected in $\mathbf{\Lambda}$ are likely to be complex numbers.

¹For simplicity, we focus our discussion on scalar, real-valued graph signals, but the values associated with each node could be discrete, complex, or even vectors (e.g., when multiple features per node are observed).

2.2 Graph filters and filter identification

Graph filters (GFs) are topology-aware linear operators whose inputs and outputs are graph signals. More specifically, graph filters implement a linear transformation that can be expressed as a polynomial of the GSO of the form

$$\mathbf{H} := \sum_{r=0}^{R-1} h_r \mathbf{S}^r = \mathbf{V} \text{diag}(\mathbf{\Psi} \mathbf{h}) \mathbf{V}^{-1} = \mathbf{V} \text{diag}(\tilde{\mathbf{h}}) \mathbf{V}^{-1}, \quad (2.1)$$

where \mathbf{H} is the graph filter, and $\mathbf{h} := [h_0, \dots, h_{R-1}]^\top$ is the vector collecting the filter coefficients h_i . The $N \times R$ Vandermonde matrix $\mathbf{\Psi}$ defined as $\Psi_{ij} := \Lambda_{ii}^{j-1}$ represents the GFT for GFs, and thus, $\tilde{\mathbf{h}} := \mathbf{\Psi} \mathbf{h}$ is the vector of size N representing the frequency response of \mathbf{H} [13, 17]. Since \mathbf{S}^r encodes the r -hop neighborhood of the graph, graph filters can be used to diffuse input graph signals \mathbf{x} across the graph as $\mathbf{y} = \sum_{r=0}^{R-1} h_r \mathbf{S}^r \mathbf{x} = \mathbf{H} \mathbf{x}$, where \mathbf{y} is the result of diffusing the signal \mathbf{x} across $R-1$ neighborhoods with h_r being the coefficients of the linear combination.

A relevant problem in the context of GFs is that of GF identification. Consider that we observe M input and output pairs $\mathbf{X} := [\mathbf{x}_1, \dots, \mathbf{x}_M]$ and $\mathbf{Y} := [\mathbf{y}_1, \dots, \mathbf{y}_M]$ whose relation is given by

$$\mathbf{Y} = \mathbf{H} \mathbf{X} + \mathbf{W}, \quad (2.2)$$

with \mathbf{W} being a zero-mean random matrix (typically assumed to have i.i.d. entries) that accounts for noisy measurements and model inaccuracies. Leveraging (2.2), the GF identification task amounts to using the input-output pairs to estimate \mathbf{H} under the model in (2.1), which, if the GSO \mathbf{S} is known, boils down to estimating the GF coefficients collected in $\mathbf{h} \in \mathbb{R}^R$. Hence, we can approach the GF identification task in the node domain by solving the convex problem

$$\min_{\mathbf{h}} \left\| \mathbf{Y} - \sum_{r=0}^{R-1} h_r \mathbf{S}^r \mathbf{X} \right\|_F^2. \quad (2.3)$$

Leveraging the frequency definition of GFs in (2.1), we rewrite the least-squares (LS) cost in (2.3) and obtain its (closed-form) solution as

$$\hat{\mathbf{h}} = \underset{\mathbf{h}}{\text{argmin}} \left\| \text{vec}(\mathbf{Y}) - ((\mathbf{V}^{-1} \mathbf{X})^\top \odot \mathbf{V}) \mathbf{\Psi} \mathbf{h} \right\|_2^2 = \mathbf{\Xi}^\dagger \text{vec}(\mathbf{Y}), \quad (2.4)$$

where $\text{vec}(\cdot)$ denotes the vectorization operation, $\mathbf{V}^{-1} \mathbf{X}$ is the frequency representation of the input signals (see Section 2.3), \odot denotes the Khatri–Rao product, $\mathbf{\Psi}$ is the GFT Vandermonde matrix, $\mathbf{\Xi} := ((\mathbf{V}^{-1} \mathbf{X})^\top \odot \mathbf{V}) \mathbf{\Psi}$, and † is the pseudoinverse operator.

From (2.4) we observe that estimating \mathbf{H} is straightforward under the assumptions of: i) $\mathbf{\Xi}$ being full rank (i.e., the inputs are sufficiently rich); and ii) \mathbf{S} being perfectly known. However, as discussed in the first chapter of this thesis, the assumption in ii) does not hold true in many practical settings. New formulations of (2.4) that account for imperfect GSOs are addressed in Chapter 5.

2.3 Models for graph signals

There is a great diversity of models capturing different relations between the signals and the underlying graph. Here, we introduce some popular models for graph signals, which will be leveraged in subsequent chapters.

Bandlimited graph signals. The notion of bandlimited graph signals links the properties of a signal to those of the spectrum of the supporting graph. To be specific, the frequency representation of the signal \mathbf{x} is given by the N -dimensional vector $\tilde{\mathbf{x}} := \mathbf{V}^{-1}\mathbf{x}$, with \mathbf{V}^{-1} acting as the GFT [13]. Then, a graph signal is said to be *low-pass* bandlimited if $\tilde{\mathbf{x}}$ satisfies that $\tilde{x}_k = 0$ for $k > K$, where $K \leq N$ is referred to as the bandwidth of \mathbf{x} . If \mathbf{x} is bandlimited with bandwidth K , it holds that

$$\mathbf{x} = \mathbf{V}_K \tilde{\mathbf{x}}_K, \quad (2.5)$$

with $\tilde{\mathbf{x}}_K = [\tilde{x}_1, \dots, \tilde{x}_K]$ collecting the active frequency components, and \mathbf{V}_K collecting the corresponding K eigenvectors. In other words, \mathbf{x} lives in a subspace of dimension K spanned by the eigenvectors \mathbf{V}_K . Nonetheless, even though bandlimited graph signals are typically associated with low-pass signals, the non-zero elements in $\tilde{\mathbf{x}}$ are not constrained to its low-frequency components. We might encounter high-pass bandlimited signals or signals whose active frequency components are scattered throughout the spectrum. Furthermore, in relevant cases we might ignore the specific frequency components that are active, further challenging the solution of inverse problems dealing with bandlimited graph signals.

Interestingly, comparing the definition of $\tilde{\mathbf{x}}$ with the definition of $\tilde{\mathbf{h}}$ from (2.1), it follows that, in contrast with classical signal processing, the GFT for graph signals is different from the GFT for GFs. Nonetheless, exploiting the frequency representations $\tilde{\mathbf{x}}$ and $\tilde{\mathbf{h}}$, we have that the output $\mathbf{y} = \mathbf{H}\mathbf{x}$ in the frequency domain is given by

$$\tilde{\mathbf{y}} = \text{diag}(\Psi\mathbf{h})\mathbf{V}^{-1}\mathbf{x} = \text{diag}(\tilde{\mathbf{h}})\tilde{\mathbf{x}} = \tilde{\mathbf{h}} \circ \tilde{\mathbf{x}}, \quad (2.6)$$

with \circ denoting the Hadamard (entry-wise) product. Note that equation (2.6) is the counterpart of the convolution theorem for time signals [17].

Diffused sparse graph signals (DSGS). A graph signal is called a DSGS when it can be modeled as a signal with only a few non-zero entries which is then diffused through the graph. Mathematically, given a GSO \mathbf{S} , a DSGS \mathbf{x} with S non-zero seeds can be written as

$$\mathbf{x} = \mathbf{H}\mathbf{s}, \quad \text{where } \mathbf{H} = \sum_{r=0}^{R-1} h_r \mathbf{S}^r \quad \text{and} \quad \|\mathbf{s}\|_0 \leq S, \quad (2.7)$$

where \mathbf{s} denotes the original sparse signal whose non-zero entries are referred to as *seeding nodes*. Clearly, signals in (2.7) can be viewed as the state reached after the diffusion process modeled by \mathbf{H} is over, and the sparse input $\mathbf{s} \in \mathbb{R}^N$ has been spread throughout the graph.

It is worth noticing that bandlimited signals and DSGS are two generative models with a similar goal: providing a simpler representation of \mathbf{x} . In this sense, the bandlimited model offers an alternative representation of \mathbf{x} that is sparse in the *frequency domain* while the DSGS model offers an alternative representation of \mathbf{x} that is sparse in the *node domain*. As happened with the frequency components, the support of the seeding nodes may be known a priori or we may need to learn it through deconvolution schemes.

Smooth graph signals. A graph signal is considered smooth on \mathcal{G} if the signal value at two connected nodes is “close” or, equivalently, if the difference between the signal value at neighboring nodes is small. A common approach to quantify the smoothness of a graph signal relies on the quadratic form [29]

$$\sum_{(i,j) \in \mathcal{E}} A_{ij} (x_i - x_j)^2 = \mathbf{x}^\top \mathbf{L}\mathbf{x}, \quad (2.8)$$

which quantifies how much the signal \mathbf{x} changes with respect to the notion of similarity encoded in the weights of \mathbf{A} . This measure will be referred to as local variation (LV) of \mathbf{x} . Note that, if the

goal is to obtain the mean LV of M graph signals collected in the $N \times M$ matrix $\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_M]$, this can be achieved by computing

$$\frac{1}{M} \sum_{m=1}^M \mathbf{x}_m^\top \mathbf{L} \mathbf{x}_m = \frac{1}{M} \sum_{m=1}^M \text{tr}(\mathbf{x}_m \mathbf{x}_m^\top \mathbf{L}) = \text{tr}(\hat{\mathbf{C}}_{\mathbf{x}} \mathbf{L}), \quad (2.9)$$

where $\hat{\mathbf{C}}_{\mathbf{x}} := \frac{1}{M} \sum_{m=1}^M \mathbf{x}_m \mathbf{x}_m^\top = \frac{1}{M} \mathbf{X} \mathbf{X}^\top$ denotes the sample estimate of the covariance of \mathbf{X} .

When compared with the previous models for graph signals, it is clear that smoothness is a more lenient assumption. For example, consider that we impose a maximum LV on \mathbf{x} with the constraint $\mathbf{x}^\top \mathbf{L} \mathbf{x} \leq c$ for some $c > 0$. Then, while bandlimited signals are constrained to live in the subspace spanned by \mathbf{V}_K , smooth signals are only constrained to lie within an ellipsoid. Last but not least, more advanced notions of smoothness can be defined by considering $\|\mathbf{x} - \mathbf{H}\mathbf{x}\|_2^2$, where \mathbf{H} represents a pre-specified low-pass GF whose filter taps/frequency response can be tailored to fit the notion of smoothness at hand.

Stationary graph signals. The definition of graph stationarity connects the statistical properties of *random* graph signals with the underlying graph. Formally, a zero-mean random graph signal \mathbf{x} is said to be stationary on \mathcal{G} if its covariance matrix $\mathbf{C}_{\mathbf{x}} = \mathbb{E}[\mathbf{x}\mathbf{x}^\top]$ is a positive-semidefinite polynomial of the GSO \mathbf{S} [74]. When \mathcal{G} is undirected so that $\mathbf{V}\mathbf{V}^\top = \mathbf{I}$, a common example of stationary graph signals arises when \mathbf{x} is the output of a linear graph-diffusion process whose input is a zero mean white signal $\mathbf{w} \in \mathbb{R}^N$, i.e., when the covariance of \mathbf{w} is $\mathbb{E}[\mathbf{w}\mathbf{w}^\top] = \mathbf{I}$ and $\mathbf{x} = \mathbf{H}\mathbf{w}$. In this particular case, we have that the covariance of \mathbf{x} is given by

$$\mathbf{C}_{\mathbf{x}} = \mathbb{E}[\mathbf{x}\mathbf{x}^\top] = \mathbf{H}\mathbb{E}[\mathbf{w}\mathbf{w}^\top]\mathbf{H}^\top = \mathbf{H}\mathbf{H}^\top = \mathbf{H}^2. \quad (2.10)$$

Since the graph filter \mathbf{H} is by definition a polynomial of the GSO \mathbf{S} , from the last equality in (2.10), it readily follows that $\mathbf{C}_{\mathbf{x}}$ is a polynomial of \mathbf{S} as well. As a result, in the spectral domain, it holds that \mathbf{S} and $\mathbf{C}_{\mathbf{x}}$ share the same eigenvectors, and moreover, we have that the matrices \mathbf{S} and $\mathbf{C}_{\mathbf{x}}$ commute, i.e., $\mathbf{C}_{\mathbf{x}}\mathbf{S} = \mathbf{S}\mathbf{C}_{\mathbf{x}}$. Finally, we emphasize that graph stationarity does not impose a deterministic condition on \mathbf{x} but, instead, it imposes a condition on the covariance of the signal.

2.4 Graph inverse problems: denoising and interpolation

The models for graph signals discussed previously in Section 2.3 have been shown to bear practical relevance in real-world datasets and are widely employed in inverse problems. Here, we briefly describe traditional approaches to leverage the properties of some of those models when the observed signals are perturbed.

In the context of *graph signal denoising*, when the perturbation consists in the presence of an additive noise in the signal of interest, we are given the noisy observation $\mathbf{x} = \mathbf{x}_0 + \mathbf{n}$ and the goal is to recover the original signal $\mathbf{x}_0 \in \mathbb{R}^N$. If \mathbf{x}_0 is known to be bandlimited and the graph is undirected, an estimate of \mathbf{x}_0 is readily given by

$$\hat{\mathbf{x}}_0 = \mathbf{V}_K \mathbf{V}_K^\top \mathbf{x}, \quad (2.11)$$

where, by projecting \mathbf{x} onto the subspace spanned by \mathbf{V}_K^\top , we remove the components of the noise \mathbf{n} orthogonal to \mathbf{V}_K^\top while retaining all the energy of \mathbf{x}_0 . Differently, if \mathbf{x}_0 is known to be smooth

and the noise is white and Gaussian, a popular approach is to solve an optimization problem of the form of

$$\hat{\mathbf{x}}_0 = \operatorname{argmin}_{\tilde{\mathbf{x}}_0} \|\mathbf{x} - \tilde{\mathbf{x}}_0\|_2^2 + \alpha \tilde{\mathbf{x}}_0^\top \mathbf{L} \tilde{\mathbf{x}}_0. \quad (2.12)$$

Here, the estimate $\hat{\mathbf{x}}_0$ is a smooth representation of the noisy observation \mathbf{x} , with the weight α controlling the trade-off between minimizing the similarity of $\hat{\mathbf{x}}_0$ and \mathbf{x} and the LV of $\hat{\mathbf{x}}_0$. Note that, if the noise is drawn from a different distribution, then a similarity metric other than the ℓ_2 norm may be preferred. Along the same lines, if additional statistical information about the noise were available (e.g., the covariance of the noise), this can also be incorporated into the minimization problem.

Now, let us consider the problem of *graph signal interpolation*. This is a relevant problem in setups where the graph signal has been corrupted with missing values or, alternatively, when only a sampled version of the signal is available due to the fact that only a subset of nodes has been observed. To be specific, consider the sampling set $\mathcal{Q} \subseteq \mathcal{V}$ with cardinality $Q \leq N$ that collects the set of nodes that have been observed, and define the (fat) selection matrix $\mathbf{\Pi}_Q \in \{0, 1\}^{Q \times N}$ whose elements satisfy: (i) $\sum_j \Pi_{Q,ij} = 1$ for all i ; and (ii) $\sum_i \Pi_{Q,ij} = 1$ if $j \in \mathcal{Q}$ and $\sum_i \Pi_{Q,ij} = 0$ otherwise. Then, if the original signal \mathbf{x} is bandlimited, and assuming that the observed values correspond to observations at different nodes, we denote the perturbed signal with missing values as $\mathbf{x}_Q := \mathbf{\Pi}_Q \mathbf{x} = \mathbf{\Pi}_Q \mathbf{V}_K \tilde{\mathbf{x}}_K$. Under these conditions, the original signal \mathbf{x} can be readily recovered via

$$\hat{\mathbf{x}} = \mathbf{V}_K \tilde{\mathbf{x}}_K = \mathbf{V}_K (\mathbf{\Pi}_Q \mathbf{V}_K)^\dagger \mathbf{x}_Q, \quad (2.13)$$

provided that the rank of the $Q \times K$ submatrix $\mathbf{\Pi}_Q \mathbf{V}_K$ is K . Nonetheless, in some settings, the missing values may be represented more accurately with alternative sampling schemes. An equally valid, but less intuitive approach to sampling a graph signal, is to fix some node i , and consider the sampling of the signal seen by this node as the GSO is applied recursively. In other words, consider that the signal has been locally diffused according to \mathbf{S} , as encoded in the matrix

$$\mathbf{Z} := [\mathbf{z}^{(0)}, \mathbf{z}^{(1)}, \dots, \mathbf{z}^{(N-1)}] = [\mathbf{x}, \mathbf{S}\mathbf{x}, \dots, \mathbf{S}^{N-1}\mathbf{x}]. \quad (2.14)$$

Then, using the matrix \mathbf{Z} and with \mathbf{e}_i denoting the i -th canonical vector, the *successively aggregated* signal at node i is the i -th row of \mathbf{Z} , that is $\mathbf{z}_i := (\mathbf{e}_i^T \mathbf{Z})^T = \mathbf{Z}^T \mathbf{e}_i$. Sampling is now reduced to the selection of Q out of the N elements of \mathbf{z}_i , that is $\mathbf{z}_{Q,i} := \mathbf{\Pi}_Q \mathbf{z}_i = \mathbf{\Pi}_Q (\mathbf{Z}^T \mathbf{e}_i)$. Leveraging the results in [23], the signal \mathbf{x} can be recovered from $\mathbf{z}_{Q,i}$ as

$$\mathbf{x} = \mathbf{V}_K (\mathbf{\Pi}_Q \mathbf{\Psi}^\top \operatorname{diag}(\mathbf{v}_i))^\dagger \mathbf{z}_{Q,i}, \quad \text{with } \mathbf{v}_i := [V_{i,1}, \dots, V_{i,N}]^\top. \quad (2.15)$$

2.5 Graph learning

Graph learning, also known as network topology inference, has developed swiftly in the last years and, currently, is among the most active research areas within GSP. Given a set of graph signals (nodal observations) collected in the matrix $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_M] \in \mathbb{R}^{N \times M}$, which are typically assumed to be independent realizations of a random network process, the goal is to discover the topology of the graph encoded in the GSO by assuming that the observed signals \mathbf{X} and the unknown graph are intimately connected. Fig. 2.2 illustrates the case where a graph learning algorithm is employed to learn the connection between the different regions of the brain based on the signals measured at each region. Intuitively, the relation between \mathbf{X} and \mathcal{G} will depend on the application at hand, with different relations between the observations and the unknown topology

leading to different graph learning algorithms. Here, we will provide a succinct summary of the most relevant approaches based on [34]. The interested reader is referred there for additional details.

One of the first methods to estimate the topology \mathcal{G} is given by *correlation networks*, where the topology is obtained from the Pearson correlation of the i.i.d. random vectors collected in \mathbf{X} . The Pearson correlation coefficient between variables x_i and x_j is denoted as ρ_{ij} and can be computed from the entries of the covariance matrix $[\mathbf{C}_{\mathbf{x}}]_{ij}$. Therefore, in the context of GSP, the GSO for correlation networks is usually set to the sampled covariance $\hat{\mathbf{C}}_{\mathbf{x}}$, or a thresholded version to ensure a sparse matrix \mathbf{S} .

While correlation networks are a simple alternative, high correlations may be due to latent network effects. For example, the random variables x_i and x_j may be highly correlated not because the nodes i and j are connected but because of a third node k that is influencing both of them. In principle, such a confounding can be resolved by considering the *partial correlation* coefficients

$$\rho_{ij|\mathcal{V}\setminus ij} := \frac{\text{cov}(x_i, x_j|\mathcal{V}\setminus ij)}{\sqrt{\text{var}(x_i|\mathcal{V}\setminus ij)\text{var}(x_j|\mathcal{V}\setminus ij)}}. \quad (2.16)$$

Here, $\mathcal{V}\setminus ij$ denotes the set of random variables except for those indexed by nodes i and j . The edge set in partial correlation networks is then defined analogously to their (unconditional) correlation network counterpart.

Of particular interest is the case when each column of \mathbf{X} is sampled independently from the same Gaussian distribution. Under such an assumption, $\rho_{ij|\mathcal{V}\setminus ij} = 0$ implies that x_i and x_j are conditionally independent given the remaining variables in $\mathcal{V}\setminus ij$. The resulting partial correlation network is known as *Gaussian Markov random field (GMRF)* or *Gaussian graphical model* [75]. Then, the key realization is that the partial correlation coefficients $\rho_{ij|\mathcal{V}\setminus ij}$, which capture the topology of the graph, can be expressed as the normalized entries of $\mathbf{C}_{\mathbf{x}}^{-1}$. In the context of GMRFs, this important matrix is known as the *precision* matrix. From the GSP perspective, the previous discussion implies that $\mathbf{S} = \mathbf{C}_{\mathbf{x}}^{-1}$. In other words, the topology of the graph is encoded in the inverse covariance matrix. Leveraging this observation, the notorious *graphical Lasso* algorithm [51] estimates \mathbf{S} through the regularized maximum likelihood (ML) estimator

$$\hat{\mathbf{S}} = \underset{\mathbf{S} \succeq 0}{\text{argmax}} \log \det(\mathbf{S}) - \text{tr}(\hat{\mathbf{C}}_{\mathbf{x}}\mathbf{S}) - \lambda \|\mathbf{S}\|_1, \quad (2.17)$$

where the $\|\mathbf{S}\|_1$ denotes the ℓ_1 norm of the vectorization of \mathbf{S} .

Finally, we consider a network topology inference approach that builds upon the more lenient assumption of stationary graph signals. First, in correlation networks, it was assumed that $\mathbf{S} = \mathbf{C}_{\mathbf{x}}$, and later on, in the graphical Lasso algorithm and other GMRF approaches the relation between

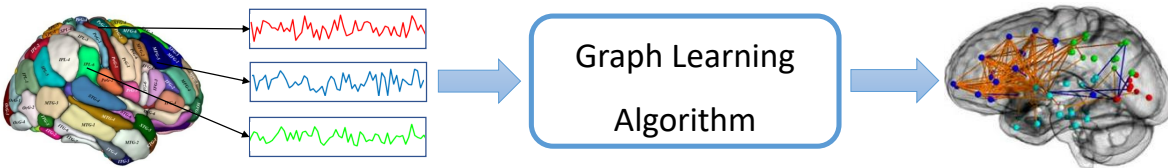


Figure 2.2: Application of a generic graph learning scheme. The input, represented on the left, is given by signals measured in the different regions of the brain (the nodes of the graph). Then, the output of the graph learning algorithm, on the right, are the inferred connections between the different regions, i.e., the estimated topology of the network.

the GSO and the covariance of the observed signals is constrained to $\mathbf{S} = \mathbf{C}_x^{-1}$. In contrast, the assumption of stationary graph signals only implies that the mapping $\mathbf{S} \rightarrow \mathbf{C}_x$ is given by a generic polynomial, hence including the previous scenarios as particular cases. While there are different formulations for this graph learning approach, one particularly interesting for this thesis is given by

$$\hat{\mathbf{S}} = \underset{\mathbf{S}}{\operatorname{argmin}} \|\mathbf{S}\|_1 \quad \text{s. t.} \quad \|\mathbf{S}\hat{\mathbf{C}}_x - \hat{\mathbf{C}}_x\mathbf{S}\|_F^2 \leq \epsilon, \quad \mathbf{S} \in \mathcal{S}, \quad (2.18)$$

which is formulated solely in the node domain thanks to the commutativity constraint $\|\mathbf{S}\hat{\mathbf{C}}_x - \hat{\mathbf{C}}_x\mathbf{S}\|_F^2$. The optimization problem finds the sparsest GSO that commutes with $\hat{\mathbf{C}}_x$, with ϵ being a small positive parameter controlling the quality of the estimate $\hat{\mathbf{C}}_x$, and with \mathcal{S} collecting the requirements for \mathbf{S} to be a specific type of GSO. A typical example is the set of adjacency matrices

$$\mathcal{S}_A := \{A_{ij} \geq 0; \mathbf{A} = \mathbf{A}^\top; A_{ii} = 0; \mathbf{A}\mathbf{1} \geq \mathbf{1}\}, \quad (2.19)$$

where we require the GSO to have non-negative weights, be symmetric, and have no self-loops, and the last constraint rules out the trivial 0 solution by imposing that every node has at least one neighbor. Analogously, the set of combinatorial Laplacian matrices is

$$\mathcal{S}_L := \{L_{ij} \leq 0 \text{ for } i \neq j; \mathbf{L} = \mathbf{L}^\top; \mathbf{L}\mathbf{1} = \mathbf{0}; \mathbf{L} \succeq \mathbf{0}\}, \quad (2.20)$$

where we require the GSO to be a positive semidefinite matrix, have non-positive off-diagonal values, have positive entries on its diagonal, and have the constant vector as an eigenvector (i.e., the sum of the entries of each row to be zero).

So far the section has been focused on a graph learning setting that, in the network science parlance, is known as the *network association* problem [6]. While network association is the most widely considered approach in the context of graph learning, two relevant variations are: the link prediction problem and the network tomography problem. *Link prediction* is a simpler problem in which a subset of the edges of the graph is observed along with the signals. This additional information can be incorporated into the previous framework by modifying the constraint set \mathcal{S} . In contrast, *network tomography* is a more challenging task where the observed signals are perturbed and observations from only a subset of the nodes are available. Precisely, developing robust algorithms that address the latter problem leveraging several GSP assumptions is the subject of Chapter 6.

2.6 Graph neural networks

Generically, we represent a GNN using a parametric nonlinear function $f_{\Theta}(\mathbf{Z}|\mathcal{G}) : \mathbb{R}^{N^{(0)} \times F^{(0)}} \rightarrow \mathbb{R}^N$ that depends on the graph \mathcal{G} . The parameters of the architecture are collected in Θ , and the matrix $\mathbf{Z} \in \mathbb{R}^{N^{(0)} \times F^{(0)}}$ represents the input of the network. Despite the many possibilities for defining a GNN, a broad range of such architectures recursively applies a graph-aware linear transformation followed by an entry-wise nonlinearity. Then, a generic deep graph-based architecture $f_{\Theta}(\mathbf{Z}|\mathcal{G})$ with L layers can be described as

$$\hat{\mathbf{Y}}^{(\ell)} = \mathcal{T}_{\Theta^{(\ell)}}^{(\ell)} \left\{ \mathbf{Y}^{(\ell-1)} | \mathcal{G} \right\}, \quad 1 \leq \ell \leq L, \quad (2.21)$$

$$Y_{ij}^{(\ell)} = g^{(\ell)} \left(\hat{Y}_{ij}^{(\ell)} \right), \quad 1 \leq \ell \leq L, \quad (2.22)$$

where $\mathcal{T}_{\Theta^{(\ell)}}^{(\ell)} \{ \cdot | \mathcal{G} \} : \mathbb{R}^{N^{(\ell-1)} \times F^{(\ell-1)}} \rightarrow \mathbb{R}^{N^{(\ell)} \times F^{(\ell)}}$ is a *graph-aware* linear transformation, $\mathbf{Y}^{(0)} = \mathbf{Z}$ and $\mathbf{y} = \mathbf{Y}^{(L)}$ denote the input and output of the architecture, $\Theta^{(\ell)} \in \mathbb{R}^{F^{(\ell-1)} \times F^{(\ell)}}$ are the

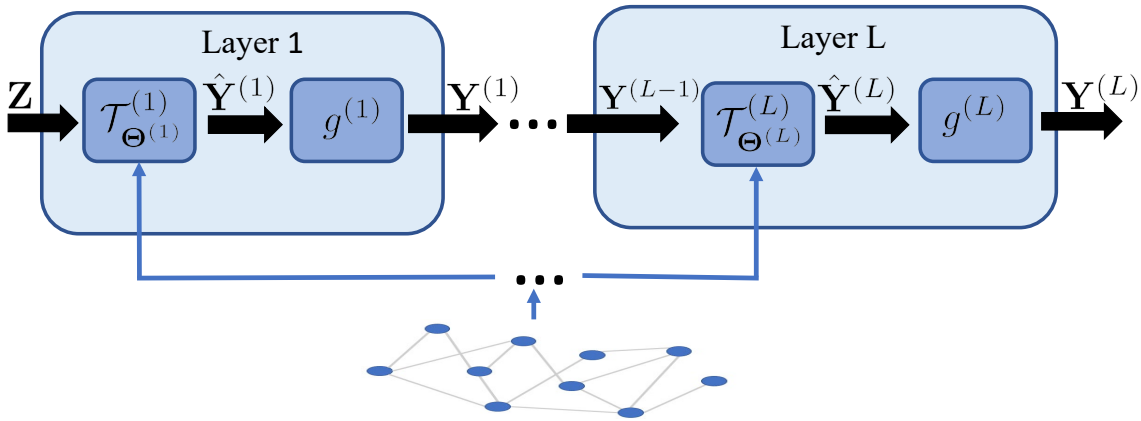


Figure 2.3: Block diagram of a generic GNN with L layers. The inputs of the architecture are the matrix \mathbf{Z} and the topology of the graph. As specified in (2.21)-(2.22), each layer is composed of a learnable graph-aware linear transformation $\mathcal{T}_{\Theta^{(\ell)}}^{\{\mathbf{Y}^{(\ell-1)}|\mathcal{G}\}}$ followed by a entry-wise non-linear transformation $g^{(\ell)}(\hat{\mathbf{Y}}_{ij}^{(\ell)})$.

parameters that define such a transformation, and $g^{(\ell)}: \mathbb{R} \rightarrow \mathbb{R}$ is a *scalar* nonlinear transformation (e.g., the ReLU function), which is oftentimes omitted in the last layer. Moreover, $N^{(\ell)}$ and $F^{(\ell)}$ represent the number of nodes and features at layer ℓ , and $\Theta = \{\Theta^{(\ell)}\}_{\ell=1}^L$ collects all the parameters of the architecture. The structure of the generic GNN specified by the recursion in (2.21)-(2.22) is depicted in Fig. 2.3. Finally, even though the output of $f_{\Theta}(\mathbf{Z}|\mathcal{G})$ are graph signals defined in \mathbb{R}^N , which is the case of interest for Chapter 3, it can be easily adapted to output graph signals with more than one feature.

2.7 Graph perturbations in GSP

The presence of noise in the observed topology represents a relevant but challenging problem that is yet to be studied in more depth by the GSP community. Current works addressing this issue customarily represent the influence that these perturbations exert in the GSO via an additive term because of its tractability, but even then, the resulting models are non-trivial. The source of this difficulty lies in the main tools used in GSP, which are based either on the GFT (eigenvectors of the GSO) or on graph filters (polynomials of the GSO). The challenges are then twofold: (i) characterizing the impact that an additive matrix perturbation has on the eigenvectors and/or a polynomial of that matrix is highly nontrivial; and (ii) even small perturbations on \mathbf{S} may lead to great discrepancies in both the eigenvectors and the associated polynomials, as we show in Chapter 5. Here, we present a succinct overview of relevant works considering the influence of noise in the observed topology to provide some context.

We start with the work presented in [52], which analyzes how perturbations in the edges affect the spectrum of the combinatorial graph Laplacian \mathbf{L} . The authors assume an additive perturbation model and define the perturbed Laplacian as $\bar{\mathbf{L}} := \mathbf{L} + \Delta\mathbf{L}$, with $\Delta\mathbf{L}$ denoting the perturbation matrix. Assuming that all the eigenvalues of \mathbf{L} have multiplicity one and that $\|\Delta\mathbf{L}\|_F \ll \|\mathbf{L}\|_F$, they perform a small perturbation analysis to quantify the influence of the perturbations in the eigenvalues and eigenvectors of \mathbf{L} . Based on this result, and assuming that the perturbation of each edge is modeled as a random event characterized by a certain probability, a statistical analysis is carried out to characterize the mean and the variance of the perturbation of the eigenvalues. Lastly, [52] studies the influence of the perturbations on the spectrum of BGS when the eigenvectors of \mathbf{L} are used as the GFT.

Differently, [54] investigated the influence of perturbations in the adjacency matrix leveraging results from the graphon theory. The perturbed adjacency matrix is also defined based on an additive perturbation model as $\bar{\mathbf{A}} = \mathbf{A} + \Delta_\epsilon \circ (\mathbf{1}_{N \times N} - 2\mathbf{A})$, where $\mathbf{1}_{N \times N}$ is the matrix of all ones of size $N \times N$. The perturbations in Δ_ϵ are modeled as a random graph drawn from either an Erdős-Rényi (ER) or a stochastic block model (SBM), and then, generalizations considering different probabilities for creating and destroying edges and dealing with weighted graphs are also proposed. Finally, the model put forth in [54] is employed to analyze the influence of the perturbations in a polynomial of the GSO of order 2.

To close this chapter, we stress that, for ease of exposition, we presented a taxonomy where the fundamental concepts, tools, and problems in the GSP framework were clearly segregated. Nonetheless, in practical settings we can encounter different combinations of the above problems and generative models giving rise to a rich gamut of GSP tasks. This can be seen in *blind deconvolution*, which is a graph filter identification problem when \mathbf{x} is a DSGS and the seeding nodes in \mathbf{s} are unknown, or in the problem approached in Chapter 5, where we simultaneously estimate a graph filter and denoise the observed topology of the graph.

Chapter 3

Non-linear denoising of graph signals

The first problem considered in the robust GSP framework proposed in this thesis involves the presence of noise in the observed graph signals. As discussed in previous chapters, the presence of noise represents a pervasive type of perturbation capable of rendering the observed data useless when the signal-to-noise ratio is low. As a result, (pre-)processing schemes that remove the noise from the observed signals are required. It is worth recalling that, because the presence of noise in the signals does not affect the topology of the graph and results in tractable problems, there are several works addressing the denoising of graph signals. In this sense, the approach described in this chapter, which encapsulates our work from [62, 63], is primarily concerned with incorporating the information encoded in the graph topology into non-linear architectures and, furthermore, providing a mathematical characterization of the denoising capabilities of the proposed architectures.

Bearing the previous comments in mind, the chapter is organized as follows. Section 3.1 gives a brief overview of the architectures developed and summarizes the main contributions. Section 3.2 formally introduces the problem at hand and presents our general approach. Section 3.3 and Section 3.4 detail the proposed architectures and provide the mathematical analysis for each of them. Numerical experiments are presented in Section 3.5 and concluding remarks are provided in Section 3.6.

3.1 Introduction

In order to develop a *graph-aware non-linear architecture* capable of removing the noise from the observed signals, the goal of this chapter is twofold. First, we explore different ways of incorporating the information encoded in the graph and propose new graph-based NN architectures to denoise graph signals. Second, we provide theoretical guarantees for the denoising capabilities of this approach and show that such guarantees are directly influenced by the properties of the graph. The mathematical analysis, performed on particular instances of these architectures, characterizes their denoising performance under specific assumptions for the original signal and its underlying graph. In addition, we provide empirical evidence about the denoising performance of our method for scenarios more general than those strictly covered by our theory, further illustrating the value of our graph-aware untrained architectures to denoise graph signals.

The presented architectures are *untrained* NNs, meaning that the parameters of the network are optimized using only the signal observation that we want to denoise, avoiding the dependency on a training set with multiple observed graph signals. The underlying assumption behind this *untrained* denoising architecture is that, due to the graph-specific structure incorporated into the different layers, when tuning the network parameters using stochastic gradient steps, the NNs are capable of learning (matching) the structure of the signal faster than that of the noise. Hence, the denoising process is carried out separately for each individual observation by fitting the weights of the NN and stopping the updates after a few iterations. This same phenomenon has been observed to hold true in non-graph deep learning architectures [76, 77]. In the context of signal denoising, the consideration of an overparametrized graph-aware architecture along with early stopping avoids overfitting to the noise.

To incorporate the topology of the graph, the first architecture multiplies the input at each layer by a fixed (non-learnable) graph filter [17], which can be seen as a generalization of the convolutional layer in [35]. The second architecture performs graph upsampling operations that, starting from a low-dimensional latent space, progressively increase the size of the input until it matches the size of the signal to denoise. The sequence of upsampling operators are designed based on hierarchical clustering algorithms [64, 78–80] so that, in contrast to [44], matrix inversions are not required, avoiding the related numerical issues.

Contributions. In summary, the contributions of this chapter are the following:

- (i) We introduce two new overparametrized and untrained GNNs for solving graph-signal denoising problems.
- (ii) We characterize theoretically the denoising performance of each of the two architectures, improving our understanding of nonlinear architectures and the influence of incorporating graph structure into NNs.
- (iii) The proposed architectures are evaluated and compared to other denoising alternatives through numerical experiments carried out with synthetic and real-world data.

3.2 GNNs for graph-signal denoising

We now formally introduce the problem of graph-signal denoising within the GSP framework, and present our approach to tackle it using untrained GNN architectures. Given the graph \mathcal{G} , let us consider the observed graph signal $\mathbf{x} \in \mathbb{R}^N$, which is a noisy version of the original graph signal \mathbf{x}_0 . With $\mathbf{n} \in \mathbb{R}^N$ being a noise vector, the relation between \mathbf{x} and \mathbf{x}_0 is

$$\mathbf{x} = \mathbf{x}_0 + \mathbf{n}. \quad (3.1)$$

Then, the goal of graph-signal denoising is to remove as much noise as possible from the observed signal \mathbf{x} to estimate the original signal \mathbf{x}_0 , which is performed by exploiting the information encoded in \mathcal{G} .

Recall that a traditional approach for the graph-signal denoising task is to solve an optimization problem of the form

$$\hat{\mathbf{x}}_0 = \operatorname{argmin}_{\check{\mathbf{x}}_0} \|\mathbf{x} - \check{\mathbf{x}}_0\|_2^2 + \alpha R(\check{\mathbf{x}}_0|\mathcal{G}). \quad (3.2)$$

Algorithm 1: Proposed graph-signal denoising method

Inputs : \mathbf{x} and \mathcal{G} **Outputs:** $\hat{\mathbf{x}}_0$ and $\hat{\Theta}(\mathbf{x})$

- 1 Set $f_{\Theta}(\mathbf{Z}|\mathcal{G})$ as explained in Section 3.3 or Section 3.4
 - 2 Generate \mathbf{Z} from iid zero-mean Gaussian distribution
 - 3 Initialize $\Theta_{(0)}$ from iid zero-mean Gaussian
 - 4 **for** $t = 1$ **to** T **do**
 - 5 | Update $\Theta_{(t)}$ minimizing (3.3) with SGD
 - 6 **end**
 - 7 $\hat{\Theta}(\mathbf{x}) = \Theta_{(T)}$
 - 8 $\hat{\mathbf{x}}_0 = f_{\hat{\Theta}(\mathbf{x})}(\mathbf{Z}|\mathcal{G})$
-

The first term promotes fidelity to the signal observations, the regularizer $R(\cdot|\mathcal{G})$ promotes denoised signals with desirable properties over the given graph \mathcal{G} , and $\alpha > 0$ controls the influence of the regularization. Common choices for the regularizer include the quadratic Laplacian $R(\mathbf{x}|\mathcal{G}) = \mathbf{x}^\top \mathbf{L}\mathbf{x}$ [42], or regularizers involving high-pass graph filters $R(\mathbf{x}|\mathcal{G}) = \|\mathbf{H}\mathbf{x}\|_2^2$ that foster smoothness on the estimated signal [13, 25].

While those traditional approaches exhibit a number of advantages (including interpretability, mathematical tractability, and convexity), they may fail to capture more complex relations between \mathcal{G} and \mathbf{x}_0 , motivating the development of nonlinear graph-denoising approaches.

As summarized in Algorithm 1, in this chapter we advocate handling the graph-signal denoising task by employing an overparametrized GNN (denoted by $f_{\Theta}(\mathbf{Z}|\mathcal{G})$) as described in (2.21)-(2.22). The weights of the architecture, collected in Θ , are learned by minimizing the loss function

$$\mathcal{L}(\mathbf{x}, \Theta) = \frac{1}{2} \|\mathbf{x} - f_{\Theta}(\mathbf{Z}|\mathcal{G})\|_2^2, \quad (3.3)$$

applying stochastic gradient descent (SGD) in combination with early stopping to avoid overfitting the noise. The entries of the parameters Θ and the input matrix \mathbf{Z} are initialized at random using an i.i.d. zero-mean Gaussian distributions, and the weights learned after a few iterations of denoising the observation \mathbf{x} are denoted as $\hat{\Theta}(\mathbf{x})$. Note that \mathbf{Z} is fixed to its random initialization. Finally, the denoised graph signal estimate is computed as

$$\hat{\mathbf{x}}_0 = f_{\hat{\Theta}(\mathbf{x})}(\mathbf{Z}|\mathcal{G}). \quad (3.4)$$

The intuition behind this approach is as follows: since the architecture is overparametrized it can in principle fit any signal, including noise. However, as shown formally later, both empirically and theoretically, the proposed architectures fit graph signals faster than the noise and, therefore, with early stopping they fit most of the signal and little of the noise, enabling signal denoising.

Remark 1. The proposed architectures are described as *untrained* NNs because, when minimizing (3.3), the weights in Θ are learned to fit *each observation* \mathbf{x} , with the denoised signal $\hat{\mathbf{x}}_0$ being the output for those particular weights. This implies that each noisy-denoised signal pair $(\mathbf{x}, \hat{\mathbf{x}}_0)$ is associated with a particular value of the weights Θ , in contrasts with trainable NNs, where the weights Θ are first learned by fitting the signals in a *training set* and later used (unchanged) to denoise signals that were not in the training set.

Regarding the specific implementation of the untrained network $f_{\Theta}(\mathbf{Z}|\mathcal{G})$, there are multiple possibilities for selecting the linear and nonlinear transformations $\mathcal{T}_{\Theta^{(\ell)}}^{(\ell)}$ and $g^{(\ell)}$ defined in equa-

tions (2.21) and (2.22), respectively. As customary in NNs dealing with signals defined in \mathbb{R}^N , we select the ReLU operator, defined as $\text{ReLU}(x) = \max(0, x)$, to be the entrywise nonlinearity $g^{(\ell)}$. Then, we focus on the design of the linear transformation, which is responsible for incorporating the structure of the graph. The two following sections postulate the implementation of two particular linear transformations $\mathcal{T}_{\Theta^{(\ell)}}^{(\ell)}$ (each giving rise to a different GNN) and analyze the resulting architectures.

3.3 Graph convolutional generator

Our first architecture to address the graph-signal denoising task is a graph-convolutional generator (GCG) network that incorporates the topology of the graph into the NN pipeline via vertex-based graph convolutions. Then, leveraging the fact that convolutions of a graph signal on the vertex domain can be represented by a graph filter $\mathbf{H} \in \mathbb{R}^{N \times N}$ [17], we define the linear transformation for the convolutional generator as

$$\mathcal{T}_{\Theta^{(\ell)}}^{(\ell)}\{\mathbf{Y}^{(\ell-1)}|\mathcal{G}\} = \mathbf{H}\mathbf{Y}^{(\ell-1)}\Theta^{(\ell)}. \quad (3.5)$$

Remember that the $F^{(\ell-1)} \times F^{(\ell)}$ matrix $\Theta^{(\ell)}$ collects the learnable weights of the ℓ -th layer, and the graph filter \mathbf{H} is given by (2.1). The coefficients $\{h_r\}_{r=0}^{R-1}$ are fixed a priori so that \mathbf{H} promotes desired properties on the estimated signal. Using the linear transformation defined in (3.5), the output of the GCG with L layers is given by the recursion

$$\mathbf{Y}^{(\ell)} = \text{ReLU}(\mathbf{H}\mathbf{Y}^{(\ell-1)}\Theta^{(\ell)}), \quad \text{for } \ell = 1, \dots, L-1, \quad (3.6)$$

$$\mathbf{y}^{(L)} = \mathbf{H}\mathbf{Y}^{(L-1)}\Theta^{(L)}, \quad (3.7)$$

where $\mathbf{Y}^{(0)} = \mathbf{Z}$ denotes the random input and the ReLU is not applied in the last layer of the architecture. With the proposed linear transformation, the GCG learns to combine the features within each node by fitting the weights of the matrices $\Theta^{(\ell)}$ while the graph filter \mathbf{H} interpolates the signal by mixing features from $R-1$ neighborhoods.

Even though the proposed GCG exploits graph convolutions to incorporate the graph topology into the architecture, it is intrinsically different from other GCNNs. The linear transformation proposed in [35], arguably one of the most popular implementations of GCNNs, is given by

$$\mathcal{T}_{\Theta^{(\ell)}}^{(\ell)}\{\mathbf{Y}^{(\ell-1)}|\mathcal{G}\} = (\mathbf{A} + \mathbf{I})\mathbf{Y}^{(\ell-1)}\Theta^{(\ell)}. \quad (3.8)$$

Recalling the definition of graph filters in (2.1), it is evident that (3.8) is a particular case of our proposed linear transformation, obtained by setting the generative graph filter to $\mathbf{H} = \mathbf{A} + \mathbf{I}$, a low-pass graph filter of degree one. In addition to representing a more general scenario, (3.6) endows the GCG with two main advantages. First, the graph filter \mathbf{H} allows us to incorporate prior information on the signals to denoise, making our GCG architecture more suitable to denoise a (high-) low-frequency signal by employing a (high-) low-pass filter. Second, in (3.8) there is an equivalence between the depth of the network and the radius of the considered neighborhood, so that gathering information from nodes that are R hops apart requires a GNN with R layers. In contrast, with the architecture considered in (3.6), the same can be achieved by considering a GCG with L layers and a graph filter \mathbf{H} of degree R/L [17], reducing the number of learnable parameters and bypassing some of the well-known over-smoothing problems associated with (3.8) [81].

Next, we adopt some simplifying assumptions to provide theoretical guarantees on the denoising capability of the GCG (Section 3.3.1). Then, we rely on numerical evaluations to demonstrate that the results also hold in more general settings (Section 3.3.2).

3.3.1 Guaranteed denoising with the GCG

To formally prove that the proposed architecture can successfully denoise the observed graph signal \mathbf{x} , we consider a two-layer GCG given by

$$f_{\Theta}(\mathbf{Z}|\mathcal{G}) = \text{ReLU}(\mathbf{HZ}\Theta^{(1)})\theta^{(2)}, \quad (3.9)$$

where $\Theta^{(1)} \in \mathbb{R}^{F \times F}$ and $\theta^{(2)} \in \mathbb{R}^F$ are the learnable coefficients. With F denoting the number of features, we consider the overparametrized regime where $F \geq 2N$, and analyze the behavior and performance of denoising with the untrained network defined in (3.9).

We start by noting that scaling the i -th entry of $\theta^{(2)}$ is equivalent to scaling the i -th column of $\Theta^{(1)}$, so that, without loss of generality, we can set the weights to $\theta^{(2)} = \mathbf{b}$, where \mathbf{b} is a vector of size F with half of its entries set to $1/\sqrt{F}$ and the other half to $-1/\sqrt{F}$. Furthermore, since \mathbf{Z} is a random matrix of dimension $N \times F$, the column space of \mathbf{Z} spans \mathbb{R}^N , and hence, minimizing over $\mathbf{Z}\Theta^{(1)}$ is equivalent to minimizing over $\Theta \in \mathbb{R}^{N \times F}$. With these considerations in place, the optimization over (3.3) can be performed replacing the two-layer GCG described in (3.9) by its simplified form

$$f_{\Theta}(\mathbf{H}) = f_{\Theta}(\mathbf{Z}|\mathcal{G}) = \text{ReLU}(\mathbf{H}\Theta)\mathbf{b}. \quad (3.10)$$

Note that we replaced $f_{\Theta}(\mathbf{Z}|\mathcal{G})$ with $f_{\Theta}(\mathbf{H})$ since the graph influence is modeled by the graph filter \mathbf{H} , and the influence of the matrix \mathbf{Z} is absorbed by the learnable weights Θ .

The denoising capability of the two-layer architecture is related to the eigendecomposition of its expected squared Jacobian [82]. However, to understand which signals can be effectively denoised with the proposed architecture, we need to connect the spectral domain of the expected squared Jacobian with the spectrum of the graph, given by the eigenvectors of the adjacency matrix.

To that end, we next compute the expected squared Jacobian of the two-layer architecture in (3.10). Denote as $\mathcal{J}_{\Theta}(\mathbf{H}) \in \mathbb{R}^{N \times NF}$ the Jacobian matrix of $f_{\Theta}(\mathbf{H})$ with respect to Θ , which is given by

$$\mathcal{J}_{\Theta}^{\top}(\mathbf{H}) = \begin{bmatrix} b_1 \mathbf{H}^{\top} \text{diag}(\text{ReLU}'(\mathbf{H}\theta_1)) \\ \vdots \\ b_F \mathbf{H}^{\top} \text{diag}(\text{ReLU}'(\mathbf{H}\theta_F)) \end{bmatrix} \in \mathbb{R}^{NF \times N}, \quad (3.11)$$

where θ_i represents the i -th column of Θ , and ReLU' is the derivative of the ReLU, which is the Heaviside step function. Then, define the $N \times N$ expected squared Jacobian matrix as

$$\mathcal{X} := \mathbb{E}_{\Theta}[\mathcal{J}_{\Theta}(\mathbf{H})\mathcal{J}_{\Theta}^{\top}(\mathbf{H})] = \sum_{i=1}^F b_i^2 \mathbb{E} \left[\text{ReLU}'(\mathbf{H}\theta_i) \text{ReLU}'(\mathbf{H}\theta_i)^{\top} \right] \circ \mathbf{H}\mathbf{H}^{\top}. \quad (3.12)$$

Moreover, from the work in [83, Sec. 3.2], we note that $\mathbb{E} \left[\text{ReLU}'(\mathbf{H}\theta_i) \text{ReLU}'(\mathbf{H}\theta_i)^{\top} \right]$ is in fact the so-called dual activation of the step function. Therefore, combining the expression for the dual activation of the step function from [83, Table 1] with (3.12), we obtain that

$$\mathcal{X} = 0.5 \left(\mathbf{1}\mathbf{1}^{\top} - \pi^{-1} \arccos(\mathbf{C}^{-1}\mathbf{H}^2\mathbf{C}^{-1}) \right) \circ \mathbf{H}\mathbf{H}^{\top}, \quad (3.13)$$

where \circ represents the Hadamard (entry-wise) product, $\arccos(\cdot)$ is computed entry-wise, \mathbf{h}_i represents the i -th column (row) of \mathbf{H} , $\mathbf{C} = \text{diag}(\|\mathbf{h}_1\|_2, \dots, \|\mathbf{h}_N\|_2)$ is a normalization term so that $\mathbf{C}^{-1}\mathbf{H}^2\mathbf{C}^{-1}$ is the autocorrelation of the graph filter \mathbf{H} .

Since \mathcal{X} is symmetric and positive (semi) definite, it has an eigendecomposition $\mathcal{X} = \mathbf{W}\Sigma\mathbf{W}^\top$. Here, the columns of the orthonormal matrix $\mathbf{W} = [\mathbf{w}_1, \dots, \mathbf{w}_N]$ are the N eigenvectors, and the nonnegative eigenvalues in the diagonal matrix Σ are assumed to be ordered as $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_N$.

After defining the two-layer GCG $f_{\Theta}(\mathbf{H})$ and its expected square Jacobian \mathcal{X} , we formally analyze its performance when denoising bandlimited graph signals. This is particularly relevant given the importance of (approximate) bandlimited graph signals both from analytical and practical points of view [9]. For the sake of clarity, we first introduce the main result (Theorem 3.1) and then we detail a key intermediate result (Lemma 3.1) that provides additional insight.

Formally, consider the K -bandlimited graph signal \mathbf{x}_0 as described in (2.5), and let the architecture $f_{\Theta}(\mathbf{H})$ have a sufficiently large number of features F :

$$F \geq \left(\frac{\sigma_1^2}{\sigma_N^2} \right)^{26} \xi^{-8} N, \quad \text{with } \xi \in (0, (2 \log(2N/\phi))^{-\frac{1}{2}}) \quad (3.14)$$

being an error tolerance parameter for some prespecified ϕ . Then, for a specific set of graphs with minimum number of nodes $N_{\epsilon, \delta}$ that is introduced later in the section (cf. Ass. 3.1), if we solve (3.3) running gradient descent with a step size $\eta \leq \frac{1}{\sigma_1^2}$, the following result holds (see Appendix 3.7).

Theorem 3.1. *Let $f_{\Theta}(\mathbf{H})$ be the network defined in equation (3.10), and assume it is sufficiently wide, i.e., it satisfies condition (3.14) for some error tolerance parameter ξ . Let \mathbf{x}_0 be a K -bandlimited graph signal spanned by the eigenvectors \mathbf{V}_K , and let \mathbf{w}_i and σ_i be the i -th eigenvector and eigenvalue of \mathcal{X} . Let \mathbf{n} be the noise present in \mathbf{x} , set ϕ and ϵ to small positive numbers, and let the conditions from Ass. 3.1 hold. Then, for any ϵ, δ , there exists some $N_{\epsilon, \delta}$ such that if $N > N_{\epsilon, \delta}$, the error for each iteration t of gradient descent with stepsize η used to fit the architecture is bounded as*

$$\begin{aligned} \|\mathbf{x}_0 - f_{\Theta(t)}(\mathbf{H})\|_2 &\leq \left((1 - \eta\sigma_K^2)^t + \delta(1 - \eta\sigma_N^2)^t \right) \|\mathbf{x}_0\|_2 \\ &+ \xi \|\mathbf{x}\|_2 + \sqrt{\sum_{i=1}^N ((1 - \eta\sigma_i^2)^t - 1)^2 (\mathbf{w}_i^\top \mathbf{n})^2}, \end{aligned} \quad (3.15)$$

with probability at least $1 - e^{-F^2} - \phi - \epsilon$.

As explained next, the fitting (denoising) bound provided by the theorem first decreases and then increases with the number of iterations t . To be more precise, let us analyze separately each of the three terms in the right hand side of (3.15). The first term captures the part of the signal \mathbf{x}_0 that is fitted after t iterations while accounting for the misalignment of the eigenvectors \mathbf{V}_K and \mathbf{W}_K . This term decreases with t and, since δ can be made arbitrary small (cf. Lemma 3.1), vanishes for moderately low values of t . The second term is an error term that is negligible if the network is sufficiently wide. Therefore, ξ can be chosen to be sufficiently small by designing the architecture according to the condition in (3.14). Finally, the third term, which depends on the noise present in each of the spectral components of the squared Jacobian $(\mathbf{w}_i^\top \mathbf{n})^2$, grows with t . More specifically, if the σ_i associated with a spectral component is very small, the term $(1 - \eta\sigma_i^2)^t$ is close to 1 and, hence, the noise power in the i -th frequency will be small. Only when t grows very large the coefficient $(1 - \eta\sigma_i^2)^t$ vanishes and the i -th frequency component of the noise is fitted. As a result, if the filter \mathbf{H} is designed such that eigenvalues of the squared Jacobian satisfy that $\sigma_K \gg \sigma_{K+1}$, then there will be a range of moderate-to-high values of t for which: i) the first term is zero and ii) only the K strongest components of the noise have been fitted, so that the third term can be approximated as $\sqrt{\sum_{i=1}^K (\mathbf{w}_i^\top \mathbf{n})^2}$. Clearly, as t grows larger, the coefficient

$((1 - \eta\sigma_i^2)^t - 1)$ will also be close to one for $i > K$, meaning that additional components of the noise will be fitted as well, deteriorating the performance of the denoising architecture. This implies that if the optimization algorithm is stopped before t grows too large, the original signal is fitted along with the noise that aligns with the signal, but not the noise present in other components.

In other words, Theorem 3.1 not only characterizes the performance of the two-layer GNN, but also illustrates that, if early stopping is adopted, our overparametrized architecture is able to effectively denoise the bandlimited graph signal. This result is related to the error bound for denoising images presented in [82], where \mathbf{x}_0 is assumed to lie in the span of \mathbf{W}_K . However, when dealing with graphs, it is unclear which signals would satisfy this requirement. Motivated by this, we assume that \mathbf{x}_0 is a bandlimited signal (i.e., lies in the span of \mathbf{V}_K), which is a natural condition employed in many applications.

As a consequence, a critical step to attain Theorem 3.1 is to relate the eigenvectors of \mathcal{X} with those of the adjacency matrix \mathbf{A} , denoted as \mathbf{V} . To achieve this, we assume that \mathbf{A} is random and provide high-probability bounds between the leading eigenvectors of \mathbf{A} and \mathcal{X} . More specifically, consider a graph \mathcal{G} drawn from a SBM [84] with K communities. Also, denote by $\mathcal{M}(\mathcal{A})$ the SBM with expected adjacency matrix $\mathcal{A} = \mathbb{E}[\mathbf{A}]$, and by β_{min} the minimum expected degree $\beta_{min} := \min_i[\mathcal{A}\mathbf{1}]_i$. Given some $\rho > 0$, we define as $\mathcal{M}_N(\beta_{min}, \rho)$ the class of SBMs $\mathcal{M}(\mathcal{A})$ with N nodes for which $\beta_{min} = \omega(\ln(N/\rho))$, where $\omega(\cdot)$ denotes the (conventional) asymptotic dominance. Then, the condition of \mathcal{G} being drawn from this SBM whose expected minimum degree increases with N is formally expressed in the following assumption.

Assumption 3.1. *The model $\mathcal{M}(\mathcal{A})$ from which \mathbf{A} is drawn satisfies $\mathcal{M}(\mathcal{A}) \in \mathcal{M}_N(\beta_{min}, \rho)$, with $\beta_{min} = \omega(\ln(N/\rho))$.*

We note that, as discussed in [85], the minimal degree condition considered in Ass. 1 ensures that nodes belonging to the same community also belong to the same connected component with high probability, which is required to relate \mathbf{A} and \mathcal{A} . Under these conditions, the following result holds.

Lemma 3.1. *Let the matrix \mathcal{X} be defined as in (3.13), set ϵ and δ to small positive numbers, and denote by \mathbf{V}_K and \mathbf{W}_K the K leading eigenvectors in the respective eigendecompositions of \mathbf{A} and \mathcal{X} . Under Ass. 3.1, there exists an orthonormal matrix \mathbf{Q} and an integer $N_{\epsilon, \delta}$ such that, for $N > N_{\epsilon, \delta}$, the bound*

$$\|\mathbf{V}_K - \mathbf{W}_K \mathbf{Q}\|_F \leq \delta,$$

holds with probability at least $1 - \epsilon$.

The proof is provided in Appendix 3.8, and it leverages Ass. 1 to relate the eigenvectors \mathbf{V}_K and \mathbf{W}_K based on the eigenvectors of the expected values of \mathbf{A} and \mathcal{X} .

For a given K , Lemma 3.1 bounds the difference between the subspaces spanned by the K leading eigenvectors of \mathbf{A} and \mathcal{X} when graphs are big enough, a result that is key in obtaining Theorem 3.1. Moreover, the lemma shows that if the lower bound $N_{\epsilon, \delta}$ increases, then the error encoded δ becomes arbitrary small. Also note that, if a larger value of K is considered, then the minimum required graph size $N_{\epsilon, \delta}$ will also be larger. An inspection of (3.13) reveals that the result in Lemma 3.1 is not entirely unexpected. Indeed, since \mathbf{H} is a polynomial in \mathbf{A} , so is \mathbf{H}^2 . This implies that \mathbf{V} are also the eigenvectors of \mathbf{H}^2 , and because \mathbf{H}^2 appears twice on the right hand side of (3.13), a relationship between the eigenvectors of \mathcal{X} and \mathbf{V} can be anticipated. However, the presence of the Hadamard product and the (non Lipschitz continuous) nonlinearity \arccos renders the exact analysis of the eigenvectors a challenging task. Consequently, we resorted to a stochastic framework in deriving Lemma 3.1.

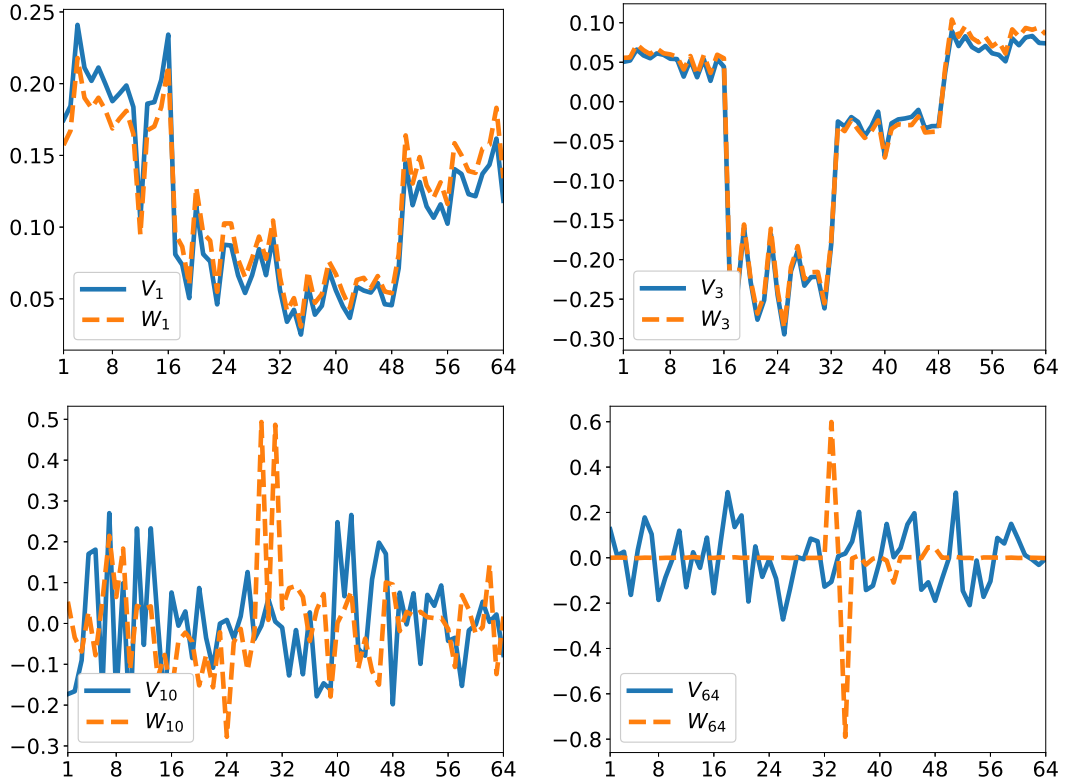


Figure 3.1: Comparison between the eigenvectors of the matrices \mathbf{A} and \mathcal{X} for an SBM graph with $N = 64$ nodes and $K = 4$ communities, and for a GCG of $L = 5$ layers. From left to right, the figures represent the first, third, tenth, and last eigenvectors.

3.3.2 Numerical inspection of the deep GCG spectrum

While for convenience, the previous section focused on analyzing the GCG architecture with $L = 2$ layers, in practice we often work with a larger number of layers. In this section, we provide numerical evidence showing that the relation between matrices \mathbf{A} and \mathcal{X} described in Lemma 3.1 also holds when $L > 2$.

To that end, Fig. 3.1 shows the pairs of eigenvectors \mathbf{v}_i and \mathbf{w}_i for the indexes $i = \{1, 3, 10, 64\}$, for a given graph \mathcal{G} drawn from an SBM with $N = 64$ nodes and 4 communities. The GCG is composed of $L = 5$ layers and, to obtain the eigenvectors of the squared Jacobian matrix, the Jacobian is computed using the *autograd* functionality of PyTorch. The nodes of the graph are sorted by communities, i.e., the first N_1 nodes belong to the first community and so on. It can be clearly seen that, even for moderately small graphs, the leading eigenvectors of \mathbf{A} and \mathcal{X} are almost identical, becoming more dissimilar as the eigenvectors are associated with smaller eigenvalues. It can also be observed how leading eigenvectors have similar values for entries associated with nodes within the same community. Moreover, Fig. 3.2 depicts the matrix product $\mathbf{V}^\top \mathbf{W}$, where it is observed that the $K = 4$ leading eigenvectors of both matrices are orthonormal. The presented numerical results strengthen the argument that the analytical results obtained for the two-layer case can be extrapolated to deeper architectures.

Another key assumption of Lemma 3.1 is that \mathcal{G} is drawn from the SBM described in $\mathcal{M}_N(\beta_{min}, \rho)$. This assumption facilitates the derivation of a bound relating the spectra of \mathbf{A} and \mathcal{X} (i.e., the subspaces spanned by the eigenvectors \mathbf{V}_K and \mathbf{W}_K). However, the results reported in Fig. 3.3

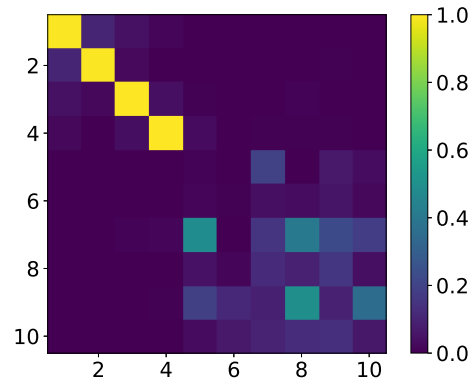


Figure 3.2: Heatmap representation of the matrix product $\mathbf{V}_K^\top \mathbf{W}_K$. The low values of the off-diagonal entries illustrate the orthogonality between both sets of eigenvectors. These eigenvectors are the same as those depicted in Fig. 3.1.

suggest that such a relation exists for other type of graphs, even though its analytical characterization is more challenging. The figure has 12 panels (3 columns and 4 rows). Each of the columns corresponds to a different graph, namely: 1) a realization of a small-world (SW) graph [86] with $N = 150$ nodes, 2) the Zachary’s Karate graph [87] with $N = 34$ nodes, and 3) a graph of $N = 316$ weather stations across the United States¹. Each of the three first rows correspond to an $N \times N$ matrix, namely: 1) the normalized adjacency matrix \mathbf{A} , 2) \mathbf{H}^2 , the squared version of a low pass graph filter and whose coefficients are drawn from a uniform distribution and set to unit ℓ_1 norm, and 3) the squared Jacobian matrix \mathcal{X} . Although we may observe some similarity between \mathbf{A} and \mathcal{X} , the relation between \mathcal{X} and the graph \mathcal{G} becomes apparent when comparing the matrices \mathbf{H}^2 and \mathcal{X} . The matrix \mathbf{H} is a random graph filter used in the linear transformation of the convolutional generator $f_\Theta(\mathbf{H})$, and it is clear that the vertex connectivity pattern of \mathcal{X} is related to that of \mathbf{H}^2 . Since \mathcal{X} and \mathbf{H}^2 are closely related and we know that the eigenvectors of \mathbf{H}^2 and those of \mathbf{A} are the same, we expect \mathbf{W} (the eigenvectors of \mathcal{X}) and \mathbf{V} (the eigenvectors of \mathbf{A}) to be related as well. To verify this, the fourth row of Fig. 3.3 represents $\mathbf{V}_K^\top \mathbf{W}_K$, i.e., the pairwise inner products of the K leading eigenvectors of \mathbf{A} and those of \mathcal{X} . It can be observed that the K leading eigenvectors are close to orthogonal, which means that the relation observed in the vertex domain carries over to the spectral domain and \mathbf{V}_K and \mathbf{W}_K expand the same subspace. These results suggest that a deep GCG could be able to denoising signals living in the subspace spanned by \mathbf{V}_K . However, because the bound in Theorem 3.1 assumed a 2-layer GCG, we address this hypothesis numerically in Section 3.5.

To summarize, the presented results illustrate that the analytical characterization provided in Section 3.3.1, which considered a 2-layer GCG operating over SBM graphs, carries over to more general setups.

3.4 Graph upsampling decoder

The GCG architecture presented in Section 3.3 incorporated the topology of \mathcal{G} via the vertex-based convolutions implemented by the graph filter \mathbf{H} . In this section, we introduce the graph decoder (GDec) architecture. In contrast to the GCG and other GCNNs, this novel graph-aware

¹Data extracted from the National Centers for Environmental Information. Available at <https://www.ncei.noaa.gov/data/global-summary-of-the-day>

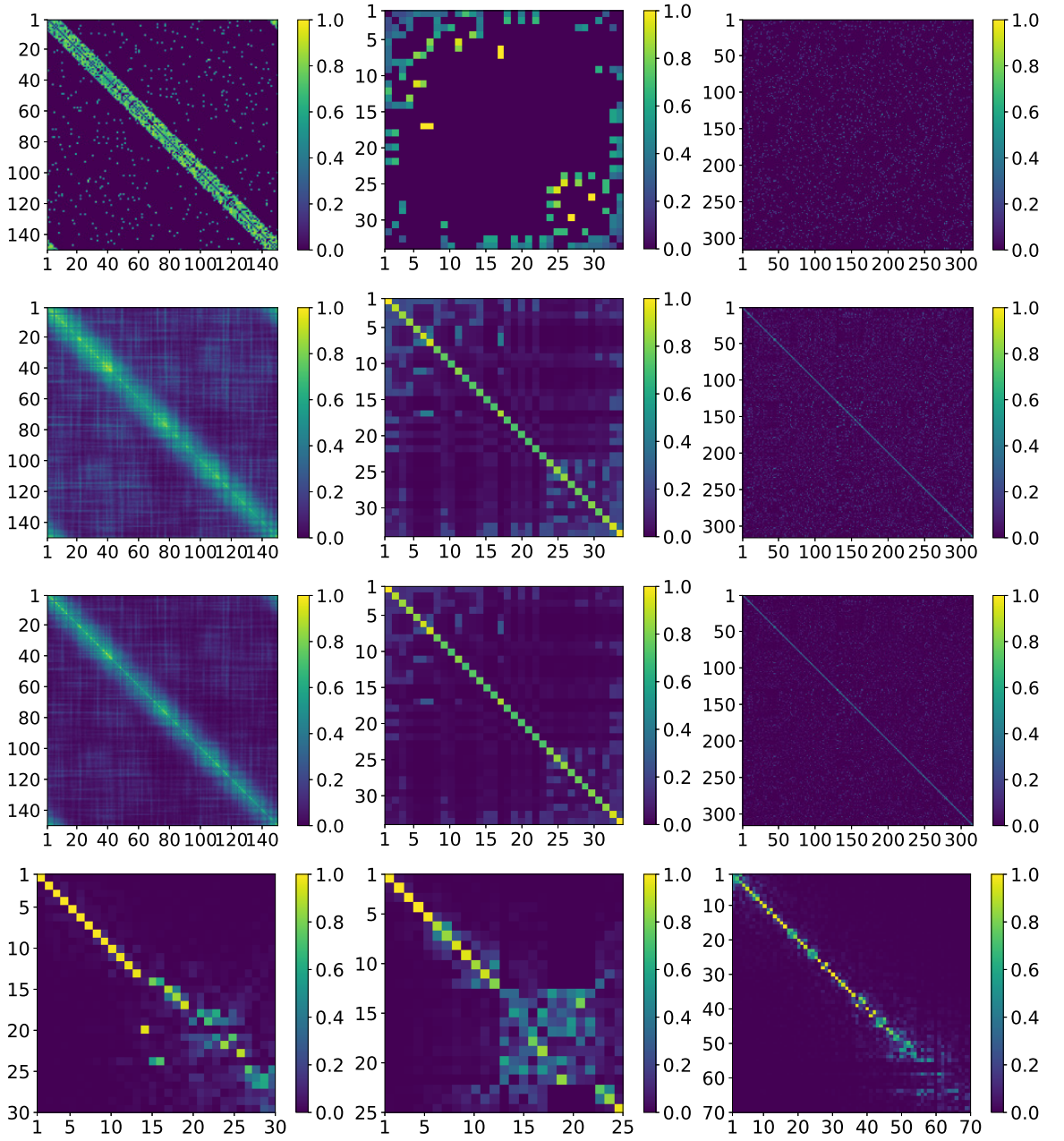


Figure 3.3: Illustrating the matrices \mathbf{A} , \mathbf{H}^2 , \mathcal{X} , and $\mathbf{V}_K^\top \mathbf{W}_K$, shown in rows 1, 2, 3, and 4, respectively, for different types of graphs. The column 1, 2, and 3 present a SW graph, the Zachary's Karate graph, and the weather stations graph. The graph filter \mathbf{H}^2 is created as a square graph filter with coefficients drawn from a uniform distribution and set to unit ℓ_1 norm. For each graph (column), it can be seen that the matrices \mathbf{A} , \mathbf{H}^2 , and \mathcal{X} are related, and that the leading eigenvectors \mathbf{V}_K and \mathbf{W}_K are close to orthogonal.

denoising NN incorporates the topology of \mathcal{G} via a (nested) collection of graph upsampling operators [62]. Specifically, we propose the linear transformation for the GDec denoiser to be given by

$$\mathcal{T}_{\Theta^{(\ell)}}^{(\ell)}\{\mathbf{Y}^{(\ell-1)}|\mathcal{G}\} = \mathbf{U}^{(\ell)}\mathbf{Y}^{(\ell-1)}\Theta^{(\ell)}, \quad (3.16)$$

where $\mathbf{U}^{(\ell)} \in \mathbb{R}^{N^{(\ell)} \times N^{(\ell-1)}}$, with $N^{(\ell)} \geq N^{(\ell-1)}$, are graph upsampling matrices to be defined soon. Note that, compared to (3.5), the graph filter \mathbf{H} is replaced with the upsampling operator $\mathbf{U}^{(\ell)}$ that *depends* on ℓ . Adopting the proposed linear transformation, the output of the GDec

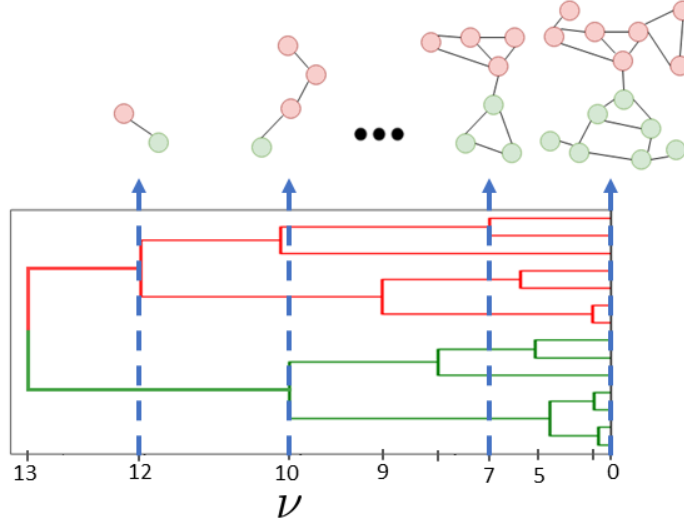


Figure 3.4: Dendrogram of an agglomerative hierarchical clustering algorithm and the resulting graphs with 2, 4, 7 and 14 nodes.

with L layers is given by the recursion

$$\mathbf{Y}^{(\ell)} = \text{ReLU}(\mathbf{U}^{(\ell)}\mathbf{Y}^{(\ell-1)}\Theta^{(\ell)}), \quad \text{for } \ell = 1, \dots, L-1, \quad (3.17)$$

$$\mathbf{y}^{(L)} = \mathbf{U}^{(L)}\mathbf{Y}^{(L-1)}\Theta^{(L)}, \quad (3.18)$$

where the ReLU is also removed from the last layer.

Similar to the GCG, the proposed GDec learns to combine the features within each node. However, the interpolation of the signals in this case is determined by the graph upsampling operators $\{\mathbf{U}^{(\ell)}\}_{\ell=1}^L$, rather than by employing convolutions. The size of the input $N^{(0)}$ is now a design parameter that will determine the implicit degrees of freedom of the architecture. Note that, from the GSP perspective, the input feature matrix $\mathbf{Y}^{(\ell-1)} \in \mathbb{R}^{N^{(\ell-1)} \times F^{(\ell-1)}}$ represents $F^{(\ell-1)}$ graph signals, each of them defined over a graph $\mathcal{G}^{(\ell-1)}$ with $N^{(\ell-1)}$ nodes. Therefore, even though the input $\mathbf{Y}^{(0)} = \mathbf{Z}$ is still a random white matrix across rows and columns, since $N^{(\ell)} \geq N^{(\ell-1)}$, the dimensionality of the input is progressively increasing.

A closer comparison with the GCG reveals that the smaller dimensionality of the input \mathbf{Z} endows the GDec architecture with fewer degrees of freedom, rendering the architecture more robust to noise. Not only that, but the graph information is now included via the graph upsampling operators $\mathbf{U}^{(\ell)}$ instead of relying on graph filters. Clearly, the method used to design the graph upsampling matrices, which is the subject of the next section, will have an impact on the type of graph signals that can be efficiently denoised using the GDec architecture.

3.4.1 Graph upsampling operator from hierarchical clustering

Regular upsampling operators have been successfully used in NN architectures to denoise signals defined on regular domains [82]. While the design of upsampling operators in regular grids is straightforward, when the signals at hand are defined on irregular domains the problem becomes substantially more challenging. The approach that we put forth in this chapter is to use agglomerative hierarchical clustering methods [78–80] to design a graph upsampling operator that leverages the graph topology. These methods take a graph as an input and return a dendrogram;

see Fig. 3.4. A dendrogram can be interpreted as a rooted-tree structure that shows different clusters at the different levels of resolution ν . At the finest resolution ($\nu = 0$) each node forms a cluster of its own. Then, as ν increases, nodes start to group together (agglomerate) in bigger clusters and, when the resolution becomes large (coarse) enough, all nodes end up being grouped in the same cluster.

By cutting the dendrogram at $L + 1$ resolutions, including $\nu = 0$, we obtain a collection of node sets with parent-child relationships inherited by the refinement of clusters. Since we are interested in performing graph upsampling, note that the dendrogram is interpreted from left to right. This can be observed in the example shown in Fig. 3.4, where the three red nodes in the second graph ($\nu = 10$, layer $\ell = 1$) are children of the red parent in the coarsest graph ($\nu = 12$, layer $\ell = 0$). In this sense, the graph upsampling operator is given by the inverse operation of the clustering algorithm. We leverage these parent-children relations to define the membership matrices $\mathbf{P}^{(\ell)} \in \{0, 1\}^{N^{(\ell)} \times N^{(\ell-1)}}$, where the entry $P_{ij}^{(\ell)} = 1$ only if the i -th node in layer ℓ is the child of the j -th node in layer $\ell - 1$. Moreover, we can further exploit the dendrogram to obtain coarser-resolution versions of the original graph \mathcal{G} . To that end, note that the clusters at layer ℓ can be interpreted as nodes of a graph $\mathcal{G}^{(\ell)}$ with $N^{(\ell)}$ nodes and adjacency matrix $\mathbf{A}^{(\ell)}$. There are several ways of defining $\mathbf{A}^{(\ell)}$ based on the original adjacency matrix \mathbf{A} . While our architecture does not focus on a particular form, in the simulations we set $A_{ij}^{(\ell)} \neq 0$ only if, in the original graph \mathcal{G} , there is at least one edge between nodes belonging to the cluster i and nodes from cluster j . In addition, the weight of the edge depends on the number of existing edges between the two clusters.

With the definition of the membership matrix $\mathbf{P}^{(\ell)}$ and the adjacency matrix $\mathbf{A}^{(\ell)}$, the upsampling operator of the ℓ -th layer is given by

$$\mathbf{U}^{(\ell)} = \left(\gamma \mathbf{I} + (1 - \gamma) \mathbf{A}^{(\ell)} \right) \mathbf{P}^{(\ell)}, \quad (3.19)$$

where $\gamma \in [0, 1]$ is a pre-specified constant. Notice that $\mathbf{U}^{(\ell)}$ first copies the signal value from the parents to the children by applying the matrix $\mathbf{P}^{(\ell)}$, and then every child performs a convex combination between this value and the average signal value of its neighbors. This design promotes that nodes descending from the same parent have similar (related) values, which conveys a notion (prior) of smoothness on the targeted graph signals. As we show in Section 3.5, the implicit smoothness prior results in a better performance when denoising smooth signals but, on the other hand, makes the architecture more sensitive to model mismatch. Therefore, when dealing with high-frequency signals, a worth-looking approach left as a future research direction is to rely on algorithms that cluster the nodes considering not only the topology of \mathcal{G} but also the properties of the graph signals.

Because the membership matrices $\mathbf{P}^{(\ell)}$ are designed using a clustering algorithm over \mathcal{G} , and the matrices $\mathbf{A}^{(\ell)}$ capture how strongly connected the clusters of layer ℓ are in the original graph, these two matrices are responsible for incorporating the information of \mathcal{G} into the upsampling operators $\mathbf{U}^{(\ell)}$. Furthermore, we remark that the upsampling operator $\mathbf{U}^{(\ell)}$ can be reinterpreted as the application of $\mathbf{P}^{(\ell)}$ followed by the application of a graph filter

$$\tilde{\mathbf{H}}^{(\ell)} = \gamma \mathbf{I} + (1 - \gamma) \mathbf{A}^{(\ell)}, \quad (3.20)$$

which sets the filter coefficients as $h_0 = \gamma$ and $h_1 = 1 - \gamma$.

3.4.2 Guaranteed denoising with the GDec

As we did for the GCG, our goal is to theoretically characterize the denoising performance of the GNN architecture defined by (3.17)-(3.19). To achieve that goal, we replicate the approach implemented in Section 3.3.1. We first derive the matrix \mathcal{X} and provide theoretical guarantees when denoising a K -bandlimited graph signal with the GDec. Then, to gain additional insight, we detail the relation between the subspace spanned by the eigenvectors \mathbf{W} and the spectral domain of \mathbf{A} . This relation is key in deriving the theoretical analysis.

We start by introducing the 2-layer GDec

$$f_{\Theta}(\mathbf{Z}|\mathcal{G}) = \text{ReLU}(\mathbf{UZ}\Theta^{(1)})\theta^{(2)}. \quad (3.21)$$

Upon following a reasoning similar to that provided after (3.10), instead of employing the previous architecture we can optimize (3.3) over its simplified version

$$f_{\Theta}(\mathbf{U}) = f_{\Theta}(\mathbf{Z}|\mathcal{G}) = \text{ReLU}(\mathbf{U}\Theta)\mathbf{b}. \quad (3.22)$$

An important difference with respect to the GCG presented in (3.10) is that the matrix Θ has a dimension of $N^{(0)} \times F$, so it spans $\mathbb{R}^{N^{(0)}}$ instead of \mathbb{R}^N . Since $N^{(0)} < N$, the smaller subspace spanned by the weights of the GDec renders the architecture more robust to fitting noise, but, on the other hand, the number of degrees of freedom to learn the graph signal of interest are reduced. As a result, the alignment between the targeted graph signals and the low-pass vertex-clustering architecture becomes more important.

The expected squared Jacobian $\mathcal{X} = \mathbb{E}_{\Theta}[\mathcal{J}_{\Theta}(\mathbf{U})\mathcal{J}_{\Theta}^{\top}(\mathbf{U})]$ is obtained following the procedure used to derive (3.13), arriving at the expression

$$\mathcal{X} = 0.5 \left(\mathbf{1}\mathbf{1}^{\top} - \frac{1}{\pi} \arccos(\tilde{\mathbf{C}}^{-1}\mathbf{U}\mathbf{U}^{\top}\tilde{\mathbf{C}}^{-1}) \right) \circ \mathbf{U}\mathbf{U}^{\top}, \quad (3.23)$$

where \mathbf{u}_i represents the i -th row of \mathbf{U} , and $\tilde{\mathbf{C}} = \text{diag}([\|\mathbf{u}_1\|_2, \dots, \|\mathbf{u}_N\|_2])$ is a normalization matrix.

Then, let \mathbf{x}_0 be a K -bandlimited graph signal and let $f_{\Theta}(\mathbf{U})$ have a number of features F satisfying (3.14). If we solve (3.3) running gradient descent with a step size $\eta \leq \frac{1}{\sigma_1^2}$, the following result holds.

Theorem 3.2. *Let $f_{\Theta}(\mathbf{U})$ be the network defined in equation (3.22). Consider the conditions described in Theorem 3.1 and let $N^{(0)}$ match the number of communities K (see Ass. 3.1). Then, for any ϵ, δ , there exists some $N_{\epsilon,\delta}$ such that if $N > N_{\epsilon,\delta}$, then the error for each iteration t of gradient descent with stepsize η used to fit the architecture is bounded as (3.15), with probability at least $1 - e^{-F^2} - \phi - \epsilon$.*

The proof of the theorem is analogous to the one provided in Appendix 3.7 but exploiting Lemma 3.2 instead of Lemma 3.1. Lemma 3.2 is fundamental in attaining Theorem 3.2 and is presented later in the section.

Theorem 3.2 formally establishes the denoising capability of the GDec when \mathbf{x}_0 is a K -bandlimited graph signal and $K = N^{(0)}$ matches the number of communities in the SBM graph. When compared with the GCG, the smaller dimensionality of the input \mathbf{Z} , and thus the smaller rank of the matrix Θ , constrains the learning capacity of the architecture, making it more robust to the presence of noise. However, this additional robustness also implies that the architecture is

more sensitive to model mismatch, since its capacity to learn arbitrary signals is smaller. Intuitively, the GDec represents an architecture tailored for a more specific family of graph signals than the GCG. Moreover, employing the GDec instead of the GCG has a significant impact on the relation between the subspaces spanned by \mathbf{V}_K and \mathbf{W}_K .

To establish the new relation between \mathbf{V}_K and \mathbf{W}_K , assume that the adjacency matrix is drawn from an SBM $\mathcal{M}(\mathcal{A})$ with K communities such that $\mathcal{M}(\mathcal{A}) \in \mathcal{M}_N(\beta_{min}, \rho)$, so that the SBM follows Ass. 3.1. In addition, set the size of the latent space to the number of communities so $N^{(0)} = K$. Under this setting, the counterpart to Lemma 3.1 for the case where $f_{\Theta}(\mathbf{U})$ is a GDec architecture follows.

Lemma 3.2. *Let the matrix \mathcal{X} be defined as in (3.23), set ϵ and δ to small positive numbers, and denote by \mathbf{V}_K and \mathbf{W}_K the K leading eigenvectors in the respective eigendecompositions of \mathbf{A} and \mathcal{X} . Under Ass. 3.1, there exist an orthonormal matrix \mathbf{Q} and an integer $N_{\epsilon, \delta}$ such that for $N > N_{\epsilon, \delta}$ the bound*

$$\|\mathbf{V}_K - \mathbf{W}_K \mathbf{Q}\|_F \leq \delta,$$

holds with probability at least $1 - \epsilon$.

Lemma 3.2 asserts that the difference between the subspaces spanned by \mathbf{V}_K and \mathbf{W}_K becomes arbitrarily small as the size of the graph increases. The proof is provided in Appendix 3.9 and the intuition behind it arises from the fact that the upsampling operator can be understood as $\mathbf{U} = \tilde{\mathbf{H}}\mathbf{P}$, where $\tilde{\mathbf{H}}$ is a graph filter of the specific form described in (3.20). Remember that \mathbf{P} is a binary matrix encoding the cluster in the layer $\ell - 1$ to which the nodes in the layer ℓ belong. Since we are only considering two layers, and we have that $N^{(0)} = K$, the matrix \mathbf{P} is encoding the node-community membership of the SBM graph and, hence, the product $\mathbf{P}\mathbf{P}^\top$ is a block matrix with constant entries matching the block pattern of \mathcal{A} . As shown in the proof, this property can be leveraged to bound the eigendecomposition of \mathbf{A} and \mathcal{X} .

3.4.3 Analyzing the deep GDec

The deep GDec composed of $L > 2$ layers can be constructed following the recursion presented in (3.17) and (3.18). In this case, by stacking more layers we perform the upsampling of the input signal in a progressive manner and, at the same time, we add more nonlinearities, which helps alleviating the rank constraint related to the input size $N^{(0)}$. In the absence of nonlinear functions, the maximum rank of the weights would be $N^{(0)}$, and thus, only signals in a subspace of size $N^{(0)}$ could be learned. By properly selecting the number of layers and the input size when constructing the network, we can obtain a trade-off between the robustness of the architecture and its learning capability.

In addition, the effect of adding more layers is also reflected on the smoothness assumption inherited from the construction of the upsampling operator. Adding more layers is related to less smooth signals, since the number of nodes in \mathcal{G} with a common parent, and thus, with similar values, is smaller.

We note that numerically illustrating that the bound between \mathbf{V}_K and \mathbf{W}_K holds true for the deep GDec, and that its denoising capability is not limited to signals defined over SBM graphs provide results similar to those in Section 3.3.2. Therefore, instead of replicating the previous section, we directly illustrate the performance of the deep GDec under more general settings in the following section, where we present the numerical evaluation of the proposed architectures.

3.5 Numerical results

This section presents different experiments to numerically validate the theoretical claims introduced in the chapter, and to illustrate the denoising performance of the GCG and the GDec. The experiments are carried out using synthetic and real-world data, and the proposed architectures are compared to other graph-signal denoising alternatives. The code for the experiments and the architectures is available on GitHub². For hyper-parameter settings and implementation details the interested reader is referred to the online available code.

3.5.1 Denoising capability of graph untrained architectures

The goal of the experiment shown in Figures 3.5a and 3.5b is to illustrate that the proposed graph untrained architectures are capable of learning the structured original signal \mathbf{x}_0 faster than the noise, which is one of the core claims of the chapter. To that end, we generate an SBM graph with $N = 64$ nodes and $K = 4$ communities, and define 3 different signals: (i) “Signal”: a piece-wise constant signal \mathbf{x}_0 with the value of each node being the label of its community; (ii) “Noise”: zero-mean white Gaussian noise \mathbf{n} with unit variance; and (iii) “Signal + Noise”: a noisy observation $\mathbf{x} = \mathbf{x}_0 + \mathbf{n}$ where the noise has a normalized power of 0.1. Fig. 3.5a and 3.5b show the normalized mean square error (NMSE), with the error for each realization being $\|\mathbf{x}_0 - \hat{\mathbf{x}}_0\|_2^2 / \|\mathbf{x}_0\|_2^2$. The mean is computed for 100 realizations of the noise as the number of epochs increases when the different signals are fitted by the 2-layer GCG and the 2-layer GDec, respectively. It can be seen how, in both cases, the error when fitting the noisy signal \mathbf{x} decreases for a few epochs until it reaches a minimum, and then starts to increase. This is because the proposed untrained architectures learn the signal \mathbf{x}_0 faster than the noise, but if they fit the observation for too many epochs, they start learning the noise as well and, hence, the NMSE increases. As stated by Theorem 3.1 and Theorem 3.2, this result illustrates that, if early stopping is applied, both architectures are capable of denoising the observed graph signals without a training step. It can also be noted that, under this setting, the GDec learns the signal \mathbf{x}_0 faster than the GCG and, at the same time, is more robust to the presence of noise. This can be seen as a consequence of GDec implicitly making stronger assumptions about the smoothness of the targeted signal.

The goal of the second test case is two-fold. First, it illustrates that the result presented in Lemma 3.1 is not constrained to the family of SBM (as specified by Ass. 1), but can be generalized to other families of random graphs as well. In addition, it measures the influence of the number of nodes in the discrepancies between \mathbf{V}_K and \mathbf{W}_K . To that end, Fig. 3.6 contains the mean eigenvector similarity measured as $\frac{1}{K} \|\mathbf{V}_K - \mathbf{W}_K \mathbf{Q}\|_F$ as a function of the number of nodes in the graph. The eigenvector similarity is computed for 50 realizations of random graphs and the presented error is the median of all the realizations. The random graph models considered are: the SBM (“SBM”), the connected caveman graph (“CAVE”) [88], the regular graph whose fixed degree increases with its size (“REG”), the small world graph (“SW”) [86], and the power law cluster graph model (“PLC”) [89]. The second term in the legend denotes the number of leading eigenvectors taken into account in each case, which depends on the number of active frequency components of the specific random graph model. We can clearly observe that for most of the random graph models, the eigenvector error goes to 0 as N increases and, furthermore, the error is below 10^{-1} even for moderately small graphs. This illustrates that, although the conditions assumed for Lemma 3.1 and Lemma 3.2 focus on the specific setting of the SBM, the results can be applied to a wider class of graphs. Here, the regular graphs are particularly interesting since

²https://github.com/reysam93/Graph_Deep_Decoder

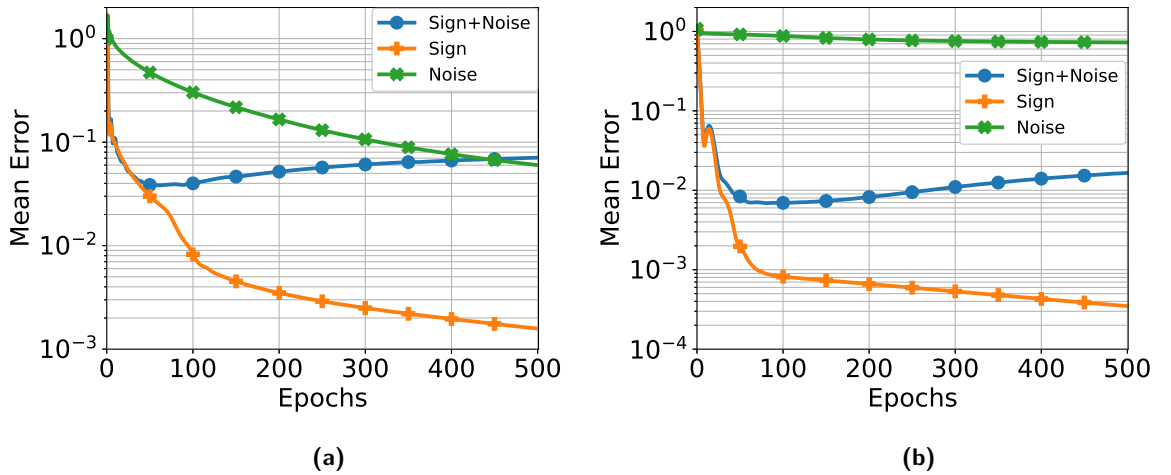


Figure 3.5: a) Error of the 2-layer GCG when fitting a piece-wise constant signal, noise, and a noisy signal, as a function of the number of epochs. The graph is drawn from an SBM with 64 nodes and 4 communities, and the normalized noise power is $P_n = 0.1$. b) Counterpart of a) but for the 2-layer GDec architecture.

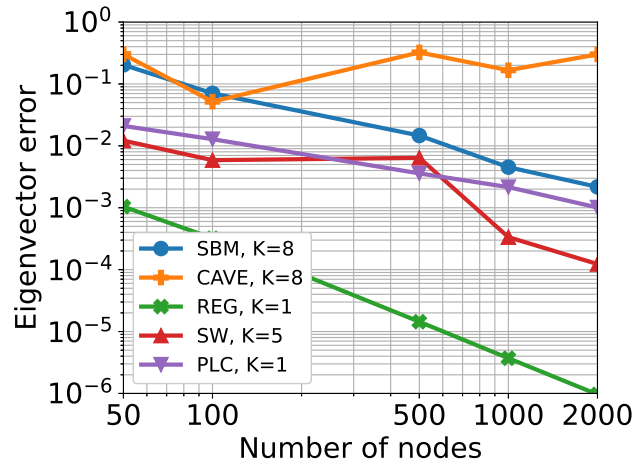


Figure 3.6: Mean distance between the K leading eigenvectors of the adjacency matrix and \mathcal{X} as a function of the graph size for several graph models.

most classical signals may be interpreted as signals defined over regular graphs. As a result, this empirical evidence motivates the extension of the proposed theorems to more general settings as a future line of work.

3.5.2 Denoising synthetic data

We now proceed to comment on the denoising performance of the proposed architectures with synthetic data. The usage of synthetic signals allows us to study how the properties of the noiseless signal influence the quality of the denoised estimate.

The first experiment, shown in Fig. 3.7, studies the error of the denoised estimate obtained with the 2-layer GCG as the number of epochs increases. The reported error is the NMSE of the estimated signal \hat{x}_0 , and the figure shows the mean values of 100 realizations of graphs and graph signals. The normalized power of the noise present in the data is 0.1. Graphs are drawn from

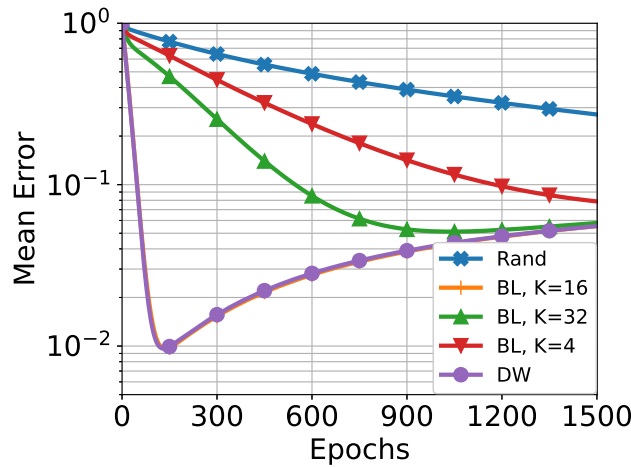


Figure 3.7: Median NMSE when the 2-layer GCG is used to denoise different families of graph signals.

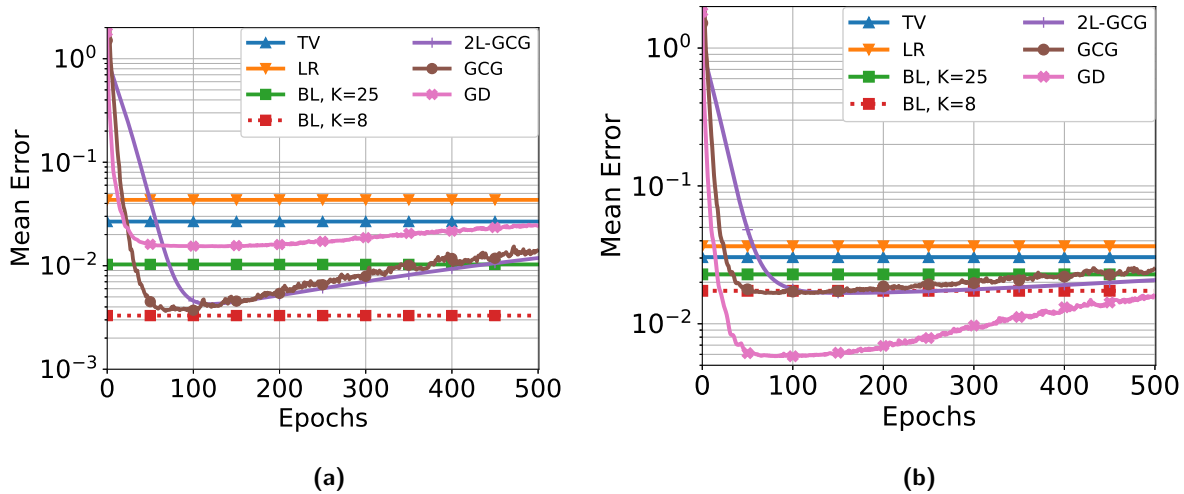


Figure 3.8: Median MSE when denoising a graph signal as a function of the number of epochs. a) Performance comparison between total variation, Laplacian regularization, bandlimited models, the 2-layer GCG, the deep GCG, and the deep GDec, when the signals are bandlimited. b) Counterpart of a) for the case where signals are diffused white.

an SBM with $N = 64$ nodes and 4 communities, and the graph signals are generated as: (i) a zero-mean white Gaussian noise with unit variance (“Rand”); (ii) a bandlimited graph signal (cf. 2.5) using the K leading eigenvectors of \mathbf{A} as base (“BL”); and (iii) a diffused white (“DW”) signal created as $\mathbf{y} = \text{med}(\mathbf{H}\mathbf{w}|\mathcal{G})$, where \mathbf{w} is a white vector whose entries are sampled from $\mathcal{N}(0, 1)$, \mathbf{H} is a low-pass graph filter, and $\text{med}(\cdot|\mathcal{G})$ represents the graph-aware median operator such that the value of the node i is the median of its neighborhood [43, 90]. The results in Fig. 3.7 show that the best denoising error is obtained when the signal is composed of just a small number of eigenvectors, and the performance deteriorates as the bandwidth (i.e., the number of leading eigenvectors that span the subspace where the signal lives) increases, obtaining the worst result when the signal is generated at random. This result is aligned with the theoretical claims since it is assumed that the signal \mathbf{x}_0 is bandlimited. It is also worth noting that the architecture also achieves a good denoising error with the “DW” model, showcasing that the GCG is also capable of denoising other types of smooth graph signals.

Next, Fig. 3.8a compares the performance of the 2-layer GCG (“2L-GCG”), the deep GCG (“GCG”) and the deep GDec (“GDec”) with the baseline models introduced in Section 3.2, which are the total variation (“TV”) [25], Laplacian regularization (“LR”) [42], and bandlimited model (“BL”) [21]. In this setting, the graphs are SBM with 256 nodes and 8 communities, and the signals are bandlimited with a bandwidth of 8. Since the “BL” model with $K = 8$ captures the actual generative model of the signal \mathbf{x}_0 , it achieves the best denoising performance. However, it is worth noting that the GCG obtains a similar result, outperforming the other alternatives. On the other hand, the “LR” obtains an error noticeably larger than that of “BL” and “GCG”, highlighting that, even though “BL” and “LR” are related models their different assumptions lead to different performances. Moreover, the benefits of using the deep GCG instead of the 2-layer architecture are apparent, since it achieves a better performance in fewer epochs.

On the other hand, Fig. 3.8b illustrates a similar experiment but with the graph signals generated as “DW”. Under this setting, it is clear that the GDec outperforms the other alternatives. These results showcase the benefits of employing a nonlinear architecture relative to classical denoising approaches. Furthermore, this experiment corroborates that the GDec is more robust to the presence of noise when the signals are aligned with the prior implicitly captured by the architecture.

3.5.3 Denoising real-world signals

Finally, we assess the performance of the proposed architectures in several real-world datasets. To the baselines considered in the previous experiments, we add the following competitive denoising algorithms: graph trend filtering (“GTF”) [41], a graph-aware median operator (“MED”) [43], a GCNN (“GCNN”) implemented as in [35], a graph attention network (“GAT”) [91], a Kron reduction-based autoencoder (“K-GAE”) [92], and the graph unrolling sparse coding architecture (“GUSC”) in [45]. Moreover, we consider the following noise distributions: (i) zero-mean Gaussian distribution, which is the noise model typically assumed for sensor measurements in signal processing; (ii) uniform distribution on some interval $[0, a]$, where $a \in \mathbb{R}_+$ is chosen accordingly to the desired noise power; and (iii) Bernoulli distribution to model errors in binary signals. Next, we describe the selected datasets and analyze the achieved results, which are summarized in Table 3.1.

Temperature. We consider a network of 316 weather stations distributed across the United States [13]. Graph signals represent daily temperature measurements in the first three months of the year 2003. The graph \mathcal{G} represents the geographical distance between weather stations and is given by the 8-nearest neighbors graph. The first and second rows of Table 3.1 list the NMSE when the noise is drawn from a Gaussian and a uniform distribution, respectively. In both cases, the noise has a normalized power of 0.3. It is clear that the GDec architecture outperforms the alternatives in both scenarios. Furthermore, we can observe that the GCG achieves a better performance than GCNN, showcasing the benefits of being able to use a more general graph filter.

S&P 500. In this experiment, we have 189 nodes representing stocks belonging to 6 different sectors of the S&P 500 with the graph signals representing the prices of those stocks at particular time instants. We follow [93] to estimate the graph \mathcal{G} assuming that the signals are drawn from a multivariate Gaussian distribution and are smooth on \mathcal{G} . We consider the noise specifications described in the previous dataset and provide the NMSE in the third and fourth rows of Table 3.1. It is worth noting that considering Gaussian noise in this dataset constitutes a more challenging denoising problem than using uniform noise. A plausible explanation is that the graph is estimated assuming that the data follows a Gaussian distribution, and hence, it is harder to separate the

Table 3.1: Denoising error of several datasets with different types of random noise

DATASET (METRIC)	METHOD	BL	TV	LR	GTF	MED	GCNN	GAT	K-GAE	GUSC	GCG	GDec
TEMPERATURE (NMSE)	Gaussian	0.062	0.117	0.095	0.066	0.053	0.123	0.045	0.134	0.044	0.056	0.035
	Uniform	0.063	0.117	0.094	0.064	0.053	0.118	0.047	0.136	0.049	0.057	0.036
S&P 500 (NMSE)	Gaussian	0.350	0.238	0.231	0.239	0.319	0.252	0.199	0.354	0.203	0.188	0.188
	Uniform	0.216	0.246	0.161	0.298	0.340	0.091	0.222	0.273	0.127	0.094	0.121
CORA (ERROR RATE)	Whole \mathcal{G}	0.154	0.142	0.115	0.126	0.167	0.099	0.141	0.135	0.099	0.093	0.121
	Conn. comp.	0.151	0.141	0.105	0.116	0.165	0.093	0.139	0.135	0.094	0.088	0.125

Gaussian noise from the true signals. In the presence of Gaussian noise, the GCG and the GDec outperform the other 8 alternatives. However, when the noise follows a uniform distribution, the best performance is obtained by the GCG and the GCNN, with GDec being the third best. In addition, we observe that traditional methods yield an error that is considerably larger than that incurred by the proposed architectures. This is aligned with our initial intuition about linear and quadratic methods being more limited when the actual relation between \mathbf{x}_0 and \mathcal{G} is more intricate, as is the case for financial data.

Cora. Lastly, we consider the Cora citation network dataset [35]. Nodes represent different scientific documents and edges capture citations among them. Like in [45], we consider the 7 class labels as binary graph signals encoding if the particular node belongs to that class. For each signal, we consider 25 realizations of Bernoulli noise that randomly flips 30% of the binary values of the signals, resulting in a total of 175 noisy graph signals. With the error rate denoting the proportion of labels correctly recovered after the denoising process, Table 3.1 shows the error metric averaged over all the signals. Moreover, since the graph is formed by several connected components, we report two results: the error rate when the whole graph is considered (fifth row) and the error rate when only the largest connected component is considered (sixth row). It can be seen that the GCG yields the best performance in both cases.

3.6 Conclusion

In this chapter, we faced the relevant task of graph-signal denoising. To approach this problem, we presented two overparametrized and untrained GNNs and provided theoretical guarantees on the denoising performance of both architectures when denoising K -bandlimited graph signals under some simplifying assumptions. Moreover, we numerically illustrated that the proposed architectures are also capable of denoising graph signals in more general settings. The key difference between the two architectures resided in the linear transformation that incorporates the information encoded in the graph. The GCG employs fixed (non-learnable) low-pass graph filters to model convolutions in the vertex domain, promoting smooth estimates. On the other hand, the GDec relies on a nested collection of graph upsampling operators to progressively increase the input size, limiting the degrees of freedom of the architecture, and providing more robustness to noise. In addition to the aforementioned analysis, we tested the validity of the proposed theorems and evaluated the performance of both architectures with real and synthetic datasets, showcasing a better performance than other classical and nonlinear methods for graph-signal denoising. Finally, we consider extending the results from Theorem 3.1 and Theorem 3.2 to more general scenarios as an interesting future line of work.

3.7 Appendix: Proof of Theorem 3.1

Let \mathbf{x}_0 be a K bandlimited graph signal as described in (2.5), which is spanned by the K leading eigenvectors of the graph \mathbf{V}_K , with $\tilde{\mathbf{x}}_0$ denoting its frequency representation. Let \mathbf{Q} be an orthonormal matrix that aligns the subspaces spanned by \mathbf{V}_K and \mathbf{W}_K , and denote as $\bar{\mathbf{x}}_0 = \mathbf{W}_K \mathbf{Q} \tilde{\mathbf{x}}_0$ the bandlimited signal using \mathbf{W}_K as basis and whose frequency response is also $\tilde{\mathbf{x}}_0$. Note that $\bar{\mathbf{x}}_0$ can be interpreted as recovering \mathbf{x}_0 from its frequency response using \mathbf{W}_K in lieu of \mathbf{V}_K . Also, note that $\mathbf{x}_0 - \bar{\mathbf{x}}_0 = (\mathbf{V}_K - \mathbf{W}_K \mathbf{Q}) \tilde{\mathbf{x}}_0$ represents the error between the signal \mathbf{x}_0 and its approximation inside the subspace spanned by \mathbf{W}_K . With these definitions in place, in [82, Th. 3] the authors showed that error when denoising a signal $\mathbf{x} = \mathbf{x}_0 + \mathbf{n}$ is bounded with probability at least $1 - e^{-F^2} - \phi$ by

$$\begin{aligned} \|\mathbf{x}_0 - f_{\Theta(t)}(\mathbf{Z}|\mathcal{G})\|_2 &\leq \|\Xi \mathbf{x}_0\|_2 + \xi \|\mathbf{x}\|_2 \\ &+ \sqrt{\sum_{i=1}^N ((1 - \eta \sigma_i^2)^t - 1)^2 (\mathbf{w}_i^\top \mathbf{n})^2}, \end{aligned} \quad (3.24)$$

with $\Xi := \mathbf{W}(\mathbf{I}_N - \eta \Sigma^2)^t \mathbf{W}^\top$, and \mathbf{I}_N the $N \times N$ identity matrix. However, note that the bound provided for $\|\Xi \mathbf{x}_0\|_2$ in [82] requires that \mathbf{x}_0 lies in the subspace spanned by \mathbf{W}_K , which is not the case. As a result, we further bound this term as

$$\begin{aligned} \|\Xi \mathbf{x}_0\|_2 &= \|\Xi(\mathbf{x}_0 + \bar{\mathbf{x}}_0 - \bar{\mathbf{x}}_0)\|_2 \\ &\stackrel{(i)}{=} \|\Xi_K \bar{\mathbf{x}}_0 + \Xi(\mathbf{V}_K - \mathbf{W}_K \mathbf{Q}) \tilde{\mathbf{x}}_0\|_2 \\ &\stackrel{(ii)}{\leq} \|\Xi_K \bar{\mathbf{x}}_0\|_2 + \|\Xi(\mathbf{V}_K - \mathbf{W}_K \mathbf{Q}) \tilde{\mathbf{x}}_0\|_2 \\ &\stackrel{(iii)}{\leq} \|\Xi_K\|_2 \|\bar{\mathbf{x}}_0\|_2 + \|\Xi\|_2 \|\mathbf{V}_K - \mathbf{W}_K \mathbf{Q}\|_F \|\tilde{\mathbf{x}}_0\|_2 \\ &\stackrel{(iv)}{\leq} (\|\Xi_K\|_2 + \delta \|\Xi\|_2) \|\mathbf{x}_0\|_2 \\ &\stackrel{(v)}{=} \left((1 - \eta \sigma_K^2)^t + \delta (1 - \eta \sigma_N^2)^t \right) \|\mathbf{x}_0\|_2. \end{aligned} \quad (3.25)$$

Here, $\Xi_K := \mathbf{W}_K(\mathbf{I}_K - \eta \Sigma_K^2)^t \mathbf{W}_K^\top$, and Σ_K represents a diagonal matrix containing the first K leading eigenvalues σ_k . We have that (i) follows from $\bar{\mathbf{x}}_0$ being bandlimited in \mathbf{W}_K , so $\Xi \bar{\mathbf{x}}_0 = \Xi_K \bar{\mathbf{x}}_0$. Then, (ii) follows from the triangle inequality, and (iii) from the ℓ_2 norm being submultiplicative and using the Frobenius norm as an upper bound for the ℓ_2 norm. In (iv) we apply the result of Lemma 3.1, which holds with probability at least $1 - \epsilon$ because $N > N_{\epsilon, \delta}$, and the fact that, since both \mathbf{W}_K and \mathbf{V}_K are orthonormal matrices, we have that $\|\mathbf{x}_0\|_2 = \|\bar{\mathbf{x}}_0\|_2 = \|\tilde{\mathbf{x}}_0\|_2$. We obtain (v) from the largest eigenvalues present in Ξ_K and Ξ .

Finally, the proof concludes by combining (3.25) and (3.24).

3.8 Appendix: Proof of Lemma 3.1

Define $\tilde{\mathcal{A}}$ as $\tilde{\mathcal{A}} := \mathbb{E}[\tilde{\mathbf{A}}] = \mathbb{E}[\mathbf{D}]^{-\frac{1}{2}} \mathcal{A} \mathbb{E}[\mathbf{D}]^{-\frac{1}{2}}$ and let \mathcal{X} be given by (3.13). Denote by \mathcal{H} a graph filter defined as a polynomial of the expected adjacency matrix $\tilde{\mathcal{A}}$, and let $\tilde{\mathcal{X}}$ be the expected squared Jacobian using the graph filter \mathcal{H} , i.e.,

$$\tilde{\mathcal{X}} = 0.5 \left(\mathbf{1}\mathbf{1}^\top - \frac{1}{\pi} \arccos(\mathbf{C}^{-1} \mathcal{H}^2 \mathbf{C}^{-1}) \right) \circ \mathcal{H}^2, \quad (3.26)$$

where \mathcal{C} is the counterpart of \mathbf{C} in (3.13), but using \mathcal{H} instead of \mathbf{H} . Given the following eigendecompositions $\tilde{\mathbf{A}} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^\top$, $\mathcal{X} = \mathbf{W}\mathbf{\Sigma}\mathbf{W}^\top$, $\tilde{\mathcal{A}} = \bar{\mathbf{V}}\bar{\mathbf{\Lambda}}\bar{\mathbf{V}}^\top$, and $\bar{\mathcal{X}} = \bar{\mathbf{W}}\bar{\mathbf{\Sigma}}\bar{\mathbf{W}}^\top$, for arbitrary orthonormal matrices \mathbf{T} and \mathbf{R} , we have that

$$\|\mathbf{V}_K - \mathbf{W}_K\mathbf{Q}\|_F \leq \|\mathbf{V}_K - \bar{\mathbf{V}}_K\mathbf{T}\|_F + \|\bar{\mathbf{V}}_K\mathbf{T} - \bar{\mathbf{W}}_K\mathbf{R}\|_F + \|\bar{\mathbf{W}}_K\mathbf{R} - \mathbf{W}_K\mathbf{Q}\|_F. \quad (3.27)$$

To prove the theorem, we bound the three terms on the right hand side of (3.27).

Bounding $\|\bar{\mathbf{V}}_K\mathbf{T} - \bar{\mathbf{W}}_K\mathbf{R}\|_F$. From the definition of an SBM, it follows that $\mathcal{A} = \mathbb{E}[\mathbf{A}] = \mathbf{B}\mathbf{\Omega}\mathbf{B}^\top$, where $\mathbf{B} \in \{0, 1\}^{N \times K}$ is an indicator matrix encoding the community to which each node belongs, and $\mathbf{\Omega}$ is a $K \times K$ matrix encoding the link probability between the communities of the graph. Therefore, $\tilde{\mathbf{A}}$ and $\tilde{\mathcal{A}}$ are both block matrices whose blocks coincide with the communities in the SBM. This implies that the eigenvectors associated with non-zero eigenvalues must span the columns of \mathbf{B} . Hence, the leading eigenvectors must be related by an orthonormal transformation, from where it follows that, given \mathbf{T} , we can always find \mathbf{R} such that

$$\|\bar{\mathbf{V}}_K\mathbf{T} - \bar{\mathbf{W}}_K\mathbf{R}\|_F = 0. \quad (3.28)$$

Bounding $\|\mathbf{V}_K - \bar{\mathbf{V}}_K\mathbf{T}\|_F$. Under Ass. 3.1, as it is shown in [85], with probability at least $1 - \rho$ we have that

$$\|\tilde{\mathbf{A}} - \tilde{\mathcal{A}}\| \leq 3\sqrt{\frac{3\ln(4N/\rho)}{\beta_{\min}}}. \quad (3.29)$$

Then, we combine the concentration (3.29) with the Davis-Kahan results [94, Th. 2], which bound the distance between the subspaces spanned by the population eigenvectors ($\bar{\mathbf{V}}_K$) and their sample version (\mathbf{V}_K). Denoting as $\bar{\lambda}_i$ the i -th eigenvalue collected in $\bar{\mathbf{\Lambda}}$, i.e. $\bar{\lambda}_i = \bar{\Lambda}_{ii}$, we obtain that there exists an orthonormal matrix \mathbf{T} such that

$$\|\mathbf{V}_K - \bar{\mathbf{V}}_K\mathbf{T}\|_F \leq \frac{\sqrt{8K}}{\bar{\lambda}_K - \bar{\lambda}_{K+1}} \|\tilde{\mathbf{A}} - \tilde{\mathcal{A}}\|_F \leq \frac{3\sqrt{8K}}{\bar{\lambda}_K} \sqrt{\frac{3\ln(4N/\rho)}{\beta_{\min}}}, \quad (3.30)$$

where we note that, since $\tilde{\mathcal{A}}$ follows an SBM, then $\bar{\lambda}_i = 0$ for all $i > K$.

Since $\beta_{\min} = \omega(\ln(N/\rho))$, we obtain that

$$\|\mathbf{V}_K - \bar{\mathbf{V}}_K\mathbf{T}\|_F \rightarrow 0, \quad \text{as } N \rightarrow \infty. \quad (3.31)$$

Bounding $\|\bar{\mathbf{W}}_K\mathbf{R} - \mathbf{W}_K\mathbf{Q}\|_F$. If we show that $\|\mathcal{X} - \bar{\mathcal{X}}\| \rightarrow 0$ as $N \rightarrow \infty$, we can then mimic the procedure in (3.29) and (3.30) to show that the difference between the leading K eigenvectors of \mathcal{X} and $\bar{\mathcal{X}}$ also vanishes. Hence, we are left to show that $\|\mathcal{X} - \bar{\mathcal{X}}\| \rightarrow 0$ as $N \rightarrow \infty$. From the definitions of \mathcal{X} and $\bar{\mathcal{X}}$, it follows that

$$\begin{aligned} \|\mathcal{X} - \bar{\mathcal{X}}\| &\leq 0.5\|\mathbf{H}^2 - \mathcal{H}^2\| + \frac{1}{2\pi} \|\arccos(\mathbf{C}^{-1}\mathcal{H}^2\mathbf{C}^{-1}) \circ \mathcal{H}^2 \\ &\quad - \arccos(\mathbf{C}^{-1}\mathbf{H}^2\mathbf{C}^{-1}) \circ \mathbf{H}^2\|. \end{aligned} \quad (3.32)$$

To bound the difference between the sampled and expected filters, we have that

$$\begin{aligned} \|\mathbf{H}^2 - \mathcal{H}^2\| &= \left\| \left(\sum_{\ell=0}^L h_\ell \tilde{\mathbf{A}}^\ell \right)^2 - \left(\sum_{\ell=0}^L h_\ell \tilde{\mathcal{A}}^\ell \right)^2 \right\| \\ &= \left\| \sum_{\ell=0}^{2L} \alpha_\ell (\tilde{\mathbf{A}}^\ell - \tilde{\mathcal{A}}^\ell) \right\| \leq \sum_{\ell=0}^{2L} \alpha_\ell \|\tilde{\mathbf{A}}^\ell - \tilde{\mathcal{A}}^\ell\|, \end{aligned} \quad (3.33)$$

for suitable coefficients α_ℓ and recalling that $L = 2$. Then, we can then leverage the fact that $\|\tilde{\mathbf{A}}\| = \|\tilde{\mathcal{A}}\| = 1$ to see that $\|\tilde{\mathbf{A}}^\ell - \tilde{\mathcal{A}}^\ell\| \leq \ell \|\tilde{\mathbf{A}} - \tilde{\mathcal{A}}\|$. We thus get that

$$\|\mathbf{H}^2 - \mathcal{H}^2\| \leq \sum_{\ell=0}^{2L} \ell \alpha_\ell \|\tilde{\mathbf{A}} - \tilde{\mathcal{A}}\| \rightarrow 0, \quad \text{as } N \rightarrow \infty, \quad (3.34)$$

where the limiting behavior follows from (3.29). Finally, to bound the second term in (3.32), we first note that the argument of the norm can be re-written as $\arccos(\mathbf{C}^{-1}\mathbf{H}^2\mathbf{C}^{-1}) \circ (\mathcal{H}^2 - \mathbf{H}^2) + (\arccos(\mathbf{C}^{-1}\mathcal{H}^2\mathbf{C}^{-1}) - \arccos(\mathbf{C}^{-1}\mathbf{H}^2\mathbf{C}^{-1})) \circ \mathcal{H}^2$. The limit in (3.34) ensures that the first of these two terms vanishes. Similarly, it follows that $\|\mathbf{C}^{-1}\mathcal{H}^2\mathbf{C}^{-1} - \mathbf{C}^{-1}\mathbf{H}^2\mathbf{C}^{-1}\| \rightarrow 0$ which, combined with the fact that \arccos is a uniformly continuous function, we can always find an $N_{\delta'}$ such that $\|\arccos(\mathbf{C}^{-1}\mathcal{H}^2\mathbf{C}^{-1}) - \arccos(\mathbf{C}^{-1}\mathbf{H}^2\mathbf{C}^{-1})\| \leq \delta'$ with high probability. Combining this result with (3.34) and applying the Davis-Kahan Theorem as done to obtain (3.30) we get that

$$\|\bar{\mathbf{W}}_K \mathbf{R} - \mathbf{W}_K \mathbf{Q}\|_F \rightarrow 0, \quad \text{as } N \rightarrow \infty. \quad (3.35)$$

Replacing (3.28), (3.31), and (3.35) into (3.27) our result follows.

3.9 Appendix: Proof of Lemma 3.2

Recall that $\tilde{\mathbf{A}} = \mathbb{E}[\tilde{\mathbf{A}}]$, and define $\tilde{\mathcal{H}} := \gamma \mathbf{I} + (1 - \gamma)\tilde{\mathbf{A}}$ as the specific graph filter introduced in Section 3.4.1 as a polynomial of $\tilde{\mathbf{A}}$. Let \mathcal{X} be given by equation (3.23), and denote by $\bar{\mathcal{X}}$ the expected squared Jacobian using the graph filter \mathcal{H} , i.e.,

$$\bar{\mathcal{X}} = 0.5 \left(\mathbf{1}\mathbf{1}^\top - \frac{1}{\pi} \arccos(\tilde{\mathbf{C}}^{-1} \mathbf{u}\mathbf{u}^\top \tilde{\mathbf{C}}^{-1}) \right) \circ \mathbf{u}\mathbf{u}^\top \quad (3.36)$$

with $\mathbf{u} = \tilde{\mathcal{H}}\mathbf{P}$ and where the matrix $\tilde{\mathbf{C}}$ is the counterpart of $\tilde{\mathbf{C}}$ in (3.23), but using \mathbf{u} in lieu of \mathbf{U} . Given the eigendecompositions $\tilde{\mathbf{A}} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^\top$, $\mathcal{X} = \mathbf{W}\mathbf{\Sigma}\mathbf{W}^\top$, $\tilde{\mathbf{A}} = \bar{\mathbf{V}}\bar{\mathbf{\Lambda}}\bar{\mathbf{V}}^\top$, and $\bar{\mathcal{X}} = \bar{\mathbf{W}}\bar{\mathbf{\Sigma}}\bar{\mathbf{W}}^\top$, analogously to Lemma 3.1, we bound the difference between \mathbf{V}_K and $\bar{\mathbf{W}}_K$ by bounding the three terms in the right hand side of (3.27).

Bounding $\|\bar{\mathbf{V}}_K \mathbf{T} - \bar{\mathbf{W}}_K \mathbf{R}\|$. We have that $\mathbf{u}\mathbf{u}^\top = \tilde{\mathcal{H}}\mathbf{P}\mathbf{P}^\top \tilde{\mathcal{H}}^\top$. Since \mathbf{P} is a binary matrix indicating to which community belongs each node, $\mathbf{P}\mathbf{P}^\top$ is a block diagonal matrix that captures the structure of the communities of the SBM. Then, because $\tilde{\mathcal{H}}$ is also block matrix with the same block pattern that the SBM, it turns out that the matrix $\bar{\mathcal{X}}$ is also a block matrix whose blocks coincide with the communities in the SBM graph. Therefore, the rest of the bound is analogous to that in Lemma 3.1.

Bounding $\|\mathbf{V}_K - \bar{\mathbf{V}}_K \mathbf{T}\|$. The relation between \mathbf{A} and $\tilde{\mathcal{A}}$ is the same as in Lemma 3.1 so the bound provided in (3.31) holds.

Bounding $\|\bar{\mathbf{W}}_K \mathbf{R} - \mathbf{W}_K \mathbf{Q}\|$. To derive this bound we show that $\|\mathbf{U}\mathbf{U}^\top - \mathbf{u}\mathbf{u}^\top\| = \|\tilde{\mathcal{H}}\mathbf{P}\mathbf{P}^\top \tilde{\mathcal{H}}^\top - \tilde{\mathcal{H}}\mathbf{P}\mathbf{P}^\top \tilde{\mathcal{H}}^\top\|$ goes to 0 as N grows. From (3.34) we have that $\|\mathbf{H} - \mathcal{H}\| \rightarrow 0$, as $N \rightarrow \infty$, and hence, $\|\tilde{\mathbf{H}} - \tilde{\mathcal{H}}\| \rightarrow 0$, as $N \rightarrow \infty$. Therefore, it can be seen that

$$\|\mathbf{U}\mathbf{U}^\top - \mathbf{u}\mathbf{u}^\top\| \rightarrow 0, \quad \text{as } N \rightarrow \infty, \quad (3.37)$$

with $\|\mathbf{U}\mathbf{U}^\top - \mathbf{u}\mathbf{u}^\top\|$ vanishing as N grows. The remainder of the derivation of the bound is analogous to that for (3.35).

Chapter 4

Signal interpolation of diffused sparse signals

We continue with GSP setups where signals are corrupted by perturbations, looking at a scenario where the (network aggregated) data has missing (limited) values. A natural and systematic approach in these setups is to model the observed values as a *sampled* signal, and then, propose an *interpolation* algorithm that reconstructs the original signal. Key to the success of the interpolation is to exploit the structure of both the graph signals and the assumed sampling scheme employed to represent the missing values. This is precisely the strategy put forth in our work in [66], which is also the main subject of this chapter.

The outline of the chapter is as follows. After briefly highlighting our contributions and remembering some concepts fundamental for this work in Section 4.1, Section 4.2 presents the main results, including the recovery schemes with known and unknown seeds, as well as unknown diffusing filter. The effect of noise, the design of the sampling matrix, and the consideration of more than one sampling node are also briefly analyzed, and next, a gamut of illustrative numerical results, including some showcasing practical relevance in real datasets, are presented in Section 4.3.

4.1 Introduction

In this chapter, we are concerned with the recovery of perturbed graph signals with a non-negligible proportion of missing values. Furthermore, we assume that the observations are gathered using an AGSS scheme, so that the (sampling and reconstruction) results in [23, 49] can be leveraged. In addition, because dealing with a large number of missing values (or, equivalently, having access to only a few sampled values) is a non-trivial problem, we assume that the original (unperturbed) signals studied in this framework are DSGS. In this sense, recall that DSGS are a class of signals that can be modeled as a sparse graph signal, i.e., a signal that is zero everywhere except in a few seeding nodes, which is then diffused through the network via a GF. First, we aim at reconstructing the signal assuming that the seeding nodes are known, but ultimately, we consider that the seeds are also unknown and our goal consists in *reconstructing both* the unperturbed signal and the seeds from the available observations. It is worth noticing that the AGSS is a sampling method for graph signals introduced in [23] where nodes successively aggregate the values of the signal in their neighborhood. Under this setting, the recovery can be guaranteed even if observations are gathered at a single node and the sampling collection process can be implemented distributively.

Contributions. The main contribution of this chapter is the generalization of the AGSS, which was originally proposed for BGS, to DSGS. Additionally, we generalize existing results for support identification and blind deconvolution to setups where observations are collected using an AGSS. The algorithms presented in this chapter are relevant also for distributed estimation and source localization. Sampling and recovery using as input the signal value at a subset of nodes were discussed in [21, 22, 47] for BGS, and in [95] for DSGS. Aggregation and space-shift sampling (a generalization of the AGSS considering multiple nodes) for BGS were investigated in [23]. Blind deconvolution and filter identification for DSGS in a centralized setup with access to the full signal (no sampling) were investigated in [16].

4.1.1 Successively aggregated graph signals

Fundamental to the approach put forth in this chapter is the AGSS, which was introduced in Section 2.4, and the definition of successively aggregated graph signals \mathbf{z}_i . Here, we briefly refresh these concepts and highlight their local nature.

Consider the signal $\mathbf{y} = \mathbf{H}\mathbf{x}$, which is the diffusion of \mathbf{x} across the graph \mathcal{G} as modeled by the GF \mathbf{H} with coefficients \mathbf{h} , and let us define the r -th shifted signal $\mathbf{z}^{(r)} := \mathbf{S}^r \mathbf{x}$ and further define the $N \times N$ matrix

$$\mathbf{Z} := [\mathbf{z}^{(0)}, \mathbf{z}^{(1)}, \dots, \mathbf{z}^{(N-1)}] = [\mathbf{x}, \mathbf{S}\mathbf{x}, \dots, \mathbf{S}^{N-1}\mathbf{x}], \quad (4.1)$$

Intuitively, \mathbf{Z} groups the signal \mathbf{x} and the result of the first $N - 1$ applications of the GSO. It is then clear that: a) the output of the graph filter can be found as $\mathbf{y} = \mathbf{H}\mathbf{x} = \mathbf{Z}\mathbf{h}$, with \mathbf{h} being zero-padded if $R < N$; b) since \mathbf{S} is a local operator, the r -th column of \mathbf{Z} can be found locally from the $(r - 1)$ -th diffusion step as $\mathbf{z}^{(r)} = \mathbf{S}\mathbf{z}^{(r-1)}$; and c) as one moves right-wise in (4.1), the columns of \mathbf{Z} can be viewed as the evolution of a process which is diffused linearly according to the local structure codified in \mathbf{S} .

Then, in the AGSS we consider the i -th node fixed and sample the signals observed at this node as the GSO \mathbf{S} is applied recursively. In other words, as the signal has been locally diffused according to \mathbf{S} , as described in (4.1). Then, with \mathbf{e}_i denoting the i -th canonical vector and leveraging the definition of the matrix \mathbf{Z} , the successively aggregated signal observed by node i is the i -th row of \mathbf{Z} , that is $\mathbf{z}_i := (\mathbf{e}_i^T \mathbf{Z})^T = \mathbf{Z}^T \mathbf{e}_i$. Under this setting, sampling is reduced to the selection of Q out of the N elements of \mathbf{z}_i , which can be done as follows

$$\mathbf{z}_{Q,i} := \mathbf{\Pi}_Q \mathbf{z}_i = \mathbf{\Pi}_Q (\mathbf{Z}^T \mathbf{e}_i). \quad (4.2)$$

4.2 Aggregation Sampling of DSGS

This section considers the problem of reconstructing DSGS from *local* observations obtained from AGSS. In contrast with BGS, the interest when dealing with DSGS can be either in recovering \mathbf{x} (signal reconstruction) or in recovering \mathbf{s} (distributed source localization and estimation). For that reason, we start by analyzing the case where $\mathbf{H} = \mathbf{I}$ and $\mathbf{x} = \mathbf{s}$. After that, we discuss the more general case where the nodes sample the signal $\mathbf{x} = \mathbf{H}\mathbf{s}$, both for known and unknown \mathbf{H} . A collaborative setting where more than one node collects the samples closes the section. To help readability, a summary of the setups considered is provided in Table 4.1.

Scenario	Sparse support	\mathbf{H}	$\mathbf{\Pi}_Q$	Obs. matrix	Equation
Sparse recovery	Known	Known $\mathbf{H} = \mathbf{I}$	Fixed	Θ	(4.6)
Active sampling	Known	Known $\mathbf{H} = \mathbf{I}$	Flexible	Θ	(4.8)
Blind sparse recovery	Unknown	Known $\mathbf{H} = \mathbf{I}$	Fixed	Θ	(4.9)
Diffused recovery	Known	Known	Fixed	Ξ	(4.6), replacing Θ with Ξ
Diffused active recovery	Known	Known	Flexible	Ξ	(4.8), replacing Θ with Ξ
Blind diffused recovery	Unknown	Known	Fixed	Ξ	(4.9), replacing Θ with Ξ
Blind deconvolution	Unknown	Unknown	Fixed	Φ	(4.12)

Table 4.1: Compilation of the settings considered in Section 4.2.

4.2.1 Aggregating the sparse input

A critical aspect to analyze the recovery $\mathbf{x} = \mathbf{s}$ from its aggregated samples is to write the relationship between the sampled signal \mathbf{z}_Q and the sparse input \mathbf{s} . For the ease of exposition, we do that in the form of a lemma.

Lemma 4.1. *Given the GSO $\mathbf{S} \in \mathbb{R}^{N \times N}$, the sampling matrix $\mathbf{\Pi}_Q$ and a sparse input $\mathbf{x} = \mathbf{s}$, the shifted signal \mathbf{z}_i and its sampled version $\mathbf{z}_{Q,i}$ can be expressed as*

$$\mathbf{z}_{Q,i} = \mathbf{\Pi}_Q \mathbf{z}_i \text{ and } \mathbf{z}_i = \mathbf{\Psi}^T \text{diag}(\mathbf{v}_i) \mathbf{U} \mathbf{s} := \Theta_i \mathbf{s}. \quad (4.3)$$

For the expression above, note that we have defined the observation matrix $\Theta_i := \mathbf{\Psi}^T \text{diag}(\mathbf{v}_i) \mathbf{U}$, where we recall that $\mathbf{\Psi}$ is the GFT for filters, $\mathbf{U} := \mathbf{V}^{-1}$ is the GFT for signals, and $\mathbf{v}_i = [V_{i,1}, \dots, V_{i,N}]^T$. While nontrivial, (4.3) can be derived after substituting (2.14) into (2.7), or by a minor modification of the proof in [23, Lemma 2]. Interestingly, (4.3) reveals that $\mathbf{z}_{Q,i}$ depends on how strongly the seeds express each of the frequencies (represented by $\mathbf{U} \mathbf{s}$), how strongly the sampling node senses each of the frequencies (represented by the frequency pattern \mathbf{v}_i) and the spectral effect of the diffusion (powers of the GSO) captured in the Vandermode matrix $\mathbf{\Psi}$.

To proceed with the recovery of the sparse input, let us denote as \mathcal{S} the support of \mathbf{s} , define $\mathbf{s}_{\mathcal{S}} := \mathbf{\Pi}_{\mathcal{S}} \mathbf{s} \in \mathbb{R}^{\mathcal{S}}$ as the vector collecting the non-zero values of \mathbf{s} , and suppose for now that this support is known. Then, we have that

$$\mathbf{z}_{Q,i} = \mathbf{\Pi}_Q \Theta_i \mathbf{s} = \mathbf{\Pi}_Q \Theta_i \mathbf{\Pi}_{\mathcal{S}}^T \mathbf{s}_{\mathcal{S}}. \quad (4.4)$$

This setup would correspond to the case where the indexes of the seeding nodes (identity of the influencers or location of the sources) is known, but their particular values are not. Clearly, for the recovery problem in (4.4) being identifiable, a necessary condition is Q , the number of observations in $\mathbf{z}_{Q,i}$, being no less than \mathcal{S} , the number of unknowns in $\mathbf{s}_{\mathcal{S}}$. Consider first the extreme case $Q = \mathcal{S}$. It is then clear that the sparse signal $\hat{\mathbf{s}}^{(i)}$, with the superscript (i) denoting the index of the sampling node, can be recovered as

$$\hat{\mathbf{s}}^{(i)} = \mathbf{\Pi}_{\mathcal{S}}^T \hat{\mathbf{s}}_{\mathcal{S}}^{(i)} \text{ and } \hat{\mathbf{s}}_{\mathcal{S}}^{(i)} = (\mathbf{\Pi}_Q \Theta_i \mathbf{\Pi}_{\mathcal{S}}^T)^{-1} \mathbf{z}_{Q,i}, \quad (4.5)$$

provided that the inverse exists, which will depend on the particular set of rows \mathcal{Q} and columns \mathcal{S} . For the standard case of $Q > S$ and the observations being corrupted by additive noise, the *observed* signal \mathbf{y}_i is given by $\mathbf{y}_i = \mathbf{z}_i + \mathbf{w}_i$. Assuming that the noise \mathbf{w}_i is zero-mean, independent of \mathbf{x} , and with known covariance matrix $\mathbf{R}_w^{(i)} := \mathbb{E}[\mathbf{w}_i \mathbf{w}_i^H]$, the best linear unbiased estimator of the sparse inputs is [96]

$$\hat{\mathbf{s}}^{(i)} = \mathbf{\Pi}_S^T \hat{\mathbf{s}}_S^{(i)} \quad \text{and} \quad \hat{\mathbf{s}}_S^{(i)} = (\mathbf{M}_Q)^\dagger (\mathbf{R}_{w,Q}^{(i)})^{-1/2} \mathbf{y}_{\mathcal{Q},i}, \quad \text{with} \quad (4.6)$$

$$\mathbf{R}_{w,Q}^{(i)} := \mathbf{\Pi}_Q \mathbf{R}_w^{(i)} \mathbf{\Pi}_Q^T \quad \text{and} \quad \mathbf{M}_Q := (\mathbf{R}_{w,Q}^{(i)})^{-1/2} \mathbf{\Pi}_Q \mathbf{\Theta}_i \mathbf{\Pi}_S^T,$$

provided that $\mathbf{R}_{w,Q}^{(i)}$ is non-singular. Note that if the noise is Gaussian, the estimator in (4.6) attains the Cramér-Rao bound.

Using (4.6), the error covariance matrix is [96]

$$\mathbf{R}_e^{(i)} := \mathbb{E}[(\mathbf{s}_S - \hat{\mathbf{s}}_S^{(i)})(\mathbf{s}_S - \hat{\mathbf{s}}_S^{(i)})^T] = (\mathbf{M}_Q^T \mathbf{M}_Q)^\dagger, \quad (4.7)$$

which depends on the noise model, the spectrum of the GSO, the seed nodes, the node taking the observations, and the sample-selection scheme adopted. (4.7) can then be used to assess the performance of the estimation. The particular error metric depends on the application at hand [97]. The most commonly used is the mean square error (MSE), which corresponds to minimizing $\text{trace}(\mathbf{R}_e^{(i)})$, with other popular metrics including the spectral norm $\lambda_{\max}(\mathbf{R}_e^{(i)})$ and the log determinant $\log \det(\mathbf{R}_e^{(i)})$.

Active sampling

Critical for the error performance is the design of a good sampling matrix. This requires solving first the optimal scheme for a fixed node i (which is a relevant problem by itself) and then selecting the best node (provided that the system operating conditions yield such a possibility). Except for trivial cases, optimizing the sampling set is NP-hard. In particular, when the interest is in the MSE, the optimal sampling matrix $\mathbf{\Pi}_Q^{(i)*}$ corresponds to

$$\begin{aligned} \min_{\mathbf{\Pi}_Q} \quad & \text{trace} \left((\mathbf{\Pi}_S \mathbf{\Theta}_i^T \mathbf{\Pi}_Q^T (\mathbf{\Pi}_Q \mathbf{R}_w^{(i)} \mathbf{\Pi}_Q^T)^{-1} \mathbf{\Pi}_Q \mathbf{\Theta}_i \mathbf{\Pi}_S^T)^\dagger \right) \\ \text{s.t.} \quad & \Pi_{Q,ij} \in \{0, 1\}, \quad \sum_j \Pi_{Q,ij} = 1, \quad \|\mathbf{\Pi}_Q\|_0 = Q, \end{aligned} \quad (4.8)$$

which is a fractional high-order polynomial minimization over binary variables. While convex relaxations to approximate (4.8) are available, greedy schemes for related problems have been shown to work well [98], and are advocated here.

Blind (sparse) recovery

In many relevant applications (e.g., those dealing with inverse problems) the seeds in \mathcal{S} are unknown. In that case, the noiseless recovery requires solving

$$\mathbf{s}^* := \arg \min_{\mathbf{s}} \|\mathbf{s}\|_0 \quad \text{s.t.} \quad \mathbf{z}_{\mathcal{Q},i} = \mathbf{\Pi}_Q \mathbf{\Theta}_i \mathbf{s}, \quad (4.9)$$

and then setting $\hat{\mathbf{x}}^{(i)} = \hat{\mathbf{s}}^{(i)} = \mathbf{s}^*$. Leveraging results from sparse recovery, it can be shown that identifiability needs $Q \geq 2S$ and the observation matrix Θ_i to be full spark [99]. The problem in (4.9), which does not account for noise, is NP-hard due to the presence of the ℓ_0 norm. A standard approach is to relax the equality with a LS cost and relax the ℓ_0 norm with a (weighted) ℓ_1 regularizer. This yields

$$\mathbf{s}_1^{(i)} := \arg \min_{\mathbf{s}} \|(\mathbf{R}_{w,Q}^{(i)})^{-1/2}(\mathbf{y}_{Q,i} - \mathbf{\Pi}_Q \Theta_i \mathbf{s})\|_2^2 + \gamma \|\mathbf{s}\|_1,$$

where $(\mathbf{R}_{w,Q}^{(i)})^{-1/2}$ accounts for the colored noise and γ is the regularization parameter.

4.2.2 Aggregating the diffused sparse input

We now analyze the recovery of $\mathbf{x} = \mathbf{H}\mathbf{s}$ from its aggregated samples, assuming first that the filter \mathbf{H} is known. The main difference with respect to the previous case is that now the relation between \mathbf{s} and $\mathbf{z}_{Q,i}$ is

$$\mathbf{z}_{Q,i} = \mathbf{\Pi}_Q \mathbf{z}_i \quad \text{and} \quad \mathbf{z}_i = \mathbf{\Psi}^T \text{diag}(\mathbf{v}_i \circ \tilde{\mathbf{h}}) \mathbf{U} \mathbf{s} := \mathbf{\Xi}_i \mathbf{s}, \quad (4.10)$$

where $\mathbf{\Xi}_i := \mathbf{\Psi}^T \text{diag}(\mathbf{v}_i \circ \tilde{\mathbf{h}}) \mathbf{U}$ and \circ denotes the entry-wise product. Compared with (4.3), we notice that the observations depend not only on the frequency pattern of the sampling node \mathbf{v}_i , but also on the frequency response of the diffusing filter $\tilde{\mathbf{h}}$. Intuitively, nodes with a frequency pattern more aligned with that of the diffusing filter (so that $|\det(\text{diag}(\mathbf{v}_i \circ \tilde{\mathbf{h}}))|$ is large) are more likely to give rise to a better reconstruction in the presence of noise.

Apart from replacing the observation matrix Θ_i with $\mathbf{\Xi}_i$, another important difference stems from the particular error to minimize. Since in this case \mathbf{x} and \mathbf{s} are different, depending on the application the focus can be on estimating \mathbf{s} and minimizing the MSE associated with $\mathbf{R}_{e,s}^{(i)} := \mathbb{E}[(\mathbf{s}_S - \hat{\mathbf{s}}_S^{(i)})(\mathbf{s}_S - \hat{\mathbf{s}}_S^{(i)})^T]$ or on estimating \mathbf{x} and minimizing the MSE associated with $\mathbf{R}_{e,x}^{(i)} := \mathbb{E}[(\mathbf{x} - \hat{\mathbf{x}}^{(i)})(\mathbf{x} - \hat{\mathbf{x}}^{(i)})^T]$. This requires to premultiply (or not) the error terms in the objectives of the optimizations presented in the previous sections. Such a dichotomy was not an issue for BGS since the frequency coefficients are a byproduct and the ultimate goal was to recover \mathbf{x} . By contrast, for DSGS both \mathbf{x} (signal reconstruction) and \mathbf{s} (source localization) are meaningful on their own.

4.2.3 Blind deconvolution

There may be scenarios where the diffusing filter $\mathbf{H} = \sum_{r=0}^{R-1} h_r \mathbf{S}^r$ is unknown. The recovery problem in this case is considerably more challenging, but it can be tackled provided that R , the order of the diffusing filter, is sufficiently small. To be specific, after some manipulations, the expression $\mathbf{x} = \mathbf{H}\mathbf{s}$ with \mathbf{H} being a graph filter can be written as $\mathbf{x} = \mathbf{V}(\mathbf{\Psi}^T \odot \mathbf{U}^T)^T \text{vec}(\mathbf{s}\mathbf{h}^T)$, where \odot stands for the Khatri-Rao product (cf. [16]). We can then relate the samples $\mathbf{z}_{Q,i}$ with the unknown \mathbf{s} and \mathbf{h} as

$$\mathbf{z}_{Q,i} = \mathbf{\Pi}_Q \mathbf{\Psi}^T \text{diag}(\mathbf{v}_i) (\mathbf{\Psi}^T \odot \mathbf{U}^T)^T \text{vec}(\mathbf{s}\mathbf{h}^T), \quad (4.11)$$

which is a system of Q *bilinear* equations. If the support \mathcal{S} is known, the number of unknowns is $R + S$. If it is not, the number is $R + N$ and the constraint $\|\mathbf{s}\|_0 \leq S$ must be added, further complicating the problem. The resultant problem can be handled by an alternating scheme that iterates between optimizing \mathbf{s} given \mathbf{h} and optimizing \mathbf{h} for the new \mathbf{s} . More sophisticated

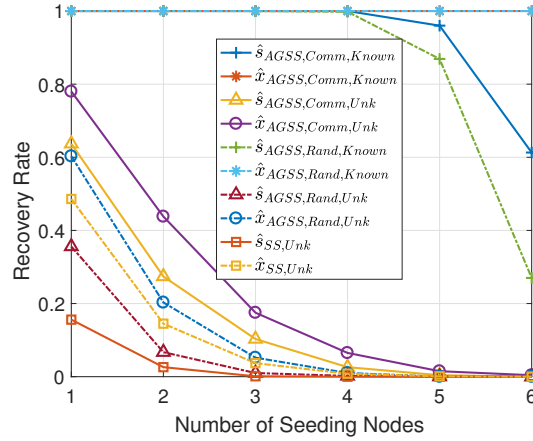


Figure 4.1: Recovery rate of DSGS in SBM graphs. Signals are recovered via the ℓ_1 -norm relaxation using the Laplacian as the GSO. 500 simulations with different graphs: $N = 50, B = 5, Q = 8, \mathbf{R}_{w,Q}^{(i)} = 10^{-5}\mathbf{I}$.

approaches include lifting techniques that define the lifted variable $\Sigma := \mathbf{s}\mathbf{h}^T$, the observation matrix $\Phi_i := \Psi^T \text{diag}(\mathbf{v}_i)(\Psi^T \odot \mathbf{U}^T)^T$, and find an approximation to

$$\min_{\Sigma} \|\mathbf{z}_{Q,i} - \Pi_Q \Phi_i \text{vec}(\Sigma)\|_2^2 + \gamma_1 \text{rank}(\Sigma) + \gamma_2 \|\Sigma\|_{2,0}, \quad (4.12)$$

where γ_1 and γ_2 are regularization parameters and $\|\Sigma\|_{2,0}$ is defined as the number of non-zero rows of Σ . The estimates $\hat{\mathbf{s}}^{(i)}$ and $\hat{\mathbf{h}}^{(i)}$ are then found the main left and right singular vectors of Σ^* , which are subject to an inherent scaling ambiguity. See [16] for further justification and suitable relaxations.

In the case of selection sampling (SS), the expression analogous to (4.11) is

$$\mathbf{x}_Q = \Pi_Q \mathbf{V}(\Psi^T \odot \mathbf{U}^T)^T \text{vec}(\mathbf{s}\mathbf{h}^T), \quad (4.13)$$

which is also a bilinear system similar to the previous one. Note that, as in the BGS case (cf. Section 2.3), the role of \mathbf{V} in SS is taken by $\Psi^T \text{diag}(\mathbf{v}_i)$ in AGSS.

4.2.4 Space-shift sampling of diffused sparse signals

In many setups, access to more than one sampling node is available. This is useful to robustify the recovery and reduce the number of required samples per node, which is convenient because the conditioning number of the Vandermonde matrix Ψ (one of the factors in Θ_i) worsens as the samples per node increase. The resultant sampling scheme is referred to as space-shift sampling [23]. To particularize it to the setup at hand, define the vectorized version of \mathbf{Z} as $\bar{\mathbf{z}}$ and then the $N^2 \times N$ matrix $\tilde{\mathbf{Y}} := [\text{diag}(\mathbf{v}_1), \dots, \text{diag}(\mathbf{v}_N)]^T \text{diag}(\tilde{\mathbf{h}})$. With these conventions, $\bar{\mathbf{z}}$ can be written as $\bar{\mathbf{z}} = (\mathbf{I} \otimes \Psi^T) \tilde{\mathbf{Y}} \mathbf{U} \mathbf{s}$, where \otimes stands for the Kronecker product. The sampled version in this case is given as $\bar{\mathbf{z}}_Q = \bar{\Pi}_Q \bar{\mathbf{z}}$, where $\bar{\Pi}_Q$ is a selection matrix of size $Q \times N^2$. The results in the previous sections can be applied to this case as well provided that Θ_i and Π_Q are replaced with $\bar{\Theta} := (\mathbf{I} \otimes \Psi^T) \tilde{\mathbf{Y}} \mathbf{U}$ and $\bar{\Pi}_Q$.

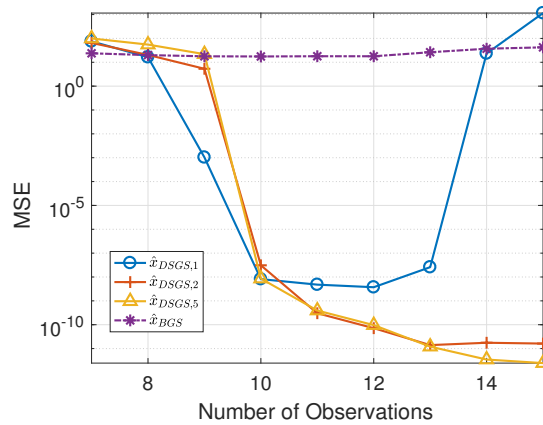


Figure 4.2: Median MSE of recovered signals defined over 95 real-world graphs using a blind diffused recovery scheme.

4.3 Numerical experiments

Short simulations to illustrate and gain intuition about some of the results presented are shown here.

Test case 1. First, we consider a SBM graph with N nodes and B communities with $N_b = N/B$ nodes each [100]. Edges exist with probability $p_{bb} = 0.4$ if the incident nodes are in the same community and with probability $p_{bb'} = 0.1$ if they are not. The remaining parameters are given in the caption of Fig. 4.1. The index of the seeding nodes is chosen uniformly at random and the seed value is drawn from a zero-mean unit variance Gaussian (ZMUVG). The filter taps have length $R = 6$ and each of them is drawn from a ZMUVG. Fig. 4.1 depicts the recovery rate, defined as the proportion of simulations for which the seeds are correctly identified and the ℓ_2 -norm of the error is less than 0.1, as the number of seeds S increases. All sampling nodes are considered, and the median error is reported. The 10 scenarios (lines) in the figure consider if: 1) $\mathbf{\Pi}_S$ is known or not (“Known”/“Unk”); 2) the sampling node i is in the same community than one of the seeds or is a random node (“Comm”/“Rand”); 3) the sampled signal is either \mathbf{s} or \mathbf{x} (“ $\hat{\mathbf{s}}$ ”/“ $\hat{\mathbf{x}}$ ”); and 4) the sampling scheme is AGSS or SS (“AGSS”/“SS”). The results confirm that recovery is harder as S increases, that blind schemes are not able to recover the signal if $S > Q/2 = 4$, and that knowledge of $\mathbf{\Pi}_S$ notably facilitates the recovery. We also observe that if the other two criteria are fixed, AGSS always outperform SS, confirming that AGSSs are more robust and less sensitive to the sampling configuration [23]. Similarly, “Comm” is always better than “Rand”. This is not surprising since the (diffused) seed values reach the sampling node faster if the node belongs to the same community. This also explains why recovering \mathbf{x} seems to be always easier than recovering (non-diffused) signal \mathbf{s} .

Test case 2. Fig. 4.2 tests our schemes in the D&D protein structure database [101], where nodes account for amino acids, links capture similarity, and signals are the expression level of the amino acids. We assume that the data can be accurately modeled as DSGS and try to recover the full signal following the blind diffused recovery scheme (label “DSGS,1”), its space-shift counterpart with 2 and 5 nodes (“DSGS,2”, “DSGS,5”), and AGSS modeling the data not as DSGS but as bandlimited (“BGS”). The median error of all graphs is reported and all sampling nodes are considered, selecting the 25th error percentile. The main observations are: 1) BGS yields the worst performance, pointing out that the DSGS model is a good fit for the information

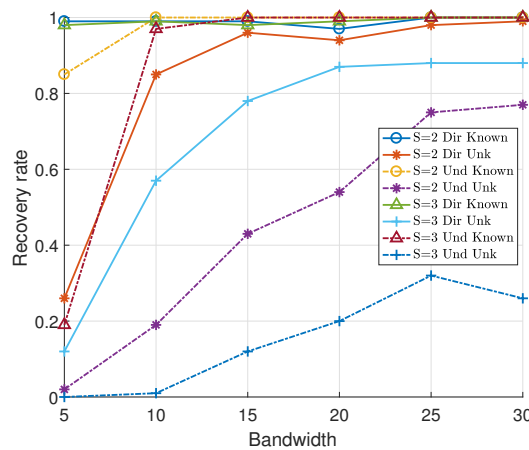


Figure 4.3: Recovery rate of DSGS in directed and undirected SBM graphs for varying filter bandwidth length. Signals are recovered via the pseudoinverse (known Π_S) and the ℓ_1 -norm relaxation (unknown Π_S) using the adjacency matrix as the GSO. Directed graphs with non-diagonalizable adjacency matrices are discarded. 100 graph-realizations for each type of graph, selecting the sampling node i from all the N nodes as the one leading to the smallest ℓ_2 -norm of $(s - \hat{s})$. The remaining parameters are: $N = 30, B = 2, p_b = 0.25, p_{bb'} = 0.05, Q = 4, \mathbf{R}_{w,Q}^{(i)} = 10^{-5}\mathbf{I}$.

in the D&D database; and 2) for the “DSGS,1” the median MSE increases when the number of observations is high. This stems from the conditioning number of Ψ as explained in Section 4.2.4. In contrast, the “DSGS,2” and “DSGS,5” schemes are more robust.

Test case 3. We test our schemes in the ETEX dataset [5], which contains $\{\mathbf{y}_t\}_{t=0}^{29}$ graph signals whose nodes correspond to different locations and t represents time. We use as GSO the adjacency of the geographical graph [102], the seed is set as $\mathbf{s} = \mathbf{y}_0$, and the signal to be sampled and recovered is $\mathbf{x} = \mathbf{y}_t$ for all $t > 0$. Using $Q = 16$ samples and the same approach than in the second test case, we run the experiment for 29 different signals (one per $t, t > 0$), obtaining MSE of $3 \cdot 10^{-5}$ and $1.5 \cdot 10^{-5}$ for “DSGS,1” and “DSGS,2” respectively.

Test case 4. In Fig. 4.3 we analyze the impact on the recovery of two factors: a) the bandwidth of the diffusing filter and b) the directivity of the supporting graph. To this end, let us consider a bandpass filter $\tilde{\mathbf{h}}$ whose non-zero band consists of W elements randomly drawn from a ZMUVG. Moreover, we consider two types of SBM graphs: one where links are directed (denoted as “Dir” in the figure) and another one with undirected links (“Und”). For this test case, the adjacency matrix is chosen as GSO. The remaining parameters are detailed in the caption of Fig. 4.3. The plotted results reveal that successful interpolation from DSGS samples is more amenable in directed than undirected graphs. The additional information about the edge direction contained in the GSO, central during both filtering and the AGSS, helps identifying the seeds in \mathcal{S} . Furthermore, lower W hinders the diffusion of the seeds, making the recovery harder. Indeed, in the extreme case of $W = 0$ the factor $\text{diag}(\mathbf{v}_i \circ \tilde{\mathbf{h}})$ in (4.10) renders the observations zero.

Test case 5. Fig. 4.4 studies the impact of the density of the graph on the recoverability of the signals. In this experiment, both the intra-cluster and the inter-cluster probability vary in the same direction, as explained in the caption of the figure. The results show that the recovery rate tends to improve when the graphs are denser. With a higher link probability, the chances that any node is close to the seeds increases, so that they can access the information related with the non-zero elements of the sampled signal. As a result, the fraction of nodes able to recover the signal increases. In addition, the plot confirms that directed graphs (“Dir”) have a better recovery

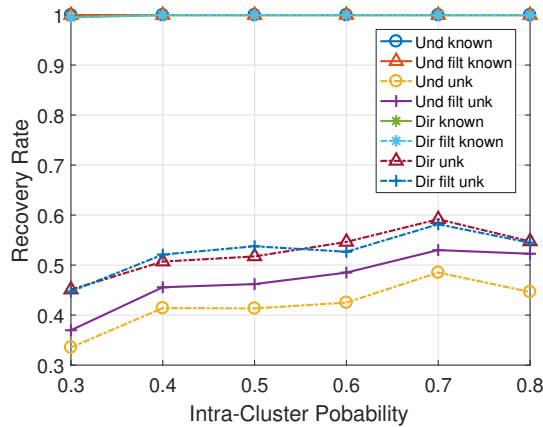


Figure 4.4: Recovery rate of sparse and DSGS in directed and undirected SBM graphs for different probabilities of inter-cluster and intra-cluster links. Signals are recovered via the pseudoinverse (known $\mathbf{\Pi}_S$) and the ℓ_1 -norm relaxation (unknown $\mathbf{\Pi}_S$) using the adjacency matrix as the GSO. The tested intra-cluster probabilities are $[0.3, 0.4, 0.5, 0.6, 0.7, 0.8]$, as shown in the horizontal axis. The corresponding inter-cluster probabilities are $[0.1, 0.15, 0.2, 0.25, 0.3]$. For each point in the figure, 100 graph-realizations are considered and, for each of those realizations, the $N = 30$ nodes are tested, so that the rates shown correspond to averages across 3000 trials. The remaining parameters are: $B = 3, Q = 9, \mathbf{R}_{w,Q}^{(i)} = 10^{-6}\mathbf{I}$.

rate than undirected graphs (“Und”), which is consistent with the results presented for the test case 3.

4.4 Conclusion

Here, we considered the presence of missing values in the (aggregated) observed data and approached the reconstruction of the original signal in the context of sampling and interpolation of graph signals. Assuming that the observed values were gathered through an AGSS, first, we proposed a convex optimization problem to interpolate sparse signals with either known or unknown support of the seeding nodes. Later on, we moved on to the more general case where the signals were DSGS and contemplated the signal interpolation when the diffusing filter \mathbf{H} was known, and then, the blind sparse recovery case where \mathbf{H} was unknown. Finally, we studied the case where the aggregated observations were collected at more than one sampling node, and we evaluated the proposed interpolation algorithms over synthetic and real-world datasets.

Chapter 5

Robust graph filter identification

In this chapter, we shift our attention from perturbations involving the observed graph signals to perturbations involving the edges of the observed graph, a prevalent type of uncertainty that is especially relevant when the networks are inferred from a set of nodal observations. Nonetheless, we may also encounter imperfections in the topology when networks are physical entities due to errors in the observation process. Regardless of their source, the presence of errors in the observed topology is critical for most GSP applications, so it is essential that they are properly accounted for in order to build a robust GSP framework. In this sense, we note that the polynomial definition of GFs renders them particularly sensitive to the presence of these perturbations, since even errors affecting only a few edges can lead to large discrepancies when high-order polynomials are involved. As a result, this chapter picks up our work in [67, 68] and addresses the relevant problem of GF identification from input-output pairs from a robust perspective. Even though we also consider the presence of noise in the observed graph signals, the proposed analysis is primarily concerned with perturbations involving the edges of the graph. Next, we provide some highlights about the method presented in this chapter and a brief summary of the resulting contributions.

5.1 Introduction

On top of its theoretical interest, the task of GF identification is practically relevant to, e.g., understanding the dynamics of network diffusion processes [9, 16, 17], as well as explaining the structure of real-world datasets [66, 103, 104]. Motivated by this, the work described in this chapter investigates the problem of estimating a GF from input-output signal pairs assuming that both the signals and the supporting graph have errors. The proposed approach is formulated in the vertex domain, avoiding the numerical instability of computing large polynomials and, at the same time, bypassing the challenges associated with robust *spectral* graph theory. To that end, we recast the robust estimation as a joint optimization problem where the GF identification objective is augmented with a graph-denoising regularizer, so that, on top of the desired GF, we also obtain an enhanced estimate of the supporting graph. The joint formulation leads to a non-convex bi-convex optimization problem, for which a provably-convergent efficient (alternating minimization) algorithm able to find an approximate solution is developed. Furthermore, to address scenarios where multiple GFs are present (e.g., when dealing with vector autoregressive (AR) spatio-temporal processes or in setups where nodes collect multi-feature vector measurements), we generalize our

framework so that multiple GFs, all defined over the same graph, are jointly identified.

Despite their theoretical and practical relevance, the number of robust GSP works is limited, due in part to the challenges emanating from the presence of graph perturbations. Initial works modeling the influence of perturbation in the spectrum of the graph Laplacian [52], and proposing a graphon-based perturbation model [54] were previously commented on in Section 2.7. More recently, [55] combines SEM with TLS to jointly infer the GF and the perturbations when the observed data is explained by a SEM. A different robust alternative is presented in [56], where the support of the graph is assumed to be known and the goal is to estimate the weights of the network topology and the coefficients of the GF. The resultant problem is non-convex and the authors adopt a sequential convex programming (SCP) approach to solve it. Finally, the presence of perturbations has also been considered in non-linear GSP tasks. An alternative definition of GFs robust to perturbations is proposed in [69], and the transferability of GFs when employed in graph neural networks is studied in [105–107].

Contributions and outline. After analyzing the influence of edge perturbations in polynomial GFs and stating the robust GF identification problem in Section 5.2, our main contributions are:

1. We formulate a non-convex optimization problem to jointly estimate the graph and the GF, develop an alternating optimization algorithm to solve it, and prove its convergence to a stationary point (Section 5.3).
2. We consider a generalization where several GFs are jointly estimated by exploiting the fact that they are polynomials of the same GSO (Section 5.4).
3. We propose an efficient implementation of the GF identification algorithm to handle graphs with a large number of nodes (Section 5.5).

The effectiveness of the proposed algorithms is evaluated numerically in Section 5.6, and some concluding remarks are provided in Section 5.7. Last but not least, while we focus on GF identification from input-output pairs, the approach put forth in this chapter can be generalized to other GSP tasks, which is a research path we plan to pursue in the near future.

5.2 GF identification with imperfect graph knowledge

This section introduces and discusses the problem of estimating a GF $\mathbf{H} = \sum_{r=0}^{N-1} h_r \mathbf{S}^r$ from noisy input-output signal pairs ($\mathbf{X} \in \mathbb{R}^{N \times M}$, $\mathbf{Y} \in \mathbb{R}^{N \times M}$) assuming that we have access to an *imperfect* GSO $\bar{\mathbf{S}} \in \mathbb{R}^{N \times N}$, which can be modeled as

$$\bar{\mathbf{S}} = \mathbf{S} + \mathbf{\Delta}, \quad (5.1)$$

where $\mathbf{S} \in \mathbb{R}^{N \times N}$ represents the true GSO and $\mathbf{\Delta} \in \mathbb{R}^{N \times N}$ is a *perturbation matrix*. Before discussing models for the perturbation matrix, we find illustrative to demonstrate the impact of $\mathbf{\Delta}$ on the GSP problem at hand.

As pointed out in the introduction, the presence of uncertainties in the topology of \mathcal{G} is particularly relevant when dealing with GFs. Indeed, due to the polynomial definition of \mathbf{H} , even small perturbations can lead to significant errors when $\bar{\mathbf{S}}$ (and not \mathbf{S}) is used as the true GSO. To see this more clearly, Fig. 5.1 provides an example that illustrates how the errors encoded in

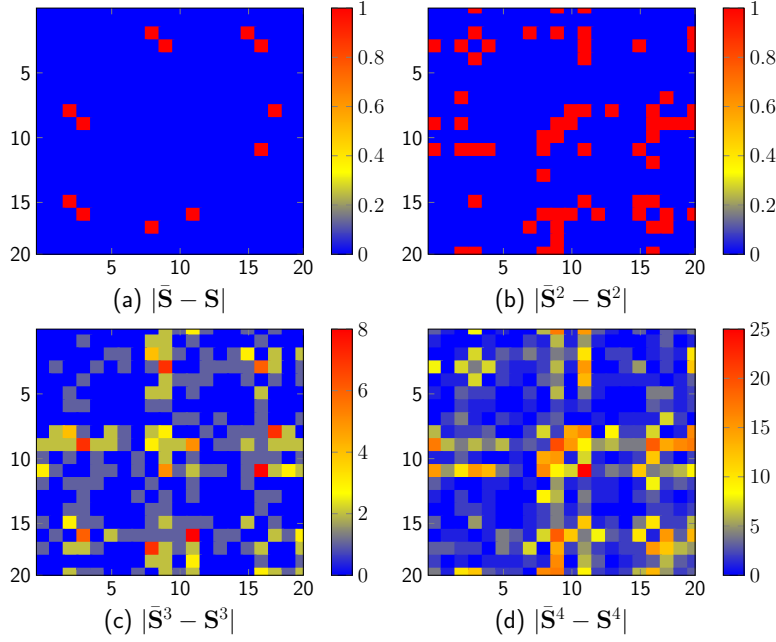


Figure 5.1: Absolute error for different powers of the matrix \mathbf{S} and its perturbed version $\bar{\mathbf{S}}$. The true GSO is the adjacency matrix of an Erdős-Rényi graph with link probability of 0.15, and $\bar{\mathbf{S}}$ is perturbed by creating and destroying links independently with a probability of 0.05.

Δ propagate for different matrix powers, demonstrating that the discrepancies between $\bar{\mathbf{S}}^r$ and \mathbf{S}^r increase swiftly as the power r grows. More rigorously, let C be a positive constant such that $\|\mathbf{S}\| \leq C$ and $\|\bar{\mathbf{S}}\| \leq C$, and define $\bar{\mathbf{H}} := \sum_{r=0}^{N-1} h_r \bar{\mathbf{S}}^r$. Then, the error generated by the perturbations is upper-bounded by

$$\|\bar{\mathbf{H}} - \mathbf{H}\| \leq \sum_{r=1}^{N-1} |h_r| \|\bar{\mathbf{S}}^r - \mathbf{S}^r\| \leq \sum_{r=1}^{N-1} |h_r| r C^{r-1} \|\Delta\|, \quad (5.2)$$

where the last inequality follows from [105, Lemma 3]. In words, the maximum difference between the true \mathbf{H} and the perturbed $\bar{\mathbf{H}}$ increases *exponentially* with the degree of the GF.

From the previous discussion, it is not surprising that the imperfect knowledge of the graph topology is also relevant when estimating the filter coefficients. In fact, ignoring the errors in Δ and attempting to estimate \mathbf{h} solving (2.4) when $\bar{\mathbf{S}}$ is used in lieu of the true (unknown) \mathbf{S} leads to a poor solution, as we illustrate numerically in Section 5.6. Motivated by this, we approach the GF identification problem from a robust perspective by taking into account the imperfect knowledge of the GSO. The resultant robust estimation task is formally stated next.

Problem 5.1. Let \mathcal{G} be a graph with N nodes, let $\mathbf{S} \in \mathbb{R}^{N \times N}$ be the true (unknown) GSO associated with \mathcal{G} , and let $\bar{\mathbf{S}} \in \mathbb{R}^{N \times N}$ be the perturbed (observed) GSO. Moreover, let $\mathbf{X} \in \mathbb{R}^{N \times M}$ and $\mathbf{Y} \in \mathbb{R}^{N \times M}$ be a pair of matrices collecting M observed input and output signals defined over \mathcal{G} and related by the model in (2.2). Our goal is to use the triplet $(\mathbf{X}, \mathbf{Y}, \bar{\mathbf{S}})$ to: i) learn the GF \mathbf{H} that best fits the model in (2.2) and ii) recover an enhanced estimation of \mathbf{S} . To that end, we make the following assumptions:

(AS1) \mathbf{H} is a polynomial of \mathbf{S} [cf. (2.1)].

(AS2) \mathbf{S} and $\bar{\mathbf{S}}$ are close according to some metric $d(\mathbf{S}, \bar{\mathbf{S}})$, i.e., the observed perturbations are “small” in some sense.

On top of the previous two assumptions, we also consider that the norm of the noise observation

matrix \mathbf{W} in (2.2) is small, which is a workhorse assumption in this type of problems. Similar to standard GF identification approaches, **(AS1)** limits the degrees of freedom of the linear operator in (2.2). However, the fact of the true \mathbf{S} being unknown adds uncertainty to the problem and, as a result, additional signal observations are required to achieve an identification performance comparable to the one obtained when $\bar{\mathbf{S}} = \mathbf{S}$. Regarding the recovery of the true GSO, **(AS2)** accounts for the hypothesis that $\bar{\mathbf{S}}$ is a perturbed observation of \mathbf{S} and, hence, matrices \mathbf{S} and $\bar{\mathbf{S}}$ are not extremely different. Note that this guarantees that “some” information about the true GSO is available, so that **(AS1)** can be effectively leveraged. While not exploited in our formulation, additional assumptions constraining the GSO could also be incorporated into the problem. Finally, the metric $d(\cdot, \cdot)$ employed to quantify the similarity between \mathbf{S} and $\bar{\mathbf{S}}$ should depend on the model for the perturbation Δ , a subject that is briefly discussed next.

5.2.1 Modeling graph perturbations

The development and analysis of graph perturbation models that combine practical relevance and analytical tractability constitutes an interesting yet challenging open line of research [53, 54]. Due to its flexibility and tractability, here we consider an additive perturbation model [cf. (5.1)], so that the focus is constrained to understanding the structural (statistical) properties of matrix $\Delta = \bar{\mathbf{S}} - \mathbf{S}$.

Consider first the case where perturbations only *create or destroy links* independently. If \mathcal{G} is an *unweighted graph*, a simple approach is to consider perturbations modeled as independent Bernoulli variables with possibly different creation/destruction probabilities. In this case, the entries of Δ would be

$$\Delta_{ij} = \begin{cases} 1 & \text{if link } (i, j) \text{ is created,} \\ -1 & \text{if link } (i, j) \text{ is destroyed,} \\ 0 & \text{otherwise.} \end{cases} \quad (5.3)$$

Since Δ models the creation and destruction of links, it is worth noting that $\Delta_{ij} = 1$ only if $S_{ij} = 0$ and $\Delta_{ij} = -1$ only if $S_{ij} = 1$. In the more general case of \mathcal{G} being a *weighted graph*, $\Delta_{ij} = -S_{ij}$ destroys an existing link while $\Delta_{ij} = z$ creates a new link. Here, z is a random variable sampled from a particular distribution (typically mimicking the weight distribution of the true \mathbf{S}). When facing this type of perturbations, a suitable distance function is the ℓ_0 norm

$$d(\mathbf{S}, \bar{\mathbf{S}}) = \|\mathbf{S} - \bar{\mathbf{S}}\|_0, \quad (5.4)$$

with the ℓ_1 norm $\|\mathbf{S} - \bar{\mathbf{S}}\|_1$ being a prudent convex relaxation.

Alternatively, rather than creating or destroying links, perturbations may represent uncertainty over the edge weights. This entails the support of matrix Δ matching that of \mathbf{S} and $\bar{\mathbf{S}}$, and the non-zero entries of Δ being sampled from a distribution that models the observation noise. For example, if the noise is zero-mean, Gaussian and white, it holds that $\Delta_{ij} \sim \mathcal{N}(0, \sigma^2)$ when $\mathbf{S}_{ij} \neq 0$ and $\Delta_{ij} = 0$ when $\mathbf{S}_{ij} = 0$. Under this setting, an appropriate distance metric is given by

$$d(\mathbf{S}, \bar{\mathbf{S}}) = \|\mathbf{S}_{\mathcal{E}} - \bar{\mathbf{S}}_{\mathcal{E}}\|_2^2, \quad (5.5)$$

where $\mathbf{S}_{\mathcal{E}}$ and $\bar{\mathbf{S}}_{\mathcal{E}}$ only select the non-zero entries (edges) in \mathbf{S} and $\bar{\mathbf{S}}$. Additionally, one can have setups where the two types of perturbations are simultaneously present. That is, perturbations may create and destroy links while the actual value of the existing links is also uncertain. In such a case, a combination of ℓ_1 and ℓ_2 norms like in elastic nets [108] is adequate.

The models previously described only consider the perturbation of edges in an independent fashion. However, there may be scenarios where the perturbations are correlated. Consider for example a communication network. If the power supply of a node stalls, the signal-to-noise ratio of all its links will be poor, and hence, links involving that node will be more likely to fail. Perturbations dependent across links can be modeled by means of a multivariate correlated Bernoulli distribution, an Ising model, or more sophisticated random graph models [109]. When prior information about the dependence of the perturbations is available, it can be incorporated into the function $d(\mathbf{S}, \bar{\mathbf{S}})$ to better extract the information encoded in $\bar{\mathbf{S}}$.

5.3 Robust GF identification

This section presents the optimization problem and the proposed algorithm to estimate \mathbf{H} and \mathbf{S} under the setting described in Problem 5.1. Given the matrices \mathbf{X} , \mathbf{Y} , and $\bar{\mathbf{S}}$, we approach the robust GF identification task by means of the following non-convex optimization

$$\begin{aligned} \hat{\mathbf{H}}, \hat{\mathbf{S}} = \underset{\mathbf{H}, \mathbf{S}}{\operatorname{argmin}} \quad & \|\mathbf{Y} - \mathbf{H}\mathbf{X}\|_F^2 + \lambda d(\mathbf{S}, \bar{\mathbf{S}}) + \beta \|\mathbf{S}\|_0 \\ \text{s. to : } \quad & \mathbf{S} \in \mathcal{S}, \quad \mathbf{S}\mathbf{H} = \mathbf{H}\mathbf{S}, \end{aligned} \quad (5.6)$$

where s. to stands for subject to. The first term in the objective promotes the linear input-output relation in (2.2), encouraging the norm of $\mathbf{W} = \mathbf{Y} - \mathbf{H}\mathbf{X}$ to be small. The use of the Frobenius norm is well-justified when the observation noise is Gaussian and white, but other types of noise could be accommodated by using a different norm. The second term incorporates the assumption **(AS2)** as a regularizer to obtain an estimate $\hat{\mathbf{S}}$ that is related to the given GSO $\bar{\mathbf{S}}$. The ℓ_0 norm in the third term accounts for the fact of \mathbf{S} being sparse. Clearly, if additional information about \mathbf{S} is available, it can be incorporated into (5.6), either as a regularizer (e.g., a statistical prior quantifying the log-likelihood of a class of GSOs) or as a constraint that *must* be satisfied (e.g., the GSO being symmetric). The latter is indeed the role of $\mathbf{S} \in \mathcal{S}$ in (2.2), with \mathcal{S} representing a (desired) family of GSOs such as the set of adjacency matrices with no self-loops (\mathcal{S} is the set of matrices with non-negative entries whose diagonal entries are zero) or the set of combinatorial graph Laplacians (matrices with non-positive off-diagonal entries and zero row-sum). Finally, the (key) constraint $\mathbf{S}\mathbf{H} = \mathbf{H}\mathbf{S}$ captures the fact of \mathbf{H} being a polynomial of \mathbf{S} and not of $\bar{\mathbf{S}}$ **(AS1)**. Note first that the constraint is pertinent, if \mathbf{H} is a polynomial of \mathbf{S} , then \mathbf{H} and \mathbf{S} have the same eigenvectors and, as a result, their product commutes [17]. More importantly for the GF-identification at hand, when the GSO is perfectly known the model $\mathbf{H} = h_0\mathbf{I} + h_1\mathbf{S} + \dots + h_{N-1}\mathbf{S}^{N-1}$ is linear in the unknown \mathbf{h} . As a result, a formulation that estimates \mathbf{h} directly (as carried out in classical non-robust approaches) is well-motivated. However, when both \mathbf{h} and \mathbf{S} are unknown, the model $\mathbf{H} = h_0\mathbf{I} + h_1\mathbf{S} + \dots + h_{N-1}\mathbf{S}^{N-1}$ is highly non-linear in \mathbf{S} , challenging the development of a tractable solution that jointly estimates \mathbf{h} and \mathbf{S} . Our formulation bypasses this problem by recasting the optimization variables as \mathbf{H} and \mathbf{S} , leading to the (more tractable) bilinear constraint in (5.6). Nonetheless, if estimating \mathbf{h} is the ultimate goal, this can be readily achieved from $\hat{\mathbf{H}}$ and $\hat{\mathbf{S}}$ as

$$\hat{\mathbf{h}} = \left(\operatorname{vec}(\mathbf{I}), \operatorname{vec}(\hat{\mathbf{S}}), \dots, \operatorname{vec}(\hat{\mathbf{S}}^{N-1}) \right)^\dagger \operatorname{vec}(\hat{\mathbf{H}}). \quad (5.7)$$

The approach put forth in (5.6) has two main advantages. First, while most works formulate the recovery of the GF in the spectral domain, our formulation operates in the vertex domain. Working on the spectral domain would imply finding the Vandermonde GFT matrix Ψ . Since this matrix involves high-order polynomials of the eigenvalues of the GSO, it is also prone to numerical instability and error accumulation [9]. Even if approaches that bypass this issue by

estimating the graph-frequency response $\tilde{\mathbf{h}} = \Psi \mathbf{h}$ in lieu of \mathbf{h} are adopted, the estimation would still be challenging since they require computing the eigenvectors \mathbf{V} , which are known to be highly sensitive to errors in the GSO (especially those associated with small eigenvalues) [52,110]. On top of this, characterizing the spectral errors and incorporating those to the optimization is not a trivial task. The second advantage emanates from casting the true GSO \mathbf{S} as an explicit optimization variable. As already explained, this approach is robust to error accumulation and facilitates the incorporation of the (additive) effect of the perturbations into the optimization. An additional benefit is that we obtain a denoised version (enhanced estimation) of the true GSO, which can be practically relevant in most real-world applications.

In a nutshell, in the context of robust GF identification, choosing a formulation that: i) works entirely in the vertex domain, ii) considers \mathbf{S} as an explicit optimization variable, and iii) codifies the GF structure via the constraint $\mathbf{HS} = \mathbf{SH}$, exhibits multiple advantages. However, it must be noted that the number of optimization variables is larger than in classical approaches (adding computational complexity) and that the bilinear filtering constraint $\mathbf{HS} = \mathbf{SH}$, while more tractable than its polynomial counterpart, is still non-convex. Alternatives to deal with these issues are discussed in later sections.

5.3.1 Alternating minimization for robust GF identification

This section presents a systematic efficient approach to find an approximate solution to (5.6). Since the goal is to design specific algorithms, from this section onwards, we particularize the GSO distance to $d(\mathbf{S}, \tilde{\mathbf{S}}) = \|\mathbf{S} - \tilde{\mathbf{S}}\|_0$, so that, according to the discussion in Section 5.2.1, the focus is on graph perturbations that create and destroy links. Apart from its practical relevance, the reason for choosing the ℓ_0 norm as a distance is also motivated by its more intricate (challenging) structure. Indeed, the algorithms presented next can be easily adapted to (more tractable) distances associated with alternative perturbation models. Having clarified this, the main obstacle to solving (5.6) is its lack of convexity, which emanates from two different *sources*: (*s1*) the ℓ_0 norms in the objective, and (*s2*) the bilinear constraint involving \mathbf{S} and \mathbf{H} . Next, we explain the strategy adopted to deal with them and find a solution to (5.6) by solving a succession of convex problems.

- Regarding the ℓ_0 norm in (*s1*), a workhorse approach is to replace it with its convex surrogate, the ℓ_1 norm. However, it is possible to exploit more sophisticated (non-convex) alternatives that typically lead to sparser solutions. The one chosen in this work is to approximate the ℓ_0 norm of a generic matrix $\mathbf{Z} \in \mathbb{R}^{I \times J}$ using the logarithmic penalty

$$\|\mathbf{Z}\|_0 \approx r_\delta(\mathbf{Z}) := \sum_{i=1}^I \sum_{j=1}^J \log(|Z_{ij}| + \delta), \quad (5.8)$$

where δ is a small positive constant [111]. The non-convexity of the logarithm can be handled efficiently by relying on a majorization-minimization (MM) approach [112], which considers an iterative linear approximation leading to an iterative re-weighted ℓ_1 norm. It is worth noting that, since we will consider an iterative algorithm to deal with the bilinearity of (5.6), the iterative nature of the re-weighted ℓ_1 norm will not impose a significant computational burden. Details on the exact form of this sparse regularizer will be provided soon, when describing the estimation of \mathbf{S} .

- To deal with the bilinear terms in (*s2*), we adopt an alternating optimization approach [113] resulting in an iterative algorithm where the optimization variables \mathbf{H} and \mathbf{S} are updated in

two separate iterative steps. At each step, we optimize over one of the optimization variables with the other remaining fixed, resulting in two simpler problems that can be solved efficiently. The details about the specific steps will be provided shortly.

Taking into account these considerations, the first task to implement our approach is to rewrite the problem in (5.6) as

$$\min_{\mathbf{S} \in \mathcal{S}, \mathbf{H}} \|\mathbf{Y} - \mathbf{H}\mathbf{X}\|_F^2 + \lambda r_{\delta_1}(\mathbf{S} - \bar{\mathbf{S}}) + \beta r_{\delta_2}(\mathbf{S}) + \gamma \|\mathbf{S}\mathbf{H} - \mathbf{H}\mathbf{S}\|_F^2, \quad (5.9)$$

where we recall that $r_{\delta}(\cdot)$ was introduced in (5.8). Note that: i) the logarithmic penalty has also been used to promote sparsity in the term $\mathbf{S} - \bar{\mathbf{S}}$ since we selected the ℓ_0 norm as the distance between \mathbf{S} and $\bar{\mathbf{S}}$, and ii) the constraint $\mathbf{S}\mathbf{H} = \mathbf{H}\mathbf{S}$ was relaxed and rewritten as a regularizer, a formulation more amenable to an alternating optimization approach.

The next task is to solve (5.9) by means of an iterative algorithm that blends techniques from alternating optimization and MM approaches. Specifically, for a maximum of t_{max} iterations, we run the following two steps at each iteration $t = 0, \dots, t_{max} - 1$.

Step 1: GF Identification. We estimate the block of N^2 variables collected in \mathbf{H} while the current estimate of the GSO, denoted as $\mathbf{S}^{(t)}$, remains fixed. This results in the convex optimization problem

$$\mathbf{H}^{(t+1)} = \arg \min_{\mathbf{H}} \|\mathbf{Y} - \mathbf{H}\mathbf{X}\|_F^2 + \gamma \|\mathbf{S}^{(t)}\mathbf{H} - \mathbf{H}\mathbf{S}^{(t)}\|_F^2, \quad (5.10)$$

an LS minimization whose closed-form solution is

$$\text{vec}(\mathbf{H}^{(t+1)}) = (\mathbf{X}\mathbf{X}^\top \otimes \mathbf{I} + \gamma(\mathbf{S}\mathbf{S}^\top \oplus \mathbf{S}^\top \mathbf{S} - \mathbf{S}^\top \otimes \mathbf{S} - \mathbf{S} \otimes \mathbf{S}))^{-1} (\mathbf{X} \otimes \mathbf{I}) \text{vec}(\mathbf{Y}). \quad (5.11)$$

Here, \otimes is the Kronecker product, \oplus is the Kronecker sum, and \mathbf{I} is the identity matrix of size $N \times N$. Also note that (5.11) omitted the iteration superscript in $\mathbf{S}^{(t)}$ to alleviate notation.

Step 2: Graph Denoising. Following an MM scheme, we optimize an upper bound of (5.9) where the logarithmic penalties are linearized. Then, we estimate the block of N^2 variables collected in \mathbf{S} while the current estimate of the GF $\mathbf{H}^{(t+1)}$ remains fixed. This yields

$$\mathbf{S}^{(t+1)} = \arg \min_{\mathbf{S} \in \mathcal{S}} \sum_{i=1}^N \sum_{j=1}^N (\lambda \bar{\Omega}_{ij}^{(t)} |S_{ij} - \bar{S}_{ij}| + \beta \Omega_{ij}^{(t)} |S_{ij}|) + \gamma \|\mathbf{S}\mathbf{H}^{(t+1)} - \mathbf{H}^{(t+1)}\mathbf{S}\|_F^2, \quad (5.12)$$

where $\bar{\Omega}^{(t)}$ and $\Omega^{(t)}$ are computed in an entry-wise fashion based on the GSO estimate from the previous iteration as

$$\bar{\Omega}_{ij}^{(t)} = \frac{1}{|S_{ij}^{(t)} - \bar{S}_{ij}| + \delta_1}, \quad \Omega_{ij}^{(t)} = \frac{1}{|S_{ij}^{(t)}| + \delta_2}. \quad (5.13)$$

The overall alternating algorithm is summarized in Algorithm 2, where a fixed number of iterations is considered. The algorithm starts by initializing the GSO as $\mathbf{S}^{(0)} = \bar{\mathbf{S}}$ (although other options could also be appropriate), and then, it iterates between Steps 1 and 2 for a fixed number of epochs (or until some stopping criterion is met). In this regard, a key feature of the algorithm is that it is guaranteed to converge to a stationary point, as is formally stated next.

Algorithm 2: Robust GF identification with graph denoising.

Input: $\mathbf{X}, \mathbf{Y}, \bar{\mathbf{S}}$
Output: $\hat{\mathbf{H}}, \hat{\mathbf{S}}$.

- 1 Initialize $\mathbf{S}^{(0)}$ as $\mathbf{S}^{(0)} = \bar{\mathbf{S}}$.
- 2 **for** $t = 0$ **to** $t_{max} - 1$ **do**
- 3 Compute $\mathbf{H}^{(t+1)}$ by solving (5.11) fixing $\mathbf{S}^{(t)}$.
- 4 Update $\boldsymbol{\Omega}^{(t)}$ and $\bar{\boldsymbol{\Omega}}^{(t)}$ as in (5.13).
- 5 Compute $\mathbf{S}^{(t+1)}$ by solving (5.12) using $\mathbf{H}^{(t+1)}$, $\boldsymbol{\Omega}^{(t)}$, and $\bar{\boldsymbol{\Omega}}^{(t)}$.
- 6 **end**
- 7 $\hat{\mathbf{H}} = \mathbf{H}^{(t_{max})}$, $\hat{\mathbf{S}} = \mathbf{S}^{(t_{max})}$.

Theorem 5.1. Denote as $f(\mathbf{H}, \mathbf{S})$ the objective function in (5.9), and let \mathcal{Z}^* be the set of stationary points of f . Let $\mathbf{z}^{(t)} = [\text{vec}(\mathbf{H}^{(t)})^\top, \text{vec}(\mathbf{S}^{(t)})^\top]^\top$ represent the solution provided by the iterative algorithm (5.11)-(5.12) after t iterations. Assuming that i) the GSO does not have repeated eigenvalues and ii) every row of $\tilde{\mathbf{X}} = \mathbf{V}^{-1}\mathbf{X}$ has at least one nonzero entry, then $\mathbf{z}^{(t)}$ converges to a stationary point of f as t goes to infinity, i.e.,

$$\lim_{t \rightarrow \infty} d(\mathbf{z}^{(t)} | \mathcal{Z}^*) = 0,$$

with $d(\mathbf{z} | \mathcal{Z}^*) := \min_{\mathbf{z}^* \in \mathcal{Z}^*} \|\mathbf{z} - \mathbf{z}^*\|_2$.

The proof relies on the convergence results shown in [114, Th. 1b] and the details are provided in Appendix 5.8. Note that the convergence of the algorithm was not self-evident since the original optimization problem in (5.9) is non-convex and Step 2 is minimizing an upper-bound of the original objective function. The sufficient conditions in i) and ii) guarantee that every graph frequency is excited so that the GF is identifiable and (5.10) has a unique solution, which is a requirement for convergence (see Proposition 5.1 in Appendix 5.8 for details). Clearly, condition ii) is fulfilled even for $M = 1$ if all the entries of the vector $\tilde{\mathbf{x}} = \mathbf{V}^{-1}\mathbf{x}$ are nonzero. Alternatively, when $M > 1$ and ii) is satisfied, condition i) can be relaxed.

Another relevant element in the proposed algorithm is the weight γ . If γ is set to a value that is too large, the GF estimated in the first iteration $\mathbf{H}^{(1)}$ will be an (almost exact) polynomial of $\bar{\mathbf{S}}$ so that the algorithm will converge quickly to the same solution as that of the non-robust design [cf. (5.6) with $\mathbf{S} = \bar{\mathbf{S}}$]. On the other hand, if γ is too close to zero the two problems decouple and the solution converges quickly to that of the two separated problems [cf. (5.10) and (5.12) with $\gamma = 0$]. As a result, the value of the parameter must be chosen carefully. In this context, schemes that start with a small γ to encourage the exploration during the warm-up phase, and then increase γ as the iteration index grows to guarantee that the final $\hat{\mathbf{H}}$ is a polynomial of $\hat{\mathbf{S}}$ are a suitable alternative for the setup at hand.

Finally, one drawback of the proposed robust GF identification algorithm is that the optimization problems in (5.10) and (5.12) may be slow when dealing with large graphs. However, we will mitigate this issue by introducing an efficient implementation that reduces the computational complexity of the overall algorithm (see Section 5.5).

5.3.2 Leveraging stationary observations

The alternating convex approximation in Algorithm 2 exploits the fact that \mathbf{X} and \mathbf{Y} are linearly related via \mathbf{H} , which is a polynomial of \mathbf{S} . However, in setups where the perturbations in $\bar{\mathbf{S}}$ are very large, obtaining accurate estimates of \mathbf{S} and \mathbf{h} from $\hat{\mathbf{H}}$ may still be challenging. One alternative to overcome this issue is to leverage the additional structure potentially present in our data. Indeed, as detailed in the introduction, it is common to consider setups where the signals exhibit additional properties depending on the supporting graph, with notable examples including graph-bandlimited signals [8, 13], diffused sparse graph signals [16, 66], or graph stationary signals [74, 115, 116]. Clearly, incorporating such additional information into the optimization problem would enhance its estimation performance.

This section explores this path, restricting our attention to the case where the observed signals are stationary on \mathcal{G} . The motivation for this decision is that, due to the tight connection between graph-stationary signals and GFs (see Chapter 2), the formulation in (5.9) and Algorithm 2 require relatively minor modifications to incorporate the assumption of \mathbf{X} and \mathbf{Y} being stationary on \mathbf{S} , leaving the incorporation of additional signal models as future work. To formulate the updated problem, recall that the covariance matrix of a stationary graph signal can be expressed as a polynomial of the GSO. Therefore, incorporating stationarity calls for modifying (5.9) as

$$\begin{aligned} \min_{\mathbf{S} \in \mathcal{S}, \mathbf{H}} \quad & \|\mathbf{Y} - \mathbf{H}\mathbf{X}\|_F^2 + \lambda r_{\delta_1}(\mathbf{S} - \bar{\mathbf{S}}) + \beta r_{\delta_2}(\mathbf{S}) + \gamma \|\mathbf{S}\mathbf{H} - \mathbf{H}\mathbf{S}\|_F^2 \\ \text{s. to:} \quad & \|\mathbf{C}_y \mathbf{S} - \mathbf{S} \mathbf{C}_y\|_F^2 \leq \epsilon_y, \|\mathbf{C}_x \mathbf{S} - \mathbf{S} \mathbf{C}_x\|_F^2 \leq \epsilon_x, \end{aligned} \quad (5.14)$$

where \mathbf{C}_y and \mathbf{C}_x denote the covariance matrices of \mathbf{Y} and \mathbf{X} , respectively. If the covariances are perfectly known, then the corresponding parameters ϵ_y and ϵ_x are set to zero. Alternatively, if the \mathbf{C}_y and \mathbf{C}_x are the sample estimates of the true covariances, then the values of ϵ_y and ϵ_x must be selected based on the quality of the estimators (accounting, e.g., for the number of available observations M).

The constraints in (5.14) capture the graph-stationarity assumption by promoting the commutativity with the true GSO. Therefore, such constraints are considered in the graph denoising step [cf. (5.12)]. In addition, since \mathbf{C}_y , \mathbf{C}_x and \mathbf{H} are all polynomials of \mathbf{S} , the equalities $\mathbf{C}_y \mathbf{H} = \mathbf{H} \mathbf{C}_y$ and $\mathbf{C}_x \mathbf{H} = \mathbf{H} \mathbf{C}_x$ must hold as well, so it is also possible to augment the GF identification step [cf. (5.10)] with the corresponding constraints. While in the interest of brevity, we do not spell out all the possible formulations here, the impact of several of these alternatives is numerically analyzed in Section 5.6. Finally, it is important to note that, since the stationarity constraints are quadratic and convex, the convergence described in Theorem 5.1 also holds true for the iterative algorithm associated with (5.14).

5.4 Joint robust identification of multiple GFs

In Section 5.3, we approached the problem of identifying a single GF \mathbf{H} defined over a single graph \mathcal{G} . However, in a variety of situations we encounter multiple processes (signals) over the same graph \mathcal{G} . Consider for example a network of weather stations measuring the temperature, humidity, and wind speed. Each of these measurements corresponds to observations of a different process, all of them taking place over a common graph. Intuitively, since all the GFs are related by the underlying graph \mathcal{G} , we propose a *joint* GF identification approach that exploits this relationship to enhance the quality of the estimation. We focus first on the case where the input-output signals associated with each GF (graph process) are observed separately. Later in the section, we address

a slightly more involved case where the GFs model the (AR) dynamics of a time-varying graph signal and, as a result, the observed signals are intertwined.

Consider a set of K unknown GFs $\{\mathbf{H}_k\}_{k=1}^K$, all represented by $N \times N$ matrices and defined over the graph \mathcal{G} . To be consistent with Problem 5.1, we assume that: i) the true \mathbf{S} is unknown and only the perturbed version $\bar{\mathbf{S}}$ is available; ii) all \mathbf{H}_k are polynomials of the *same* GSO \mathbf{S} ; and iii) for each k , matrices $\mathbf{X}_k \in \mathbb{R}^{N \times M_k}$ and $\mathbf{Y}_k \in \mathbb{R}^{N \times M_k}$ collect the observed input and output graph signals and are related via

$$\mathbf{Y}_k = \mathbf{H}_k \mathbf{X}_k + \mathbf{W}_k, \quad (5.15)$$

with $\mathbf{H}_k = \sum_{r=0}^{N-1} h_{r,k} \mathbf{S}^r$ and \mathbf{W}_k being a white random matrix capturing observation noise and model inaccuracies. Then, we aim at estimating the GFs $\{\mathbf{H}_k\}_{k=1}^K$ in a joint fashion while taking into account the inaccuracies in the topology of \mathcal{G} . This is summarized in the following problem statement.

Problem 5.2. *Let \mathcal{G} be a graph with N nodes, let $\mathbf{S} \in \mathbb{R}^{N \times N}$ be the true (unknown) GSO associated with \mathcal{G} , and let $\bar{\mathbf{S}} \in \mathbb{R}^{N \times N}$ be the perturbed (observed) GSO. Moreover, let $\mathbf{X}_k \in \mathbb{R}^{N \times M_k}$ and $\mathbf{Y}_k \in \mathbb{R}^{N \times M_k}$ be the matrices collecting the M_k observed input and output graph signals associated with $k = 1, \dots, K$ network processes, all defined over \mathcal{G} and adhering to the model in (5.15). Our goal is to use $\{\mathbf{X}_k\}_{k=1}^K$, $\{\mathbf{Y}_k\}_{k=1}^K$, and $\bar{\mathbf{S}}$ to learn the K GFs $\{\mathbf{H}_k\}_{k=1}^K$ that best fit the data, along with an enhanced estimation of \mathbf{S} . To that end, we make the following assumptions:*

(AS2) \mathbf{S} and $\bar{\mathbf{S}}$ are close according to some metric $d(\mathbf{S}, \bar{\mathbf{S}})$, i.e., the observed perturbations are "small" in some sense.

(AS3) Every \mathbf{H}_k is a polynomial of \mathbf{S} .

Assumption **(AS2)**, which was also considered in Problem 5.1, promotes the tractability of the problem by ensuring that \mathbf{S} and $\bar{\mathbf{S}}$ are related. As discussed in Section 5.2.1, the distance function $d(\cdot, \cdot)$ must be selected depending on the perturbation model at hand. **(AS3)** captures the key fact that all the matrices \mathbf{H}_k are GFs of the *same* GSO, establishing a link that can be leveraged via a joint estimation (optimization) of the K GFs. Implementing an approach similar to that in Section 5.3 (i.e., working on the vertex domain, considering the true GSO as an explicit optimization variable, accounting for the GF structure via a commutativity constraint, and assuming that the graph perturbations create and destroy links), the multi-filter counterpart to (5.9) that codifies Problem 5.2 is

$$\begin{aligned} \min_{\mathbf{S} \in \mathcal{S}, \{\mathbf{H}_k\}_{k=1}^K} & \sum_{k=1}^K \alpha_k \|\mathbf{Y}_k - \mathbf{H}_k \mathbf{X}_k\|_F^2 + \lambda r_{\delta_1} (\mathbf{S} - \bar{\mathbf{S}}) \\ & + \beta r_{\delta_2} (\mathbf{S}) + \sum_{k=1}^K \gamma \|\mathbf{S} \mathbf{H}_k - \mathbf{H}_k \mathbf{S}\|_F^2. \end{aligned} \quad (5.16)$$

Ideally, the value of the positive weight α_k must be selected based on the norm of \mathbf{W}_k (e.g., prior information on the noise level and the number of signal pairs M_k). If none is available, then $\alpha_k = 1$ for all k . Equally important, the fact of pursuing a joint optimization implies that each \mathbf{H}_k contributes with a regularization term $\|\mathbf{S} \mathbf{H}_k - \mathbf{H}_k \mathbf{S}\|_F^2$ promoting the commutativity of the k -th GF with the *single* \mathbf{S} . Intuitively, having the same \mathbf{S} in all these terms couples the optimization across k and contributes to reduce the uncertainty over \mathbf{S} , leading to enhanced estimates of both \mathbf{S} and $\{\mathbf{H}_k\}_{k=1}^K$. As a result, the joint GF identification approach is expected to provide better results than estimating each \mathbf{H}_k separately by solving K instances of (5.9). We validate this hypothesis numerically via the experiments in Section 5.6.

Following a motivation similar to that in the previous section, we deal with the non-convex minimization in (5.16) designing an alternating optimization algorithm that breaks the bilinear terms $\mathbf{S}\mathbf{H}_k$ and $\mathbf{H}_k\mathbf{S}$, and approximates the logarithmic terms with a linear upper-bound. The resulting algorithm solves iteratively the following two subproblems for $t = 1, \dots, t_{\max}$ iterations.

Step 1: Multiple GF Identification. Given the current estimate $\mathbf{S}^{(t)}$, we solve the optimization problem in (5.16) with respect to each $\mathbf{H}^{(k)}$. This yields

$$\mathbf{H}_k^{(t+1)} = \underset{\mathbf{H}_k}{\operatorname{argmin}} \alpha_k \|\mathbf{Y}_k - \mathbf{H}_k \mathbf{X}_k\|_F^2 + \gamma \|\mathbf{S}^{(t)} \mathbf{H}_k - \mathbf{H}_k \mathbf{S}^{(t)}\|_F^2, \quad (5.17)$$

whose closed-form solution can be found using (5.11) replacing γ with γ/α_k , \mathbf{X} with \mathbf{X}_k , and \mathbf{Y} with \mathbf{Y}_k . Note that since the only coupling across GFs is via the GSO, (5.17) estimates each $\mathbf{H}_k^{(t+1)}$ separately from the other GFs, solving K LS problems (each with N^2 unknowns). Furthermore, if multiple processors are available, (5.17) can be run in parallel across k .

Step 2: Graph Denoising. Given the current estimates of the GFs $\{\mathbf{H}_k^{(t+1)}\}_{k=1}^K$, we follow an MM scheme that, minimizing a linear upper-bound of the logarithmic penalties, yields the estimate of the GSO via

$$\begin{aligned} \mathbf{S}^{(t+1)} = \underset{\mathbf{S} \in \mathcal{S}}{\operatorname{argmin}} & \sum_{ij=1}^N (\lambda \bar{\Omega}_{ij}^{(t)} |S_{ij} - \bar{S}_{ij}| + \beta \Omega_{ij}^{(t)} |S_{ij}|) \\ & + \sum_{k=1}^K \gamma \|\mathbf{S} \mathbf{H}_k^{(t+1)} - \mathbf{H}_k^{(t+1)} \mathbf{S}\|_F^2, \end{aligned} \quad (5.18)$$

where Ω and $\bar{\Omega}$ are obtained as in (5.13).

The solution to Problem 5.2 is simply given by $\hat{\mathbf{S}} = \mathbf{S}^{(t_{\max})}$ and $\hat{\mathbf{H}}_k = \mathbf{H}_k^{(t_{\max})}$ for every k . Similar to (5.9), convergence to a stationary point of (5.16) is guaranteed, as formally stated next.

Corollary 5.1. *Denote as $f(\{\mathbf{H}_k\}_{k=1}^K, \mathbf{S})$ the objective function in (5.16). If the vector $\mathbf{z}^{(t)} = [\operatorname{vec}(\mathbf{H}_1^{(t)})^\top, \dots, \operatorname{vec}(\mathbf{H}_K^{(t)})^\top, \operatorname{vec}(\mathbf{S})^\top]^\top$ represents the solution provided by the iterative algorithm (5.17)-(5.18) after t iterations and every \mathbf{X}_k excites all graph frequencies, then $\mathbf{z}^{(t)}$ converges to a stationary point of f as the number of iterations t goes to infinity.*

The key to prove Theorem 5.1, which established the convergence to a stationary point for the robust estimation of a single GF, was to show that the optimization problem in (5.9) and the proposed algorithm satisfied the conditions in [114, Th. 1b]. The formulation we put forth for the multi-filter case resembles closely that of the single-filter case, and, as a result, it is not difficult to show that those conditions also hold true for the problem in (5.16) (see Appendix 5.8 for details).

The discussion and formulations in Section 5.3.2 dealing with incorporating additional information about the input-output signals into the optimization are also pertinent for the setup in this section. The details of such a formulation are omitted for brevity, but it will be explored in the experimental section.

5.4.1 Joint GF identification for time series

A slightly different, practically relevant, setup where multiple GFs need to be estimated is that of graph-based multivariate time series. In that setup, each variable is associated with a node of the

graph and the multiple graph-signal observations correspond to different instants of a time-varying graph signal. AR and moving-average (MA) modeling of time series has a long tradition, with common approaches to decrease the degrees of freedom including limiting the memory of the series and assuming that matrices of coefficients relating different time instants are low rank [117]. In the context of graph signals and network processes, a natural approach is to constrain the matrices of coefficients to be GFs, all defined over the same graph [118, 119]. This section introduces a variation of the problem in (5.16) tailored to this setup.

To introduce the multiple-graph identification problem formally, let \mathbf{X}_κ and \mathbf{Y}_κ denote a collection of M_κ graph signals corresponding to measurements of a network process for $\kappa = 1, \dots, \kappa_{max}$ time instants. Suppose now that \mathbf{Y}_κ can be accurately modeled by an AR dynamics with memory K so, at every instant κ , the observations \mathbf{Y}_κ satisfy the equation

$$\mathbf{Y}_\kappa = \sum_{k=1}^K \mathbf{H}_k \mathbf{Y}_{\kappa-k} + \mathbf{X}_\kappa, \text{ with } \mathbf{H}_k = \sum_{r=0}^{N-1} h_{r,k} \mathbf{S}^r, \quad (5.19)$$

where \mathbf{X}_κ is the exogenous input, and the GF \mathbf{H}_k models the influence that the signal observations from the time instant $\kappa - k$ exert on the (current) signal at time κ .

Suppose now that: i) we have access to an estimated (imperfect) graph $\bar{\mathbf{S}}$, ii) the value of the graph signals at different time instants is available, and iii) our goal is to estimate the set of matrices (GFs) $\{\mathbf{H}_k\}_{k=1}^K$ in (5.19) that describe the dynamics of the multivariate time series. This can be accomplished as

$$\begin{aligned} \min_{\mathbf{S} \in \mathcal{S}, \{\mathbf{H}_k\}_{k=1}^K} & \sum_{\kappa=K+1}^{\kappa_{max}} \left\| \mathbf{Y}_\kappa - \mathbf{X}_\kappa - \sum_{k=1}^K \mathbf{H}_k \mathbf{Y}_{\kappa-k} \right\|_F^2 \\ & + \lambda r_{\delta_1} (\mathbf{S} - \bar{\mathbf{S}}) + \beta r_{\delta_2} (\mathbf{S}) + \sum_{k=1}^K \gamma \|\mathbf{S} \mathbf{H}_k - \mathbf{H}_k \mathbf{S}\|_F^2. \end{aligned} \quad (5.20)$$

The main difference relative to (5.16) is in the first term, which accounts for the new observation model [cf. (5.15) vs. (5.19)]. Note that we assume that the exogenous input \mathbf{X}_κ is observed. If that were not the case, it would suffice to remove \mathbf{X}_κ from the objective (possibly updating the Frobenius norm in case statistical knowledge about \mathbf{X}_κ were available). Albeit the differences, the problem in (5.20) is closely related to (5.16), with the sources of non-convexities being the same. As a result, we approach its solution with a modified version of Algorithm 2 which, at each iteration t , runs two steps. In the first one, we estimate each of the K GFs by solving

$$\begin{aligned} \mathbf{H}_k^{(t+1)} = \operatorname{argmin}_{\mathbf{H}_k} & \sum_{\kappa=K+1}^{\kappa_{max}} \left\| \mathbf{Y}_\kappa - \mathbf{X}_\kappa - \mathbf{H}_k \mathbf{Y}_{\kappa-k} - \sum_{k' < k} \mathbf{H}_{k'}^{(t+1)} \mathbf{Y}_{\kappa-k'} \right. \\ & \left. - \sum_{K \geq k' > k} \mathbf{H}_{k'}^{(t)} \mathbf{Y}_{\kappa-k'} \right\|_F^2 + \sum_{k=1}^K \gamma \left\| \mathbf{S}^{(t)} \mathbf{H}_k - \mathbf{H}_k \mathbf{S}^{(t)} \right\|_F^2, \end{aligned} \quad (5.21)$$

which is different from the previous GF identification step [cf. (5.17)]. In contrast, the graph-denoising step in (5.18) remains the same. Note that (5.21) updates each GF separately in a cyclic way by solving an LS problem with N^2 unknowns. Alternative implementations include using $\mathbf{H}_{k'}^{(t)}$ in lieu of $\mathbf{H}_{k'}^{(t+1)}$ for all $k' < k$ (so that a parallel implementation is enabled) as well as considering a single LS problem with KN^2 unknowns.

Finally, it is worth emphasizing that the formulation introduced in this section can be used as a starting point to design more general robust schemes for multivariate time series defined over

a graph. Dealing with both AR and MA matrices, assuming that the memory of the system is not known, having only partial/statistical information on the exogenous input, and observing the signals at only a subset of nodes are all examples of setups of interest. Since our goal in this section was to demonstrate the relevance of a robust multiple GF formulation in the context of multivariate time series, to facilitate exposition we restricted our discussion to the relatively simple case in (5.19), but many other setups (including those previously listed) will be subject of our future work.

5.5 Efficient implementation of the robust GF identification algorithm

The algorithms proposed up to this point are able to find a solution to the robust GF identification problem in polynomial time. However, their computational complexity scales with the number of nodes as N^7 . To facilitate the deployment in setups where N is large, this section puts forth an efficient implementation that reduces the number of operations.

The new algorithm (summarized in Algorithm 3) preserves the core structure of Algorithm 2, with an outer loop that, at each iteration, runs two steps: one involving the estimation of the GF(s) and another one dealing with the estimation of the GSO. The main difference is that now, instead of finding the exact solution to those two problems, we obtain an approximate solution. While the details, which are step-dependent, will be specified in the next paragraphs, the overall idea is that for each of the steps we run a few simple (gradient/proximal) iterations. Although Algorithm 3 involves two nested loops, the complexity of the problems in the inner loop is cut down significantly, so that the overall computational overhead is reduced.

To be specific, we describe next the two steps that, at each iteration of the outer loop $t = 0, \dots, t_{max} - 1$, Algorithm 3 runs.

Step 1: Efficient GF Identification. Solving the GF-identification step with the closed-form solution presented in (5.11) involves inverting a matrix of size $N^2 \times N^2$, which requires $\mathcal{O}(N^6)$ operations. To explain our alternative implementation, let $f_1(\mathbf{H}|\mathbf{S}^{(t)})$ denote the objective function in (5.10). Since f_1 is strictly convex and smooth, it can be efficiently optimized using a gradient descent approach [120].

To that end, for each iteration t of the outer loop, we define the inner iteration index τ as well as the sequence of variables $\check{\mathbf{H}}^{(\tau)}$ with $\tau = 0, \dots, \tau_{max_1}$, which is initialized as $\check{\mathbf{H}}^{(0)} = \mathbf{H}^{(t)}$. With this notation at hand, at each iteration $\tau = 0, \dots, \tau_{max_1} - 1$ of the inner loop, we update $\check{\mathbf{H}}^{(\tau+1)}$ via

$$\check{\mathbf{H}}^{(\tau+1)} = \check{\mathbf{H}}^{(\tau)} - \mu \nabla f_1(\check{\mathbf{H}}^{(\tau)}|\mathbf{S}^{(t)}). \quad (5.22)$$

Here, $\mu > 0$ is the step size and ∇f_1 denotes the gradient of f_1 with respect to \mathbf{H} , which is given by

$$\nabla f_1(\mathbf{H}|\mathbf{S}^{(t)}) = 2(\mathbf{H}\mathbf{X}\mathbf{X}^\top - \mathbf{Y}\mathbf{X}^\top) + 2\gamma(\mathbf{S}^{(t)\top}(\mathbf{S}^{(t)}\mathbf{H} - \mathbf{H}\mathbf{S}^{(t)}) - (\mathbf{S}^{(t)}\mathbf{H} - \mathbf{H}\mathbf{S}^{(t)})\mathbf{S}^{(t)\top}). \quad (5.23)$$

When the τ_{max_1} gradient updates are computed, we conclude the GF-identification step by setting $\mathbf{H}^{(t+1)} = \check{\mathbf{H}}^{(\tau_{max_1})}$.

Since each gradient calculation involves the multiplication of $N \times N$ matrices, the resultant computational complexity is $\mathcal{O}(\tau_{max_1} N^3)$, which may go down to $\mathcal{O}(\tau_{max_1} N^{2.4})$ if an efficient multiplication algorithm is employed [121]. For large values of N , this complexity is substantially smaller than that required to find the inverse of an $N^2 \times N^2$ matrix.

Step 2: Efficient graph denoising. Since the optimization in (5.12) involves N^2 variables (the entries in \mathbf{S}), using an off-the-shelf convex solver incurs a computational complexity of $\mathcal{O}(N^7)$ [120]. Inspired by the Lasso regression algorithm [122], we optimize individually over each entry S_{ij} in an iterative manner. The main idea is running multiple rounds of N^2 efficient scalar optimizations rather than dealing with a single but demanding N^2 -dimensional problem. To provide the details of the scheme developed to estimate \mathbf{S} , we need to specify the set of constraints \mathcal{S} and introduce some definitions. Let us focus on the set of adjacency matrices $\mathcal{S}_{\mathcal{A}} := \{\mathbf{S} | S_{ij} \geq 0, S_{ii} = 0\}$ and define the vectors $\mathbf{s} := \text{vec}(\mathbf{S})$, vector $\bar{\mathbf{s}} := \text{vec}(\bar{\mathbf{S}})$, and the matrix $\boldsymbol{\Sigma}^{(t)} := \mathbf{H}^{(t+1)\top} \oplus -\mathbf{H}^{(t+1)}$. With these definitions in place, the minimization in (5.12) is equivalent to solving

$$\begin{aligned} \min_{\mathbf{s}} \quad & \sum_{i=1}^{N^2} \left(\lambda \bar{\omega}_i^{(t)} |s_i - \bar{s}_i| + \beta \omega_i^{(t)} s_i \right) + \gamma \|\boldsymbol{\Sigma}^{(t)} \mathbf{s}\|_2^2, \\ \text{s. to : } \quad & \mathbf{s} \geq 0, \quad \mathbf{s}_{\mathcal{D}} = 0, \end{aligned} \quad (5.24)$$

where $\mathbf{s}_{\mathcal{D}}$ collects the elements in the diagonal of \mathbf{S} , and the vectors $\bar{\boldsymbol{\omega}}^{(t)}$ and $\boldsymbol{\omega}^{(t)}$ are computed according to (5.13) but with $\bar{\mathbf{s}}^{(t)}$ and $\mathbf{s}^{(t)}$ in lieu of $\bar{\mathbf{S}}^{(t)}$ and $\mathbf{S}^{(t)}$. The constraint $\mathbf{s}_{\mathcal{D}} = 0$, implies that only the $N^2 - N$ elements of \mathbf{s} representing the off-diagonal entries of \mathbf{S} need to be optimized. The key point to find those $N^2 - N$ values is to leverage that the non-differentiable part of the cost in (5.24) is separable across s_i , postulate $N^2 - N$ scalar optimization problems (coupled via the ℓ_2 term in the cost), and address the optimization following a projected cyclic coordinate descent scheme.

To define clearly the operation of Step 2 at each iteration t of the outer loop, we need to introduce some notation. First, let us denote as τ the iteration index for the inner loop, define the sequence of variables $\check{s}^{(\tau)}$ where $\tau = 0, \dots, \tau_{max_2}$, and initialize the sequence as $\check{s}^{(0)} = \mathbf{s}^{(t)}$. Moreover, with $\ell \notin \mathcal{D}$ denoting an index of the off-diagonal elements of the GSO, let $\boldsymbol{\sigma}_\ell \in \mathbb{R}^{N^2}$ denote the associated ℓ -th column of $\boldsymbol{\Sigma}^{(t)}$, $\omega_\ell \geq 0$ and $\bar{\omega}_\ell \geq 0$ the associated entries of $\boldsymbol{\omega}^{(t)}$ and $\bar{\boldsymbol{\omega}}^{(t)}$, and $\check{s}_\ell^{(\tau)} \in \mathbb{R}$ the associated entry of $\check{s}^{(\tau)}$ (note that dependence on t was omitted to facilitate readability). Then, at every iteration $\tau = 0, \dots, \tau_{max_2} - 1$ of the inner loop, Algorithm 3 optimizes over each \check{s}_ℓ separately in a cyclic (successive) way. The advantage of this approach is that the solution to the *scalar* optimization over \check{s}_ℓ is given in closed form by the following projected soft-thresholding operation

$$\check{s}_\ell^{(\tau+1)} = \begin{cases} \left(-\bar{\lambda}_\ell + u_\ell^{(\tau)} \right)^+ & \text{if } \bar{s}_\ell < -\bar{\lambda}_\ell + u_\ell^{(\tau)}, \\ \left(\bar{\lambda}_\ell + u_\ell^{(\tau)} \right)^+ & \text{if } \bar{s}_\ell > \bar{\lambda}_\ell + u_\ell^{(\tau)}, \\ \bar{s}_\ell & \text{otherwise,} \end{cases} \quad (5.25)$$

$$\text{with } \bar{\lambda}_\ell = \frac{\lambda \bar{\omega}_\ell}{\gamma \boldsymbol{\sigma}_\ell^\top \boldsymbol{\sigma}_\ell} \quad \text{and} \quad u_\ell^{(\tau)} = \frac{-\beta \omega_\ell - \gamma \boldsymbol{\sigma}_\ell^\top \mathbf{r}_\ell^{(\tau)}}{\gamma \boldsymbol{\sigma}_\ell^\top \boldsymbol{\sigma}_\ell}.$$

Here, $(\cdot)^+$ denotes the operation $(x)^+ = \max(0, x)$, and

$$\mathbf{r}_\ell^{(\tau)} := \sum_{j < \ell} \boldsymbol{\sigma}_j \check{s}_j^{(\tau+1)} + \sum_{j > \ell} \boldsymbol{\sigma}_j \check{s}_j^{(\tau)}. \quad (5.26)$$

Note that (5.25) is a soft-thresholding operation with respect to the term $|s_i - \bar{s}_i|$. Also, the constraints in $\mathcal{S}_{\mathcal{A}}$ are satisfied due to the projection operator $(\cdot)^+ := \max\{\cdot, 0\}$, and because we do not optimize over the elements of the diagonal of \mathbf{S} .

At first sight, computing each \check{s}_ℓ requires roughly N^2 operations, so estimating the whole vector \mathbf{s} would entail a computational complexity of $\mathcal{O}(N^4)$. However, a closer inspection of the

Algorithm 3: Reduced-complexity robust GF identification.

Input: $\mathbf{X}, \mathbf{Y}, \bar{\mathbf{S}}$
Output: $\hat{\mathbf{H}}, \hat{\mathbf{S}}$.

- 1 Initialize $\mathbf{H}^{(0)}$ and $\mathbf{S}^{(0)}$
- 2 $\bar{\mathbf{s}} = \text{vec}(\bar{\mathbf{S}})$
- 3 **for** $t = 0$ **to** $t_{\max} - 1$ **do**
 - // GF-identification step
 - 4 $\check{\mathbf{H}}^{(0)} = \mathbf{H}^{(t)}$
 - 5 **for** $\tau = 0$ **to** $\tau_{\max_1} - 1$ **do**
 - 6 $\check{\mathbf{H}}^{(\tau+1)} = \check{\mathbf{H}}^{(\tau)} + \mu \nabla f_1(\check{\mathbf{H}}^{(\tau)} | \mathbf{S}^{(t)})$
 - 7 **end**
 - 8 $\mathbf{H}^{(t+1)} = \check{\mathbf{H}}^{(\tau_{\max_1})}$
 - // Graph denoising step
 - 9 $[\boldsymbol{\sigma}_1, \dots, \boldsymbol{\sigma}_{N^2}] = \mathbf{H}^{(t+1)\top} \oplus \mathbf{H}^{(t+1)}$
 - 10 $\check{\mathbf{s}}^{(0)} = \text{vec}(\mathbf{S}^{(t)})$
 - 11 Update $\bar{\boldsymbol{\omega}}^{(t)}, \boldsymbol{\omega}^{(t)}$ via (5.13) using $\bar{\mathbf{s}}$ and $\check{\mathbf{s}}^{(0)}$
 - 12 **for** $i = 0$ **to** $\tau_{\max_2} - 1$ **do**
 - 13 **for** $\ell \notin \mathcal{D}$ **do**
 - 14 Obtain $\mathbf{r}_\ell^{(\tau)}$ via (5.26)
 - 15 Obtain $\check{s}_\ell^{(\tau+1)}$ via (5.25) using $\boldsymbol{\sigma}_\ell, \mathbf{r}_\ell^{(\tau)}, \omega_\ell, \bar{\omega}_\ell$
 - 16 **end**
 - 17 **end**
 - 18 $\mathbf{S}^{(t+1)} = \text{unvec}(\check{\mathbf{s}}^{(\tau_{\max_2})})$
 - 19 **end**
 - 20 $\hat{\mathbf{H}} = \mathbf{H}^{(t_{\max})}, \hat{\mathbf{S}} = \mathbf{S}^{(t_{\max})}$.

vectors $\boldsymbol{\sigma}_\ell$ reveals that no more than $2N$ of their entries are non-zero because $\boldsymbol{\sigma}_\ell$ are the columns of the Kronecker sum of two $N \times N$ matrices. We exploit this sparsity and reduce the number of operations required to compute each s_ℓ to approximately $2N$, rendering the final computational complexity of the graph denoising step to $\mathcal{O}(2\tau_{\max_2}N^3)$.

The pseudocode describing the efficient implementation of Steps 1 and 2 is provided in Algorithm 3. The summary is as follows. We postulate a nested algorithm with two loops. The outer loop runs t_{\max} iterations. The inner loop runs two steps: Step 1, with τ_{\max_1} iterations, and Step 2, with τ_{\max_2} iterations. While the complexity for Algorithm 2 scaled as $\mathcal{O}(t_{\max}N^7)$, with t_{\max} being typically small, the overall computational complexity of Algorithm 3 is roughly $\mathcal{O}(t_{\max}(\tau_{\max_1} + \tau_{\max_2})N^3)$, which is encouraging, since $2N^2$ variables are optimized and it scales with N significantly better than Algorithm 2. Solving Steps 1 and 2 optimally requires setting large values for τ_{\max_1} and τ_{\max_2} . Nonetheless, we observe that in most tested setups the approach of setting small values for τ_{\max_1} and τ_{\max_2} (at the cost of setting a slightly higher value for t_{\max}) typically yields a faster convergence. Finally, implementations where the number of iterations is not fixed but selected based on some convergence criterion are also sensible alternatives.

We close the section noting that we developed Algorithm 3 for the setting described in Problem 5.1 because the notation was simpler and facilitated the discussion. Nonetheless, an analogous approach may be followed for the joint estimation of K GFs (cf. Section 5.4), resulting in an algorithm with complexity per GF similar to that for Algorithm 3.

5.6 Numerical results

This section discusses several numerical experiments to gain insights and assess the performance of the robust GF identification algorithms. Unless specified otherwise, for a variable of interest Θ , we report its normalized estimation error defined as

$$nerr(\hat{\Theta}, \Theta) := \frac{\|\hat{\Theta} - \Theta\|_F^2}{\|\Theta\|_F^2}, \quad (5.27)$$

where $\hat{\Theta}$ and Θ denote the estimated and the true value, respectively. The code implementing our algorithms and the experiments presented next is available on GitHub¹. The interested reader is referred there for additional details and tests.

5.6.1 Synthetic experiments

We start by evaluating our algorithms with synthetic data, which is key to gain intuition. Unless otherwise stated, graphs are sampled from ER random graph model with a link probability of $p = 0.2$ and $N = 20$ nodes; $\bar{\mathbf{S}}$ is obtained by randomly creating and destroying 10% of the links in \mathbf{S} ; $M = 50$ signals \mathbf{X} and \mathbf{Y} are generated according to (2.2), with the columns of \mathbf{X} being drawn from a multivariate Gaussian distribution $\mathcal{N}(\mathbf{0}, \mathbf{I})$, so the signals \mathbf{Y} are stationary on \mathbf{S} ; signals in \mathbf{Y} are corrupted with white Gaussian noise with a normalized power of $\eta_{\mathbf{W}} = 0.05$; and the reported error corresponds to the median of $nerr$ across 64 realizations of graphs and graph signals.

Test case 1. The first experiment evaluates the influence of perturbations as the order of the GF R increases. The number of observed pairs of signals considered is $M = 100$ and 10% of the edges in \mathbf{S} are perturbed. Results are reported in Fig. 5.2, where the x-axis shows R and the y-axis $nerr(\hat{\mathbf{h}}, \mathbf{h})$. The algorithms considered are: (i) the GF identification algorithm that ignores perturbations [see (2.4)], denoted as “FI”; (ii) the robust GF identification algorithm from Algorithm 2 (“RFI”); (iii) a variation of “RFI” where the reweighted ℓ_1 norm is replaced by the standard ℓ_1 norm (“RFI- ℓ_1 ”); and (iv) the robust GF identification algorithm accounting for the stationarity of \mathbf{Y} (“RFI-ST”). First, we observe that the error of the “FI” algorithm, while small for low values of R , increases rapidly as R grows. This is aligned with the discussion of high-order polynomials in Section 5.2 and illustrates the merits of the robust algorithms. Moreover, “RFI-ST” presents the best performance illustrating the importance of exploiting additional structure when it is available. Finally, comparing the error of “RFI” and “RFI- ℓ_1 ” showcases the benefits of replacing the ℓ_1 norm with its reweighted version.

Test case 2. The next experiment tests the influence of different types of perturbations in the robust and non-robust GF identification algorithms. Figs. 5.3a and 5.3b illustrate the error of the estimated GF $\hat{\mathbf{H}}$ and the denoised GSO $\hat{\mathbf{S}}$ as the ratio of perturbed links in $\bar{\mathbf{S}}$ increases. Graphs are sampled from the SW [86] random graph model and $\bar{\mathbf{S}}$ is obtained by creating new links, destroying existing links, or simultaneously creating and destroying links, which are respectively denoted as “C”, “D”, and “C/D” in the legend. Since the non-robust “FI” algorithm does not perform graph denoising we show the error $nerr(\bar{\mathbf{S}}, \mathbf{S})$, denoted as “ $\bar{\mathbf{S}}$ ” in Fig. 5.3b. Furthermore, because the number of perturbed links is fixed, the error of $\bar{\mathbf{S}}$ is the same for the considered perturbations and it is only plotted once. From the figures, we observe that destroying links is the most harmful

¹https://github.com/reysam93/graph_denoising

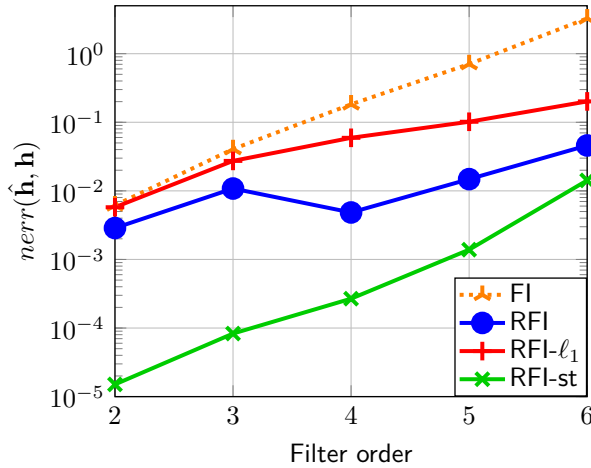


Figure 5.2: Comparison of the error when estimating $\hat{\mathbf{h}}$ via robust and non-robust algorithms in the presence of perturbations as the order of the GF increases.

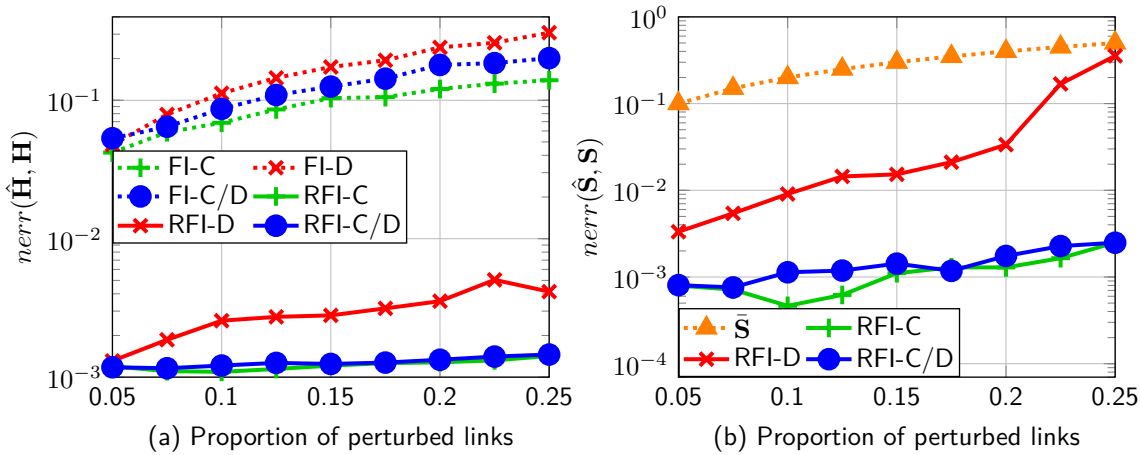


Figure 5.3: Assessing the performance of the robust GF identification algorithm and the impact of perturbations in the topology. (a) and (b) respectively show the error of estimating $\hat{\mathbf{H}}$ and $\hat{\mathbf{S}}$ using a robust or a non-robust approach for several types of perturbations.

perturbation, especially when the focus is on $\hat{\mathbf{S}}$. This may be explained because destroying links is prone to produce non-connected graphs. Nevertheless, the results show the resilience of the “RFI” algorithm, which provides low-error estimates $\hat{\mathbf{H}}$ and $\hat{\mathbf{S}}$ even when more than 20% of the links are perturbed.

Test case 3. Next, we compare the performance of our algorithms with other robust alternatives. Fig. 5.4 reports, for each algorithm, $n_{err}(\hat{\mathbf{H}}, \mathbf{H})$ as the ratio of perturbed links increases. The baselines considered are the TLS-SEM algorithm from [55], and LLS-SCP from [56]. We note that the TLS-SEM algorithm is tailored to graph signals following a SEM of the form

$$\mathbf{Y} = \mathbf{A}\mathbf{Y} + \mathbf{X} = (\mathbf{I} - \mathbf{A})^{-1}\mathbf{X}, \quad (5.28)$$

where the observations at the i -th node are represented by the values of the neighbors of i and an exogenous input. As a result, the TLS-SEM algorithm may not be well suited to deal with signals generated according to the more general model in (2.2). Taking this into account, to offer a more favorable comparison we consider two types of graph signals: (i) signals generated according to (5.28), denoted as “SEM”; and (ii) signals generated according to (2.2), denoted as “H”. It is

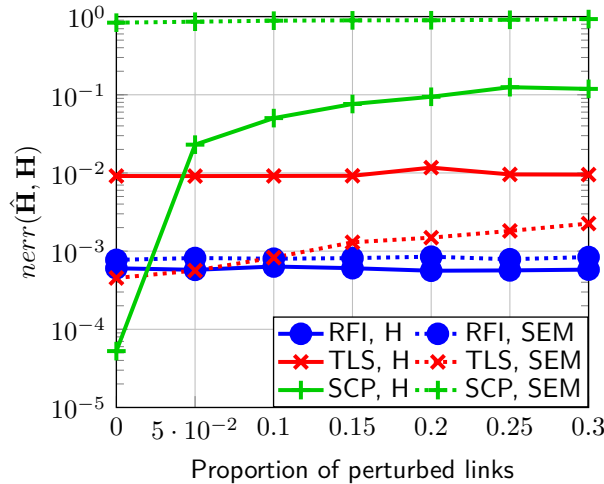


Figure 5.4: Normalized error of $\hat{\mathbf{H}}$ when estimated with the proposed algorithm and with other baselines as the ratio of perturbed links increases. Different graph-signal models are considered.

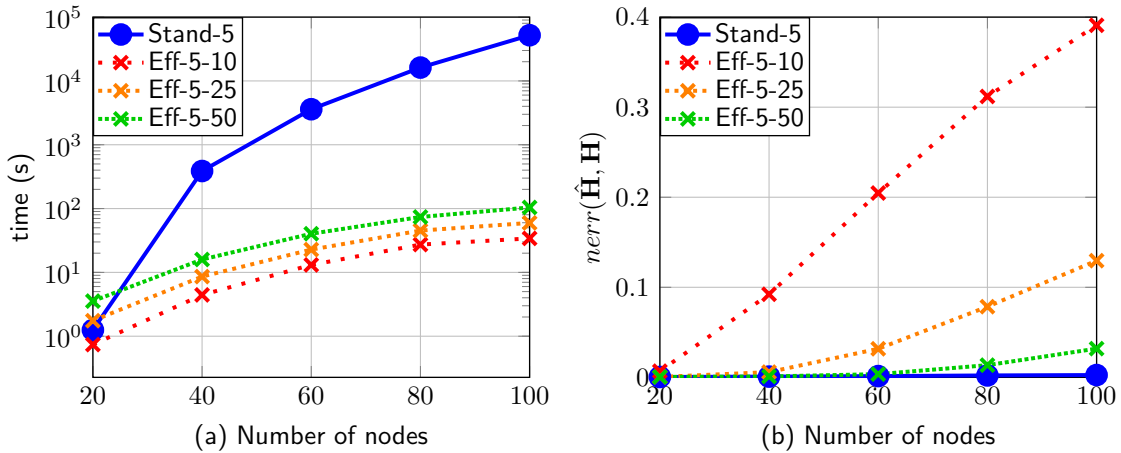


Figure 5.5: Comparing the performance of several robust GF identification algorithms. (a) and (b) respectively show the running time and error of $\hat{\mathbf{H}}$ using Algorithm 2 and Algorithm 3 as the number of nodes increases. Different values for the maximum number of iterations of the inner loops are considered.

worth noting that the “SEM” can be considered as a particular case of the model “H” when the GF $\mathbf{H}_{SEM} = (\mathbf{I} - \mathbf{A})^{-1}$ is employed.

Looking at the results in Fig. 5.4 we observe the following. When the “SEM” model is considered, TLS-SEM (denoted as “TLS”) obtains the best performance when the perturbation probability is small, and then, the performance of “TLS” and that of the “RFI” algorithm become comparable. This illustrates that our algorithm is especially suitable to deal with a large number of perturbed links. On the other hand, when the “H” model is considered, we observe that the “RFI” algorithm consistently outperforms the baselines in the presence of perturbations. The good performance of the “RFI” algorithm on both signal models highlights the flexibility of the proposed formulation since it considers more lenient assumptions than the other alternatives.

Test case 4. Now, we compare the performance of the standard and the efficient implementation of the robust identification algorithm, as described in Algorithms 2 and 3. The results are shown in Figs. 5.5a and 5.5b, where the figures depict the running time measured in seconds and $nerr(\hat{\mathbf{H}}, \mathbf{H})$

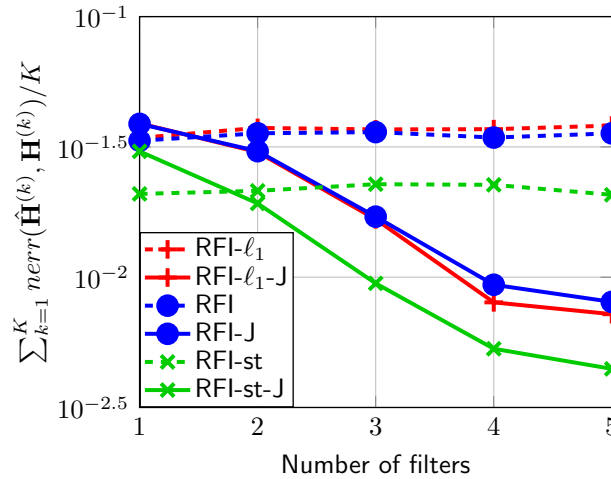


Figure 5.6: Error performance when estimating K GFs using the separate and joint approach for different values of K .

as N increases. The legend identifies first the algorithm employed, then the number of iterations of the outer loop (t_{max}), and finally the iterations of the inner loops (with $\tau_{max_1} = \tau_{max_2}$). As expected, Fig. 5.5a shows that Algorithm 3 is remarkably faster than Algorithm 2 even with medium-sized graphs, achieving a running time 10^3 times smaller when $N = 100$. On the other hand, in Fig. 5.5b we observe that “Eff-5-50” has an error that is close to the standard implementation (“Stand-5”) even though it is considerably faster. Furthermore, the trade-off between speed and estimation accuracy is also evident. “Eff-5-10” is the fastest implementation but the quality of its estimated GF may not be enough for graphs with more than 40 nodes.

Test case 5. The last experiment with synthetic data studies the benefits of the joint GF estimation. All the GFs are polynomials of the same \mathbf{S} , and for each \mathbf{H}_k we consider $M_k = 15$ noisy observations with $\eta_w = 0.01$. Fig. 5.6 shows the results, with the y-axis being the normalized error averaged across the K graphs, i.e., $\frac{1}{K} \sum_{k=1}^K nerr(\hat{\mathbf{H}}_k, \mathbf{H}_k)$, and the x-axis representing K . We compare the performance of estimating the GFs jointly (marked as “J” in the legend) or separately for the three algorithms (“RFI- ℓ_1 ”, “RFI”, and “RFI-st”) described in Test case 1. Note that “RFI-J” corresponds to the formulation in (5.16). The first thing we observe from the results in Fig. 5.6 is that the error decreases as K increases when a joint algorithm is employed. This is aligned with the discussion in Section 5.4 and illustrates the benefit of exploiting the common structure. In addition, algorithms accounting for the stationary of \mathbf{Y} outperform the non-stationary alternatives even though we only have $M = 15$ signals to estimate the covariance $\hat{\mathbf{C}}_y$.

5.6.2 Real-world datasets

To close the numerical evaluation, we test our robust GF identification algorithms over two real-world datasets.

Weather station network. This test case evaluates the ability of our algorithms to predict the temperature measured by a network of stations using the data from previous days. The data comes from the “Global Summary of the Day” dataset of the National Centers for Environmental

Models	1-Step		3-Step	
	TTS=0.25	TTS = 0.5	TTS=0.25	TTS = 0.5
LS	$6.9 \cdot 10^{-3}$	$3.1 \cdot 10^{-3}$	$2.1 \cdot 10^{-2}$	$9.1 \cdot 10^{-3}$
LS-GF	$3.3 \cdot 10^{-3}$	$3.3 \cdot 10^{-3}$	$8.4 \cdot 10^{-3}$	$8.5 \cdot 10^{-3}$
TLS-SEM	$4.0 \cdot 10^1$	$3.7 \cdot 10^{-2}$	$6.8 \cdot 10^{-1}$	$5.5 \cdot 10^{-2}$
RFI	$3.4 \cdot 10^{-3}$	$3.1 \cdot 10^{-3}$	$8.5 \cdot 10^{-3}$	$7.5 \cdot 10^{-3}$
AR(3)-RFI	$3.2 \cdot 10^{-3}$	$2.8 \cdot 10^{-3}$	$7.8 \cdot 10^{-3}$	$6.9 \cdot 10^{-3}$

Table 5.1: Performance of the algorithms in predicting the temperature for 2 prediction horizons (1 and 3) and 2 values (25% and 50%) of train-test split (TTS). The metrics shown are the average of the normalized error at each timestep $\frac{1}{M} \sum_{\kappa=1}^M nerr(\hat{\mathbf{y}}_{\kappa}, \mathbf{y}_{\kappa})$ for all samples.

Information² and we used daily temperature measurements from $N = 17$ stations in California during 2017 & 2018. Specifically, with $\mathbf{y}_{\kappa} \in \mathbb{R}^N$ collecting the measurements of the 17 stations at day κ , we consider an AR model without exogenous inputs, so that $\mathbf{y}_{\kappa} \approx \sum_{k=1}^K \mathbf{H}_k \mathbf{y}_{\kappa-k}$. The data samples were divided into two subsets, the first one (training) was used to obtain the GFs \mathbf{H}_k and the second one (evaluation) was used to assess the performance and the generalization power of the GFs obtained. Also, the data is normalized so that the signal at each station for all time samples has unitary norm.

The underlying \mathcal{G} was constructed as the unweighted 5-nearest neighbors graph, using the geographical distance between stations. Since temperature relations across stations are likely to be due to a range of factors (including, e.g., altitude), the considered adjacency (based only on geographical positions) may be imperfect, rendering our robust algorithms better suited for this task.

The estimation performance of the different algorithms is shown in Table 5.1. Since in this case the ground-truth GF is not known, we use the signal denoising error $nerr(\mathbf{y}_{\kappa}, \hat{\mathbf{y}}_{\kappa})$ to assess the quality of the schemes. In this specific experiment, the error is measured over all samples (both training and test subsets), to see a clear downward trend when increasing the number of training samples, or equivalently, the train-test split (TTS) value. The algorithms evaluated are “LS”, “LS-GF” (which postulates a GF with coefficients $\hat{\mathbf{h}} = \arg\min_{\mathbf{h}} \|\mathbf{Y} - \sum_r h_r \mathbf{S}^r \mathbf{X}\|_F^2$), “TLS”, “RFI” (which assumes an AR(1) process) and “AR(3)-RFI”. Two values of TTS (0.25 and 0.50) and two prediction horizons (1 and 3) are considered. The main observation is that “AR(3)-RFI” yields the best performance in all settings. Additionally, the results for TTS=0.25 demonstrate the benefits of considering the underlying graph in the low-sample regime, since even “LS-GF”, which relies on the imperfect $\bar{\mathbf{S}}$, outperforms “LS”. On the other hand, “LS-GF” does not seem to improve its prediction as TTS increases, while our two algorithms yield a lower prediction error.

Air quality station network. We consider an experimental setup (AR model, graph creation method...) similar to that for the weather station data but, in this case, we use 2018 & 2019 data from the United States Environmental Protection Agency³ to predict the ozone levels in a network of 17 outdoor stations in California. The stations chosen were those with at least 330 measurements each year for a selection of pollutants, and missing data was filled via first-order interpolation.

The goal here is to analyze how the prediction horizon affects the prediction error. The value of TTS chosen was 0.5, i.e. evaluation data represented 50% of the samples. Fig. 5.7 shows

²<https://www.ncei.noaa.gov/data/global-summary-of-the-day/archive/>

³<https://www.epa.gov/outdoor-air-quality-data>

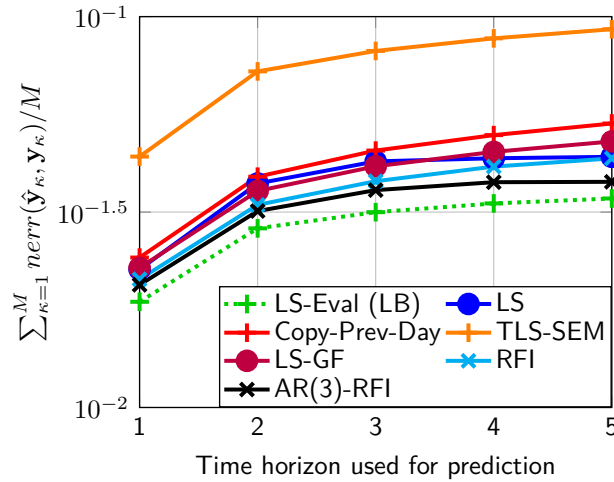


Figure 5.7: Performance of the algorithms predicting ozone levels in the AirData station network, as the time horizon of the prediction increases.

the performance of the algorithms when predicting ozone levels. As a baseline, “LS-Eval (LB)” shows the error measured on the evaluation data when obtaining the GF also using evaluation data, therefore representing a lower bound for the LS error using AR models of order 1. Also, “Copy-Prev-Day” represents the error obtained by the “identity GF”, which copies the previous day’s measurement. As in the previous example, the best performing algorithm is “AR(3)-RFI”, whose performance is close to the baseline, followed by “RFI”.

5.7 Concluding remarks

This chapter put forth a framework dealing with estimation problems in GSP where the information about (the links of) the supporting graph is uncertain. Specifically, we addressed the problem of estimating a GF (i.e., a polynomial of the GSO) from input and output graph signals under the key assumption that only a perturbed version of the true GSO was available. In contrast to the majority of existing approaches that operate on the spectral domain, we recast the true graph as an additional estimation variable and formulated an optimization problem that *jointly* estimated the GF and the true (unknown) GSO. We focused first on the case where only one GF needed to be estimated and, then, shifted to (multi-feature and AR graph signal) setups where multiple GFs have to be jointly identified. The formulated optimizations operated completely in the vertex domain and bypassed the problem of computing high-order polynomials, avoiding the challenges of dealing with the influence of perturbations in the graph spectrum as well as the numerical instability and error propagation associated with high-order matrix polynomials. While non-convex, upon blending techniques from alternating optimization and MM, the proposed algorithm was shown to be capable to find a stationary point in polynomial time. This algorithm was later modified so that the scaling of the computational complexity with respect to the number of nodes in the graph is reduced. Future work includes delving into the robust estimation of ARMA time-varying graph signals, consideration of additional graph perturbation models, and application of our robust estimation framework to other GSP problems, to name a few.

5.8 Appendix: Proof of Theorem 5.1

The proof relies on the results presented in [114, Th. 1b], so it suffices to show that our formulation and algorithm fulfill the required conditions in [114]. To that end, recall that $f(\mathbf{z})$ is the objective function in (5.9), and let $\mathbf{z}_1 := \text{vec}(\mathbf{H})$ and $\mathbf{z}_2 := \text{vec}(\mathbf{S})$ denote the $B = 2$ blocks of variables considered in our algorithm. Moreover, at each step, the function $f(\mathbf{z})$ is approximated by $u_1(\mathbf{z}_1)$ and $u_2(\mathbf{z}_2)$, corresponding to the objective functions in (5.10) and (5.12). Then, to ensure the convergence of our iterative algorithm the following conditions are required.

(C1) Each function $u_b(\mathbf{z}_b)$ must be a global upper bound of $f(\mathbf{z})$ and the first-order behavior of $u_b(\mathbf{z}_b)$ and $f(\mathbf{z})$ must be the same.

(C2) $f(\mathbf{z})$ must be regular (cf. [114]) at every point in \mathcal{Z}^* .

(C3) The level set $\mathcal{Z}^{(0)} = \{\mathbf{z} \mid f(\mathbf{z}) \leq f(\mathbf{z}^{(0)})\}$ is compact.

(C4) At least one of the problems in (5.10) and (5.12) must have a unique solution.

Next, we address each of the four conditions separately, proving that our approach satisfies all of them.

Condition **(C1)** requires the surrogate functions $u_b(\mathbf{z}_b)$ to be global upper bounds of $f(\mathbf{z})$. For the first block ($b = 1$), it is easy to see that $u_1(\mathbf{z}_1) = f(\mathbf{z})$ when the block \mathbf{z}_2 remains constant, so it satisfies the requirements. Regarding $u_2(\mathbf{z}_2)$, we approximate $f(\mathbf{z})$ with the first-order Taylor series of the logarithmic penalty. Because the \log is a concave differentiable function, it follows that its Taylor series of order one constitutes a global upper bound. Moreover, because $u_2(\mathbf{z}_2)$ is a first-order Taylor series approximation of $f(\mathbf{z})$, it also follows that the first-order behavior of $f(\mathbf{z})$ and $u_2(\mathbf{z}_2)$ is the same. Therefore, u_2 also satisfies the requirement, and hence, **(C1)** is fulfilled.

To prove **(C2)**, according to [114], a function $f(\mathbf{z})$ is regular if its non-smooth components are separable across the different blocks of variables. To show this, we decompose f as $f = g_A + g_B$, with functions g_A and g_B being defined as

$$g_A(\mathbf{H}, \mathbf{S}) = \|\mathbf{Y} - \mathbf{H}\mathbf{X}\|_F^2 + \gamma \|\mathbf{H}\mathbf{S} - \mathbf{S}\mathbf{H}\|_F^2,$$

$$g_B(\mathbf{S}) = \lambda \sum_{i,j=1}^N \log(|S_{ij} - \bar{S}_{ij}| + \delta_2) + \beta \sum_{i,j=1}^N \log(|S_{ij}| + \delta_1).$$

Since g_A is a smooth function and the non-smooth function g_B only depends on the variables on the second block, $\mathbf{z}_2 = \text{vec}(\mathbf{S})$, it follows that $f(\mathbf{z})$ is a regular function for all feasible points.

Next, we show that the level set $\mathcal{Z}^{(0)} = \{\mathbf{z} \mid f(\mathbf{z}) \leq f(\mathbf{z}^{(0)})\}$ is compact as required by **(C3)**. We start by noting that the entries of \mathbf{S} are continuous subsets of \mathbb{R} , (e.g., $S_{ij} \in \mathbb{R}_+$ when $\mathbf{S} = \mathbf{A}$), and that $\mathbf{H} \in \mathbb{R}^{N \times N}$, so $f(\mathbf{z})$ is continuous. Moreover, $f(\mathbf{z}) \leq f(\mathbf{z}^{(0)})$ implies that the functions $\|\mathbf{Y} - \mathbf{H}\mathbf{X}\|_F^2$ and $\log(|S_{ij}| + \delta_1)$ are all bounded, rendering the domain of $f(\mathbf{z})$ bounded. It follows then that the level set $\mathcal{Z}^{(0)}$ is compact.

Finally, we need to prove that either (5.10) or (5.12) has a unique solution, so that **(C4)** is fulfilled. Proposition 5.1 (see below) states that, under the two conditions required by Theorem 5.1 (i.e., \mathbf{S} does not have repeated eigenvalues, and the graph signals \mathbf{X} excite every graph frequency), the solution to (5.10) is unique. This confirms that **(C4)** is satisfied, concluding the proof.

Proposition 5.1. Let $\mathbf{H} \in \mathbb{R}^{N \times N}$, $\mathbf{S} = \mathbf{V} \text{diag}(\boldsymbol{\lambda}) \mathbf{V}^{-1} \in \mathbb{R}^{N \times N}$, and $\mathbf{X} \in \mathbb{R}^{N \times M}$ be the GF, the GSO, and the input signals in (5.10). Then, (5.10) has a unique solution w.r.t. \mathbf{H} if the following conditions are satisfied:

1. $\lambda_i \neq \lambda_{i'}$, for all $i \neq i'$ and $(i, i') \in \{1, \dots, N\}^2$.
2. Every row of $\tilde{\mathbf{X}} = \mathbf{V}^{-1} \mathbf{X}$ has at least one non-zero entry.

Proof. To simplify exposition, we focus first on the (most restrictive) setup of having only $M = 1$ input-output pair. Defining $\hat{\mathbf{h}} := \text{vec}(\mathbf{H})$, we can reformulate (5.10) as

$$\min_{\hat{\mathbf{h}} \in \mathbb{R}^{N^2}} \gamma \|(\mathbf{I} \otimes \mathbf{S} - \mathbf{S}^\top \otimes \mathbf{I}) \hat{\mathbf{h}}\|_2^2 + \|\mathbf{y} - (\mathbf{x}^\top \otimes \mathbf{I}) \hat{\mathbf{h}}\|_2^2, \quad (5.29)$$

where lowercase symbols \mathbf{y} and \mathbf{x} are used to emphasize that the output and input signals are a single N -dimensional vector. Upon defining $\mathbf{D} := \mathbf{I} \otimes \mathbf{S} - \mathbf{S}^\top \otimes \mathbf{I}$, and $\mathbf{E} := \mathbf{x}^\top \otimes \mathbf{I}$, solving (5.29) is equivalent to solving

$$\min_{\hat{\mathbf{h}} \in \mathbb{R}^{N^2}} \left\| \begin{bmatrix} \mathbf{0}_{N^2} \\ \mathbf{y} \end{bmatrix} - \mathbf{F} \hat{\mathbf{h}} \right\|_2^2 \text{ with } \mathbf{F} := \begin{bmatrix} \gamma \mathbf{D} \\ \mathbf{E} \end{bmatrix} \quad (5.30)$$

To prove that (5.30) has a unique solution, it suffices to show that \mathbf{F} is full column rank, i.e. $\nexists \mathbf{n} \in \mathbb{R}^{N^2}$ such that $\mathbf{F} \mathbf{n} = \mathbf{0}_{N+N^2}$. To show this, we first identify $\mathcal{N}(\mathbf{D})$, the null space of \mathbf{D} , and then show that $\mathbf{E} \mathbf{n} \neq \mathbf{0}_N$ for all $\mathbf{n} \in \mathcal{N}(\mathbf{D}) \setminus \{\mathbf{0}_{N^2}\}$.

We start with the characterization of $\mathcal{N}(\mathbf{D})$. Given the Kronecker structure of \mathbf{D} , each of its N^2 eigenvalues has the form $\lambda_k - \lambda_{k'}$, with $(\mathbf{V}^{-1})^\top \otimes \mathbf{V}$ being the associated eigenvectors. Leveraging that $\lambda_i \neq \lambda_{i'}$ for $i \neq i'$, it follows that only when $i = i'$ the eigenvalue of \mathbf{D} is zero. As a result, $\text{rank}(\mathbf{D}) = N^2 - N$ and $\dim(\mathcal{N}(\mathbf{D})) = N$. Equally important, the N eigenvectors associated with the N zero eigenvalues are given by $(\mathbf{V}^{-1})^\top \odot \mathbf{V}$, which, as a result, constitutes a basis spanning $\mathcal{N}(\mathbf{D})$. More formally, we concluded that $\mathcal{N}(\mathbf{D}) = \{((\mathbf{V}^{-1})^\top \odot \mathbf{V}) \boldsymbol{\theta} \mid \forall \boldsymbol{\theta} \in \mathbb{R}^N\}$. Note that \odot denotes the Khatri-Rao product.

Thus, to show that \mathbf{F} in (5.30) is full column rank we just need to prove that the only element $\mathbf{n} \in \mathcal{N}(\mathbf{D})$ that renders $\mathbf{E} \mathbf{n} = \mathbf{0}_N$ is the all-zero vector $\mathbf{0}_{N^2}$. To do so, we leverage the characterization of $\mathcal{N}(\mathbf{D})$ and write $\mathbf{E} \mathbf{n}$ as

$$\begin{aligned} \mathbf{E} \mathbf{n} &= (\mathbf{x}^\top \otimes \mathbf{I}) ((\mathbf{V}^{-1})^\top \odot \mathbf{V}) \boldsymbol{\theta} = (\mathbf{x}^\top (\mathbf{V}^{-1})^\top \odot \mathbf{V}) \boldsymbol{\theta} \\ &= \mathbf{V} \text{diag}(\boldsymbol{\theta}) (\mathbf{x}^\top (\mathbf{V}^{-1})^\top)^\top = \mathbf{V} \text{diag}(\boldsymbol{\theta}) \mathbf{V}^{-1} \mathbf{x} \\ &= \mathbf{V} \text{diag}(\boldsymbol{\theta}) \tilde{\mathbf{x}} = \mathbf{V} (\boldsymbol{\theta} \circ \tilde{\mathbf{x}}), \end{aligned} \quad (5.31)$$

where we used the property $(a \otimes b)(c \odot d) = ac \odot bd$, and \circ denotes the entry-wise product. Since \mathbf{V} is invertible, the first and last terms in (5.31) demonstrate that $\mathbf{E} \mathbf{n} = \mathbf{0}_N$ requires $\boldsymbol{\theta} \circ \tilde{\mathbf{x}} = \mathbf{0}_N$. However, condition 2) in Proposition 5.1 states that $\tilde{x}_i \neq 0$ for all i ; hence, $\boldsymbol{\theta} \circ \tilde{\mathbf{x}} = \mathbf{0}_N$ requires $\boldsymbol{\theta} = \mathbf{0}_N$. This implies that the only element in $\mathcal{N}(\mathbf{D})$ that renders $\mathbf{E} \mathbf{n} = \mathbf{0}_N$ is $\mathbf{n} = (\mathbf{V}^{-1})^\top \odot \mathbf{V} \mathbf{0}_N = \mathbf{0}_{N^2}$, concluding the proof.

The proof can be generalized for $M > 1$. In that case, the matrix \mathbf{E} has size $MN \times N^2$ and the counterpart to (5.31) establishes that having $\mathbf{E} \mathbf{n} = \mathbf{0}$ requires $\text{vec}(\text{diag}(\boldsymbol{\theta}) \tilde{\mathbf{X}}) = \mathbf{0}_{MN}$. Since Proposition 5.1 assumes that each row of $\tilde{\mathbf{X}}$ has at least one nonzero entry, it follows that $\boldsymbol{\theta} = \mathbf{0}_{MN}$, concluding the proof. \square

Chapter 6

Robust network topology inference

The last perturbation considered in our robust framework is the presence of hidden nodes, a perturbation particularly relevant (and common) in the context of network topology inference. Even though most of the existing works approach the problem of identifying the network topology based on the assumption that the whole vertex set is observed, oftentimes this scenario is not realistic. In fact, in many relevant settings, the *observed* data may correspond only to a subset of the nodes from the original graph while the rest of them remains unobserved or *hidden*. The existence of these hidden nodes entails a challenge for most of the existing methods, which require important adjustments to develop a robust alternative. Although noticeable less than its “complete network” counterpart, topology inference of networks with hidden variables has attracted some attention, with examples including Gaussian graphical model selection [57], inference of linear Bayesian networks [59], nonlinear regression [60], or brain connectivity [61], to name a few.

Motivated by this, the primary goal of this chapter is to develop a joint network topology algorithm that is robust to the presence of hidden nodes and harnesses the similarity between the graphs being estimated to enhance the quality of the estimation. In this chapter, the key assumption that enables learning the graph topology from a set of nodes is that the observed signals are observations from a random network process stationary on the unknown graphs. Therefore, we first introduce how the presence of hidden nodes influences the graph stationarity assumption, and then, we analyze how to leverage the graph similarity between nodes that are not observed. This work has been published in [70, 71].

After a brief introduction presented in Section 6.1, the remaining chapter is organized as follows. First, Section 6.2 provides a general overview of the structure resulting from the presence of hidden nodes. Then, Section 6.3 introduces the structure and problem formulation when a single graph is being learned, and Section 6.4 addresses the joint network topology inference from stationary observations. Finally, Section 6.5 evaluates the performance of the proposed algorithms, and Section 6.6 provides some concluding remarks.

6.1 Introduction

When reviewing the literature addressing the problem of network topology inference, it holds that the standard approach entails learning the topology of a single graph when observations (measurements) from all the nodes in the network are available. Nonetheless, in many relevant settings we only have access to observations from a subset of nodes, with the remaining nodes being unobserved or hidden. The existence of these hidden nodes constitutes a relevant and challenging problem since closely related values from two observed nodes may be explained not only by an edge between the two nodes but by a third latent node connected to both of them. Furthermore, many contemporary setups involve *multiple related networks*, each of them with a subset of available signals. This is the case, for example, in multi-hop communication networks in dynamic environments, in social networks where the same set of users may present different types of interactions, or in brain analytics where observations from different patients are available and the goal is to estimate their brain functional networks. When there exist several closely related networks, we can boost the performance of network topology inference by approaching the problem in a joint fashion that allows us to capture the relationship between the different graphs [123–126].

Based on the previous discussion, in this chapter we approach the problem of joint network topology inference with hidden nodes from stationary observations. Although the assumption of graph stationarity has been successfully adopted in the context of the network-topology inference problem, a formulation robust to the presence of hidden variables is still missing. To fill this gap, first we detail how hidden nodes impact the classical definition of graph stationarity and introduce the formulation of the single-network-recovery problem as a constrained optimization that accounts explicitly for the modified definitions. Then, we propose a topology inference method that simultaneously performs joint estimation of *multiple* graphs and accounts for the presence of hidden variables. Then, to fully benefit from the joint inference formulation, a critical aspect is to capture the similarity among graphs not only accounting for the observed nodes but also for the hidden ones. This is achieved by carefully exploiting the structure inherent to the presence of latent variables with a regularization inspired by group Lasso [72]. The proposed method is evaluated using synthetic and real-world graphs and compared with other related approaches.

Contributions. To summarize, the main contributions of this chapter are the following.

- (i) We develop a joint network topology inference optimization-based framework that accounts for the presence of hidden variables and exploit the graph similarity on the whole graph, not only on the observed nodes.

- (ii) We quantify the performance of the proposed algorithms using numerical experiments.

To introduce notation and facilitate exposition, we start by reviewing topology inference methods in the presence of hidden variables for partial correlation networks as well as their generalization for graph stationary signals. This review (including the generalization to stationary signals) was published in [116]. While being one of the authors of that paper, we emphasize that the contents of [116] are not included as a contribution of this Ph.D. thesis and, that, the contributions of this chapter are limited to those listed above as (i) and (ii).

6.2 Topology inference model in the presence of hidden variables

The first step towards formally stating the network topology inference from a robust perspective is to properly describe the structure resulting from the presence of hidden variables. To that end, let us assume that the unknown graph \mathcal{G} is an undirected graph with N nodes, consider that there is a random process associated with \mathcal{G} , and denote by $\mathbf{X} := [\mathbf{x}_1, \dots, \mathbf{x}_M] \in \mathbb{R}^{N \times M}$ a collection of M independent realizations of such a process. In this chapter, to model the presence of hidden nodes, we assume that we only have access to the subset of nodes $\mathcal{O} \subseteq \mathcal{V}$ with cardinality $O \leq N$. Meanwhile, the remaining $H = N - O$ nodes collected in the subset $\mathcal{H} = \mathcal{V} \setminus \mathcal{O}$ stay unobserved, and thus, only the entries of \mathbf{X} associated with the subset \mathcal{O} are observed. For the sake of simplicity and without loss of generality, we assume that the observed nodes correspond to the first O nodes of the graph. Hence, defining $\mathbf{X}_{\mathcal{O}} \in \mathbb{R}^{O \times M}$ as the submatrix formed by the first O rows of \mathbf{X} , it is clear that $\mathbf{X}_{\mathcal{O}}$ collects the values of the M available signals at the O observed nodes. Lastly, if the $N \times N$ matrices \mathbf{S} and \mathbf{C} represent, respectively, the full GSO and covariance matrix of the random graph process, these matrices, as well as the signals \mathbf{X} , present the following block structure

$$\mathbf{X} = \begin{bmatrix} \mathbf{X}_{\mathcal{O}} \\ \mathbf{X}_{\mathcal{H}} \end{bmatrix}, \quad \mathbf{S} = \begin{bmatrix} \mathbf{S}_{\mathcal{O}} & \mathbf{S}_{\mathcal{O}\mathcal{H}} \\ \mathbf{S}_{\mathcal{H}\mathcal{O}} & \mathbf{S}_{\mathcal{H}} \end{bmatrix}, \quad \mathbf{C} = \begin{bmatrix} \mathbf{C}_{\mathcal{O}} & \mathbf{C}_{\mathcal{O}\mathcal{H}} \\ \mathbf{C}_{\mathcal{H}\mathcal{O}} & \mathbf{C}_{\mathcal{H}} \end{bmatrix}. \quad (6.1)$$

Here, the submatrices $\mathbf{S}_{\mathcal{O}}$ and $\mathbf{C}_{\mathcal{O}}$, both of dimension $O \times O$, are associated with the observed variables. The observed GSO $\mathbf{S}_{\mathcal{O}}$ describes the connections between the observed nodes while the remaining blocks collect connections involving hidden nodes. Similarly, the observed covariance $\mathbf{C}_{\mathcal{O}}$ corresponds to the covariance of the random variables defined at the observed nodes. Since the graph is undirected, both \mathbf{S} and \mathbf{C} are symmetric, and thus, $\mathbf{S}_{\mathcal{O}\mathcal{H}} = \mathbf{S}_{\mathcal{H}\mathcal{O}}^{\top}$ and $\mathbf{C}_{\mathcal{O}\mathcal{H}} = \mathbf{C}_{\mathcal{H}\mathcal{O}}^{\top}$. Clearly, the sample covariance matrix $\hat{\mathbf{C}} = \frac{1}{M} \mathbf{X} \mathbf{X}^{\top}$ present the same block structure as \mathbf{C} .

With the previous definitions in place, the problem of network topology inference in the presence of hidden nodes aims at estimating the submatrix $\mathbf{S}_{\mathcal{O}}$ from the observed graph signals $\mathbf{X}_{\mathcal{O}}$ while accounting for the presence of hidden nodes. This is depicted in Fig. 6.1, where the detrimental effects of ignoring the hidden nodes is also illustrated. However, despite having observations from O nodes, there are still $H = N - O$ nodes that remain unseen and influence the observed signals $\mathbf{X}_{\mathcal{O}}$, rendering the inference problem challenging and severely ill-conditioned. As a result, to ensure the tractability of the problem, we assume that the number of hidden nodes is substantially smaller than the number of observed nodes ($O \lesssim N$) and, more importantly, we consider that there exists a known property relating the full graph signals \mathbf{X} to the full GSO \mathbf{S} . The particular relationship is further developed in subsequent sections, where we assume that \mathbf{X} is stationary (Section 6.3 and Section 6.4) on \mathbf{S} , giving rise to different network topology inference problem that are formally introduced later in the chapter. Then, the key issue to address is how the relation between \mathbf{X} and \mathbf{S} , which involves the full signals and GSO, translates to the submatrices $\mathbf{X}_{\mathcal{O}}$, $\mathbf{S}_{\mathcal{O}}$, and $\mathbf{C}_{\mathcal{O}}$ in (6.1). Nonetheless, before discussing our specific solution, a relevant question that is addressed next is how classical topology-inference approaches handle the problem of latent nodal variables.

6.2.1 Correlation and partial correlation networks with hidden variables

A key question when addressing the network topology inference problem in the (more general) scenario of hidden nodes is how to modify the existing formulations that deal with the full observable case (i.e. $O = \mathcal{V}$). The answer to that question is very different for the so-called direct methods (which consider that a link between i and j exists based only on correlation/similarity metrics between the signals observed at i and j) and indirect methods (which consider that the link (i, j)

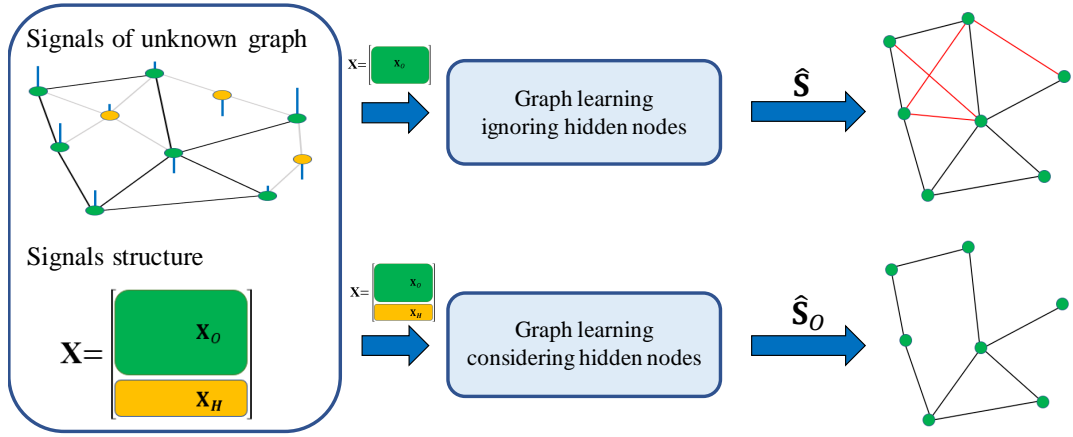


Figure 6.1: Depiction of the importance of modeling the influence of hidden variables. The left box contains the true (unknown) graph with the nodes in \mathcal{H} represented in yellow, and the graph signals collected in \mathbf{X} , which is a block matrix as in (6.1). Then, the diagram on the top infers the graph assuming that observations from all nodes are available, i.e., assuming $\mathbf{X} = \mathbf{X}_O$, and hence it wrongly estimates some edges (represented in red). On the other hand, the diagram below takes into account the whole matrix \mathbf{X} even though only \mathbf{X}_O is observed, and thus, it takes into account the presence of hidden variables and provides an accurate estimation.

exists based on the global relations/dependencies among all nodes and, hence, depends on the full observation matrices). Correlation networks, which basically assume that $\hat{\mathbf{S}}$ corresponds to (a thresholded version) of \mathbf{C} , fall into the first category. The generalization to setups with hidden nodes is indeed trivial and given by $\hat{\mathbf{S}}_O = \mathbf{C}_O$, where only the direct influence between each pair of observed nodes is considered. A relevant example within the second category are partial correlation methods (including, those for GMRF) which, in their simplest form, assume that $\mathbf{S} = \mathbf{C}^{-1}$. Under this setting, the key to include hidden variables resides in noticing that, using the expression for recovering a block of the inverse of a matrix, we can write $\mathbf{C}_O^{-1} = \mathbf{S}_O - \mathbf{Q}$, with $\mathbf{Q} = \mathbf{S}_{O\mathcal{H}}\mathbf{S}_{\mathcal{H}}^{-1}\mathbf{S}_{O\mathcal{H}}$ being a low-rank matrix since $H \ll O$. Leveraging this, the authors in [57] modified the celebrated graphical Lasso (GL) algorithm to deal with hidden variables via an augmented maximum-likelihood estimator. The resulting algorithm is known as latent variable graphical Lasso (LVGL) and is given by

$$\begin{aligned} \max_{\mathbf{S}_O, \mathbf{Q}} \quad & \log \det(\mathbf{S}_O - \mathbf{Q}) - \text{tr}(\hat{\mathbf{C}}_O(\mathbf{S}_O - \mathbf{Q})) - \lambda \|\mathbf{S}_O\|_1 - \gamma \|\mathbf{Q}\|_*, \\ \text{s. t.} \quad & \mathbf{S}_O - \mathbf{Q} \succeq \mathbf{0}, \quad \mathbf{Q} \succeq \mathbf{0}, \end{aligned} \quad (6.2)$$

where $\hat{\mathbf{C}}_O$ represents the sample covariance estimate that can be obtained from the samples in \mathbf{X}_O , the nuclear norm $\|\cdot\|_*$ promotes low-rank solutions, and λ and γ are (tunable) regularization constants. Clearly, if all the nodes are observed, we have that $\hat{\mathbf{C}}_O = \hat{\mathbf{C}}$ and $\mathbf{Q} = \mathbf{0}$ and, if those are replaced in (6.2), the classical GL formulation is recovered.

Rather than assuming that the relation between \mathbf{X} and \mathbf{S} is given by either correlations or partial-correlations, this chapter looks at setups where the operating assumption is that the observed signals are stationary on the graph. Section 6.3 first introduces this setup, and then, Section 6.4 present a joint network topology inference approach to leverage the similarity of several related graphs. Section 6.5 evaluates numerically the performance of the developed algorithms and compares it with that of classical correlation and LVGL schemes.

6.3 Topology inference from stationary signals

Now, we describe a setup in which the signals \mathbf{X} are assumed to be independent samples of a random network process stationary in \mathcal{S} . The resulting inference problem is formally stated next.

Problem 6.1. *Given the matrix \mathbf{X}_O collecting the signal values at the observed nodes of the graph \mathcal{G} , find the sparsest matrix \mathbf{S}_O encoding the structure of \mathcal{G} under the assumptions that:*

(AS1) *The number of hidden nodes is substantially smaller than the number of observed nodes, i.e., $H \ll O$.*

(AS2) *The graph signals in $\mathbf{X} \in \mathbb{R}^{N \times M}$ are independent realizations of a process that is stationary in \mathcal{S} .*

Intuitively, **(AS1)** promotes the feasibility of the inference problem by ensuring that a considerable amount of the nodes are observed. More interestingly, assumption **(AS2)** implies that the GSO and the covariance of the graph process share the eigenvectors, which is tantamount to assuming that the mapping between \mathbf{S} and \mathbf{C} can be accurately represented by a matrix polynomial. However, because the stationarity assumption involves the whole matrices \mathbf{S} and \mathbf{C} but only the observed submatrix $\hat{\mathbf{C}}_O$ is available, we need to generalize this mapping to include the effect of the hidden variables over the observed ones. Key to that end is to note that the assumption of stationarity **(AS1)** implies that \mathbf{S} and \mathbf{C} are simultaneously diagonalizable. The formulation in [32], which is valid only if $\mathcal{O} = \mathcal{V}$, approached this by extracting the eigenvectors of the (sample) covariance and, then, formulated an optimization problem guaranteeing that the extracted eigenvectors and those of the GSO were the same. Since obtaining the submatrix of the eigenvectors corresponding to the observed nodes is challenging, here we impose the graph stationarity by requiring the equality $\mathbf{C}\mathbf{S} = \mathbf{S}\mathbf{C}$ to hold [127]. Indeed, suppose that we focus on the upper left $O \times O$ block in both sides of the equality. Then, leveraging the expressions introduced in (6.1), we have that

$$\mathbf{C}_O \mathbf{S}_O + \mathbf{C}_{O\mathcal{H}} \mathbf{S}_{\mathcal{H}O} = \mathbf{S}_O \mathbf{C}_O + \mathbf{S}_{O\mathcal{H}} \mathbf{C}_{\mathcal{H}O}. \quad (6.3)$$

Equation (6.3) shows that, when hidden variables are present, we cannot simply ask \mathbf{S}_O and \mathbf{C}_O to commute, but we need also to account for the terms $\mathbf{C}_{O\mathcal{H}} \mathbf{S}_{\mathcal{H}O}$ and $\mathbf{S}_{O\mathcal{H}} \mathbf{C}_{\mathcal{H}O}$. Since **(AS1)** states that $H \ll O$, a reasonable approach is to *lift* the problem by defining the matrix $\mathbf{P} := \mathbf{C}_{O\mathcal{H}} \mathbf{S}_{\mathcal{H}O} \in \mathbb{R}^{O \times O}$ and, then, exploit **(AS1)** to impose that $\text{rank}(\mathbf{P}) \leq H \ll O$. Moreover, since both \mathbf{S} and \mathbf{C} are symmetric matrices, it follows that $(\mathbf{C}_{O\mathcal{H}} \mathbf{S}_{\mathcal{H}O})^\top = \mathbf{S}_{O\mathcal{H}} \mathbf{C}_{\mathcal{H}O}$ and, hence, $\mathbf{P}^\top = \mathbf{S}_{O\mathcal{H}} \mathbf{C}_{\mathcal{H}O}$. Then, by making the general assumption that graphs are typically sparse, we can formulate Problem 6.1 as an optimization framework in which we attempt to find a sparse matrix \mathbf{S}_O and a low-rank matrix \mathbf{P} that satisfy **(AS1)** and **(AS2)** by solving

$$\begin{aligned} \min_{\mathbf{S}_O, \mathbf{P}} \quad & \|\mathbf{S}_O\|_0 \\ \text{s. t.} \quad & \mathbf{C}_O \mathbf{S}_O + \mathbf{P} = \mathbf{S}_O \mathbf{C}_O + \mathbf{P}^\top, \\ & \text{rank}(\mathbf{P}) \leq H, \\ & \mathbf{S}_O \in \mathcal{S}, \end{aligned} \quad (6.4)$$

where the equality constraint accounts for **(AS2)** and the second constraint for **(AS1)**, as explained before. Notice that the first constraint assumes perfect knowledge of the observed covariance matrix \mathbf{C}_O , but this might not be attainable in practice, motivating the need for robust versions of this formulation as discussed in Section 6.3.2. Finally, recall that the set \mathcal{S} specify additional properties that \mathbf{S}_O must satisfy. In the remainder of this section we will focus on the case where the GSO is an adjacency matrix, so we consider the set $\mathcal{S} = \mathcal{S}_A$.

Even though (6.10) indeed solves Problem 6.1, it is a non-convex optimization problem, and hence, challenging to solve, motivating the development of convex approximations.

6.3.1 Topology inference with stationary observations as a convex optimization

It is important to notice that the presence of the rank constraint and the ℓ_0 norm in (6.4) render the problem non-convex and computationally hard to solve, motivating the need for convex relaxations. To achieve this, instead of enforcing a rank constraint, we will augment the objective with the *nuclear norm* as its convex surrogate. Similarly, the ℓ_0 norm can be replaced with the ℓ_1 norm, its closest convex approximation. After applying these relaxations to (6.4) we obtain the following *convex* optimization problem

$$\begin{aligned} \min_{\mathbf{S}_O, \mathbf{P}} \quad & \|\mathbf{S}_O\|_1 + \gamma \|\mathbf{P}\|_* \\ \text{s. t.} \quad & \mathbf{C}_O \mathbf{S}_O + \mathbf{P} = \mathbf{S}_O \mathbf{C}_O + \mathbf{P}^\top, \quad \mathbf{S}_O \in \mathcal{S}_A, \end{aligned} \quad (6.5)$$

where γ is a regularization constant encoding the relative importance of the low-rank vs the sparsity promoting term.

Even though replacing the original ℓ_0 norm with the convex ℓ_1 norm constitutes a common approach, it is well-known that non-convex surrogates can lead to sparser solutions. Indeed, a more sophisticated alternative in the context of sparse recovery is to define δ as a small positive number and replace the ℓ_0 norm with a (non-convex) logarithmic penalty [111] as

$$\|\mathbf{S}_O\|_0 \approx \sum_{i,j=0}^O \log(|S_{ij}| + \epsilon_0) \quad (6.6)$$

with ϵ_0 being a small positive constant. However, because the logarithm is a concave function it renders the optimization non-convex. An efficient way to handle this issue consists on relying on a MM approach [112], which considers an iterative linear approximation to the concave objective and leads to an *iterative* re-weighted ℓ_1 minimization. For the problem at hand, and with $t = 1, \dots, T$ being the iteration index, the resulting optimization is

$$\begin{aligned} \mathbf{S}_O^{(t+1)} &:= \underset{\mathbf{S}_O, \mathbf{P}}{\operatorname{argmin}} \sum_{i,j=1}^O W_{ij}^{(t)} [S_O]_{ij} + \gamma \|\mathbf{P}\|_* \\ \text{s. t.} \quad & \mathbf{C}_O \mathbf{S}_O + \mathbf{P} = \mathbf{S}_O \mathbf{C}_O + \mathbf{P}^\top, \\ & \mathbf{S}_O \in \mathcal{S}_A, \end{aligned} \quad (6.7)$$

with $W_{ij}^{(t)}$ being defined as $W_{ij}^{(t)} = (S_{ij}^{(t-1)} + \epsilon_0)^{-1}$. Since the iterative algorithm penalizes (assigns a larger weight to) entries of \mathbf{S}_O that are close to zero, the obtained solution is typically sparser at the expense of a higher computational cost. Finally, note that the absolute values can be removed whenever the constraint $[S_O]_{ij} \geq 0$ is enforced.

6.3.2 Robust network inference

In most scenarios the covariance matrix \mathbf{C}_O is not known perfectly but, instead, only an estimate $\hat{\mathbf{C}}_O$ is available. This is indeed the case for the setting described in Problem 6.1, where we only have access to a finite number M of graph signals and we estimate the covariance as

$\hat{\mathbf{C}}_{\mathcal{O}} = M^{-1}(\mathbf{X}_{\mathcal{O}}\mathbf{X}_{\mathcal{O}}^{\top})$. While different choices to accommodate the discrepancies between the true covariance and its sampled version exist, here we simply chose to relax the commutative constraint in (6.5), rewriting it using a Frobenius norm

$$\begin{aligned} \min_{\mathbf{S}_{\mathcal{O}}, \mathbf{P}} \quad & \|\mathbf{S}_{\mathcal{O}}\|_1 + \gamma\|\mathbf{P}\|_* \\ \text{s. t.} \quad & \|\hat{\mathbf{C}}_{\mathcal{O}}\mathbf{S}_{\mathcal{O}} + \mathbf{P} - \mathbf{S}_{\mathcal{O}}\hat{\mathbf{C}}_{\mathcal{O}} - \mathbf{P}^{\top}\|_F \leq \epsilon, \quad \mathbf{S}_{\mathcal{O}} \in \mathcal{S}_{\mathbf{A}}. \end{aligned} \quad (6.8)$$

The value of the non-negative constant ϵ should be selected based on prior knowledge on the noise level present in the observations and, more importantly, the number of samples M used to estimate the covariance. It must be large enough to ensure the problem is feasible but not too large so the constraint is not active. In the limit, when an infinite number of realizations is available, then $\mathbf{C} = \hat{\mathbf{C}}$ so we can set $\epsilon = 0$ and (6.8) boils down to (6.5). The same update on the constraint can be applied to (6.7) to account for imperfect knowledge of the covariance.

6.4 Joint inference from stationary signals in the presence of hidden variables

The last network topology inference task that we will be analyzing from a robust perspective deals with the estimation of multiple related graphs. The driving idea is to exploit the existing relation between the different graphs in a joint optimization framework to enhance the quality of the estimated networks.

To formally introduce the problem of joint graph topology inference in the presence of hidden variables, let us assume that K undirected graphs $\{\mathcal{G}^{(k)}\}_{k=1}^K$ are defined over the same set of nodes \mathcal{V} , and denote as $\mathbf{X}^{(k)} = [\mathbf{x}_1^{(k)}, \dots, \mathbf{x}_{M_k}^{(k)}] \in \mathbb{R}^{N \times M_k}$ the collection of (zero-mean) M_k signals defined on top of each unknown graph $\mathcal{G}^{(k)}$. Furthermore, in analogy to previous sections, consider that for each graph only O nodes contained in the subset \mathcal{O} are observed while the remaining H node in the set \mathcal{H} remain hidden. Without loss of generality, let the signals associated with the observed nodes be collected in the first O rows of $\mathbf{X}^{(k)}$ and denote them as $\mathbf{X}_{\mathcal{O}}^{(k)} \in \mathbb{R}^{O \times M_k}$. Then, the distinction between observed and hidden nodes renders each matrix $\mathbf{S}^{(k)}$ and $\hat{\mathbf{C}}^{(k)}$ with the characteristic block structure introduced in (6.1). Also note that, since the K graphs are undirected, the matrices $\mathbf{S}^{(k)}$ and $\hat{\mathbf{C}}^{(k)}$ are symmetric.

With these considerations in place, the problem of joint topology inference in the presence of hidden variables is described next.

Problem 6.2. *Given the $O \times M_k$ matrices $\{\mathbf{X}_{\mathcal{O}}^{(k)}\}_{k=1}^K$ collecting the signal values at the observed nodes for each graph $\mathcal{G}^{(k)}$, find the sparsest matrices $\{\mathbf{S}_{\mathcal{O}}^{(k)}\}_{k=1}^K$ encoding the structure of the K graphs under the assumptions that:*

(AS1) *The number of hidden nodes is much smaller than the number of observed nodes, i.e., $H \ll O$.*

(AS2) *The signals $\mathbf{X}^{(k)}$ are realizations of a random process that is stationary in $\mathbf{S}^{(k)}$.*

(AS3) *The distance between the K graphs is small according to a particular metric $d(\mathbf{S}^{(k)}, \mathbf{S}^{(k')})$.*

Accounting for the hidden variables implies modeling their influence over the observed nodes without any additional observation, thus rendering the inference problem a challenge. To ensure the tractability of the problem, **(AS1)** guarantees that most of the nodes are observed while **(AS2)** establishes a relation between the graph signals and the whole unknown graph via graph stationarity. Then, **(AS3)** guarantees that the K graphs are similar so we can benefit from inferring

them in a joint setting. The key question now is how to exploit the graph similarity from edges involving observed nodes but also from edges involving hidden nodes.

In the following section, we exploit the aforementioned assumptions and the block structure resulting from the presence of hidden variables to approach Problem 6.2 by solving a convex optimization problem.

6.4.1 Modeling hidden variables in the joint inference problem

The first step towards formulating an optimization problem that solves Problem 6.2 is to account for the presence of the hidden nodes in the stationary assumption **(AS2)**. This can be achieved by following the same reasoning as in Section 6.3. Then, for each graph k , we define the associated $O \times O$ matrix $\mathbf{P}^{(k)} := \mathbf{C}_{\mathcal{O}\mathcal{H}}^{(k)}(\mathbf{S}_{\mathcal{O}\mathcal{H}}^{(k)})^\top$, which combined with the commutativity of $\mathbf{C}^{(k)}$ and $\mathbf{S}^{(k)}$ that stems from the stationarity assumption results in

$$\mathbf{C}_{\mathcal{O}}^{(k)}\mathbf{S}_{\mathcal{O}}^{(k)} + \mathbf{P}^{(k)} = \mathbf{S}_{\mathcal{O}}^{(k)}\mathbf{C}_{\mathcal{O}}^{(k)} + (\mathbf{P}^{(k)})^\top. \quad (6.9)$$

Also note that the matrices $\mathbf{P}^{(k)}$ are the product of two matrices of sizes $O \times H$ and $H \times O$ so due to **(AS1)** it follows that the rank of $\mathbf{P}^{(k)}$ is upper bounded by H .

With the previous considerations in place, we approach the sparse joint topology inference problem in the presence of hidden nodes by means of the following non-convex optimization problem

$$\begin{aligned} \min_{\{\mathbf{S}_{\mathcal{O}}^{(k)}, \mathbf{P}^{(k)}\}_{k=1}^K} & \sum_k \alpha_k \|\mathbf{S}_{\mathcal{O}}^{(k)}\|_0 + \sum_{k < k'} \beta_{k,k'} d_S(\mathbf{S}_{\mathcal{O}}^{(k)}, \mathbf{S}_{\mathcal{O}}^{(k')}) \\ & + \sum_{k < k'} \eta_{k,k'} d_P(\mathbf{P}^{(k)}, \mathbf{P}^{(k')}) \\ \text{s. t.} & \text{rank}(\mathbf{P}^{(k)}) \leq H, \\ & \|\hat{\mathbf{C}}_{\mathcal{O}}^{(k)}\mathbf{S}_{\mathcal{O}}^{(k)} + \mathbf{P}^{(k)} - \mathbf{S}_{\mathcal{O}}^{(k)}\hat{\mathbf{C}}_{\mathcal{O}}^{(k)} - (\mathbf{P}^{(k)})^\top\|_F^2 \leq \epsilon, \\ & \mathbf{S}_{\mathcal{O}}^{(k)} \in \mathcal{S}. \end{aligned} \quad (6.10)$$

The first and second constraints capture assumptions **(AS1)** and **(AS2)**, with ϵ being a small positive number capturing the fidelity of the sample covariance. The set \mathcal{S} encodes the properties of the desired GSOs. In this section we will focus on the case where the GSO is given by the adjacency matrix of the underlying undirected graph with non-negative weights and no self-loops. Thus, from now onwards we set the feasibility set $\mathcal{S} = \mathcal{S}_{\mathbf{A}}$, with $\mathcal{S}_{\mathbf{A}}$ as defined in (2.19). Other GSOs such as the normalized Laplacian can be accommodated via minor adaptations to \mathcal{S} ; see [32].

Similar to standard joint inference approaches [125], the objective function of (6.10) captures the similarity of the K graphs with the function $d_S(\cdot, \cdot)$. Nevertheless, when accounting for the presence of hidden variables, assumption **(AS3)** is also reflected in the unobserved blocks of the GSOs. This important observation, captured by the function $d_P(\cdot, \cdot)$, allows us to incorporate additional structure reducing the degrees of freedom and rendering the problem more manageable. More specifically, note that the matrix $\mathbf{P}^{(k)}$ is given by the product of $\mathbf{C}_{\mathcal{O}\mathcal{H}}^{(k)}$ and $(\mathbf{S}_{\mathcal{O}\mathcal{H}}^{(k)})^\top$ with the latter being a submatrix of a sparse GSO, so it can be seen that the matrices $\mathbf{P}^{(k)}$ present a column-sparse structure. Furthermore, since the K graphs are similar, the submatrices $\mathbf{S}_{\mathcal{O}\mathcal{H}}^{(k)}$ are also similar, which implies that the matrices $\mathbf{P}^{(k)}$ present a similar column-sparsity pattern. In other words, the columns with non-zero entries are likely to be placed in the same positions for the different matrices $\mathbf{P}^{(k)}$. By designing a distance function $d_P(\cdot, \cdot)$ that exploits this additional

structure we improve the estimation of the matrices $\mathbf{P}^{(k)}$, resulting in a better estimation of the matrices $\mathbf{S}_O^{(k)}$.

The non-convexity of (6.10), which arises from the presence of the rank constraint and the ℓ_0 norm, renders the optimization problem computationally hard to solve, leading us to implement some convex relaxations that are detailed next.

6.4.2 Convex relaxations for the joint topology inference

The rank constraints are commonly avoided by augmenting the objective function with a nuclear norm penalty, which promotes low-rank solutions by seeking matrices with sparse singular values. However, this penalty does not preserve the characteristic column sparsity of the matrices $\mathbf{P}^{(k)}$. To circumvent this issue, in contrast to [116], we employ the group Lasso regularization [72] and rely on the fact that, in this particular setting, we can promote low rankness by reducing the number of non-zero columns while still achieving a reliable estimate. Then, we replace the ℓ_0 norm by a reweighted ℓ_1 minimization [111], an iterative algorithm rooted on a logarithmic penalty that: i) converges to a stationary point [114]; and ii) usually outperforms the widely used ℓ_1 norm.

By leveraging the aforementioned relaxations we address the joint topology inference problem in the presence of hidden variables by solving an iterative method. Under this approach, for each iteration, we solve the following convex problem

$$\begin{aligned}
\min_{\{\mathbf{S}_O^{(k)}, \mathbf{P}^{(k)}\}_{k=1}^K} & \sum_k \alpha_k \text{vec}(\mathbf{W}^{(k)})^\top \text{vec}(\mathbf{S}_O^{(k)}) & (6.11) \\
& + \sum_{k < k'} \beta_{k,k'} \|\mathbf{S}_O^{(k)} - \mathbf{S}_O^{(k')}\|_1 \\
& + \sum_k \gamma_k \|\mathbf{P}^{(k)}\|_{2,1} + \sum_{k < k'} \eta_{k,k'} \left\| \begin{bmatrix} \mathbf{P}^{(k)} \\ \mathbf{P}^{(k')} \end{bmatrix} \right\|_{2,1} \\
& + \sum_k \mu_k \|\hat{\mathbf{C}}_O^{(k)} \mathbf{S}_O^{(k)} + \mathbf{P}^{(k)} - \mathbf{S}_O^{(k)} \hat{\mathbf{C}}_O^{(k)} - (\mathbf{P}^{(k)})^\top\|_F^2 \\
\text{s. t.} & \mathbf{S}_O^{(k)} \in \mathcal{S}.
\end{aligned}$$

To compute the weight matrices $\mathbf{W}^{(k)}$, let $t = 1 \dots T$ denote the iteration index (omitted in the expression above to alleviate the notation), and compute the k -th weight matrix for the t -th iteration as $W_{ij}^{(k,t)} = (S_{O_{ij}}^{(k,t-1)} + \delta)^{-1}$ with $S_{O_{ij}}^{(k,t-1)}$ being the solution obtained during the $(t-1)$ -th iteration and δ a small positive constant. Hence, for each iteration t we first compute the weight matrices $\mathbf{W}^{(k,t)}$ and, then, employ those to estimate the matrices $\mathbf{S}_O^{(k,t)}$ and $\mathbf{P}^{(k,t)}$. Coming back to the formulation in (6.11), note that the distance $d_S(\cdot, \cdot)$ is set to the ℓ_1 norm to promote similar edges on the K graphs. The norm $\|\cdot\|_{2,1}$ represents the group Lasso penalty by first computing the ℓ_2 norm of the columns of the input matrix and then the ℓ_1 norm of the resulting vector. To capture the similar column-sparsity pattern of the matrices $\mathbf{P}^{(k)}$ resulting from the similarity of the K graphs, we design the function $d_P(\cdot, \cdot)$ relying on the group Lasso penalty. More specifically, we concatenate each pair of matrices $\mathbf{P}^{(k)}$ and $\mathbf{P}^{(k')}$ to create a tall matrix and then promote column sparsity on the tall matrix with the $\ell_{2,1}$ norm. Note that a column of all zeros in the tall matrix implies that the same column in $\mathbf{P}^{(k)}$ and $\mathbf{P}^{(k')}$ will only contain zeros, thus promoting the desired structure. Finally, it is worth noting that we moved the commutativity constraint to the objective function. Due to the iterative nature of the proposed method, the estimation of the observed GSO during the first iteration might be far from the true GSO, and hence, a more restrictive constraint

as the one employed in (6.10) might be misleading. Augmenting the objective function with the commutativity penalty is more amenable to an iterative approach.

6.5 Numerical experiments

In this section we assess numerically the performance of the proposed algorithms, comparing them with different baselines, including the LVGL counterpart presented in (6.2). To compare the different scenarios raised in this chapter, first we consider the setting where a single graph is learned from stationary observations (Section 6.3), and then, we evaluate the performance of the proposed joint network topology algorithm (Section 6.4).

6.5.1 Numerical experiments based on joint inference

Lastly, we evaluate the performance of the joint network topology inference algorithms developed in Section 6.4, which are evaluated over synthetic and real-world graphs. The error of the recovered graphs is computed as

$$\sum_{k=1}^K \|\mathbf{S}_o^{(k)} - \hat{\mathbf{S}}_o^{(k)}\|_F^2 / K \|\mathbf{S}_o^{(k)}\|_F^2, \quad (6.12)$$

and when the graphs are randomly generated, they are sampled from an ER model with $N = 20$ nodes and edge probability $p = 0.2$. The code for the following experiments is available on GitHub¹ and the interested reader is referred there for specific implementation details.

Test case 1. We evaluate the influence of the hidden variables and its detrimental effect on the topology inference task when the true covariance matrix is known. The results are depicted in Fig. 6.2a, where we report the error of the recovered graphs, computed according to (6.12), for several models as the number of hidden variables increases on the x-axis. The error is averaged over 64 realizations with ER graphs. The considered models are: (i) ‘‘PGL’’, which stands for the method introduced in (6.11); (ii) ‘‘PNN’’, which denotes the reweighed algorithm proposed in [70] augmented with the joint penalty $d_S(\cdot, \cdot)$ to perform the joint inference; and (iii) ‘‘No hidden’’, which is a joint inference method unaware of the presence of hidden variables similar to the work in [125]. In addition, for each model we let K take the values in $\{3, 6\}$. Looking at the results, we can observe that ‘‘PGL’’ and ‘‘PNN’’, which take into account the presence of hidden variables, outperform the method ‘‘No hidden’’, showcasing the benefit of a robust formulation. Also, the method proposed in (6.11) outperforms ‘‘PNN’’, the other alternative accounting for hidden variables. This reflects the advantage of employing the group Lasso regularization and incorporating the graph similarity through the careful design of the function $d_P(\cdot, \cdot)$. Lastly, it is worth noting that the performance improves for higher values of K , achieving better results when more related graphs are available.

Test case 2. Next, we evaluate the influence of the number of observed signals and compare the performance of the proposed approach with other related alternatives. In this experiment, only a single hidden node is considered. To that end, in Fig. 6.2b we show the mean normalized error of the recovered graphs on the y-axis as the number of samples increases on the x-axis. The

¹https://github.com/reysam93/hidden_joint_inference/tree/ICASSP2022

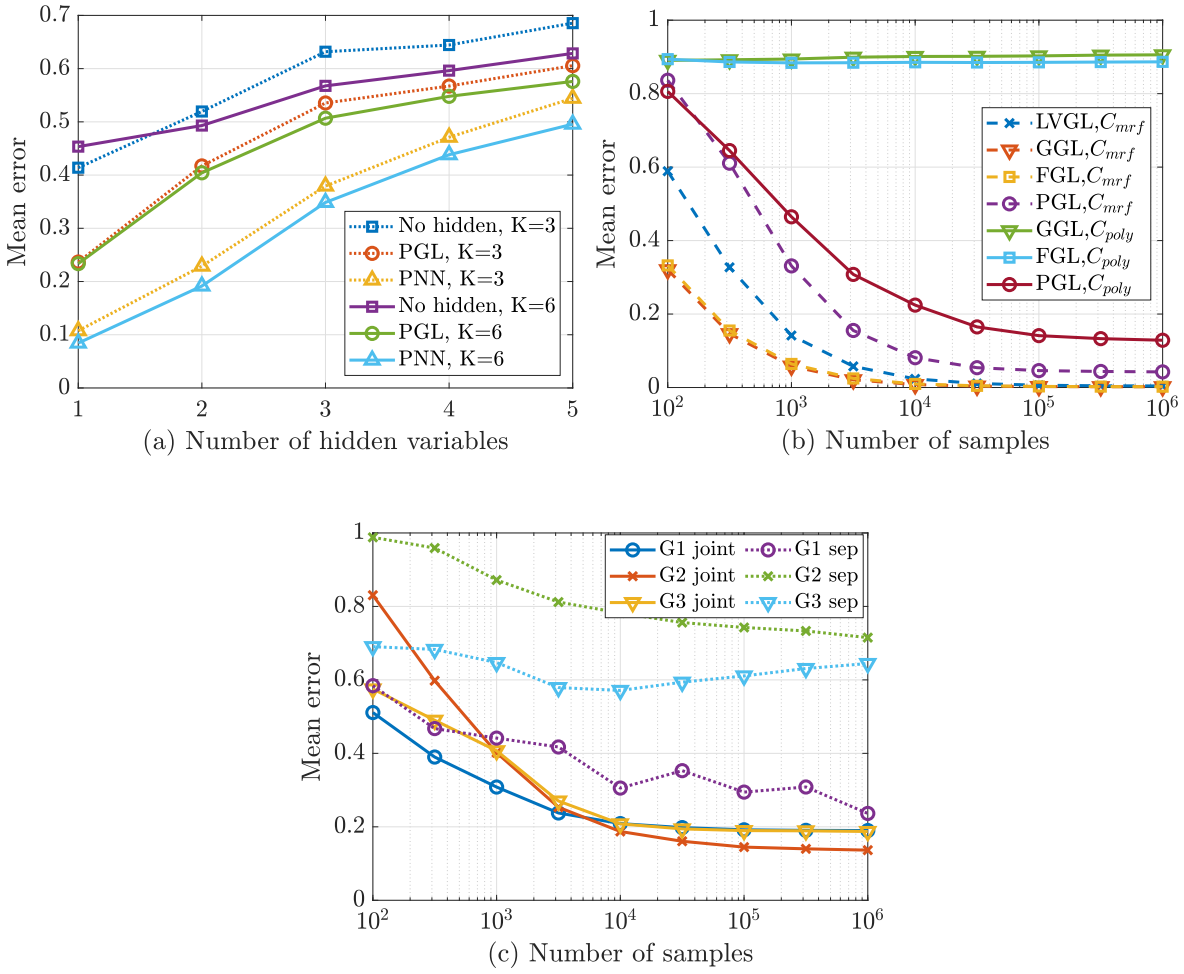


Figure 6.2: Numerical validation of the proposed algorithm. a) Mean error of 100 realizations as the number of hidden variables increases for different models and values of K . b) Mean error of the recovered graphs for several algorithms as the number of samples increases. c) Mean error of the recovered graphs for joint and separate approaches as the number of samples increases. The first two experiments use ER graphs with $N = 20$ and $p = 0.2$ and the third one employs real-world graphs.

error is computed as in the previous experiment and the mean is considered over 30 realizations of $K = 3$ ER graphs with 10 realizations of random covariance matrices for each, resulting in a total of 300 realizations. We compare the proposed model (“PGL”) with latent variable graphical Lasso (“LVGL”) [57], and with group and fusion graphical Lasso (“GGL” and “FGL”), both from [124]. For each model, signals are generated using two different types of covariance matrices: (i) $\mathbf{C}_{MRF} = (\sigma\mathbf{I} + \phi\mathbf{S})^{-1}$ where ϕ is a positive random number and σ is a positive number so that \mathbf{C}_{MRF}^{-1} is positive semi-definite; and $\mathbf{C}_{poly} = \mathbf{H}^2$ where the matrix \mathbf{H} is a graph filter with random coefficients \mathbf{h} . By looking at the Fig. 6.2b, it can be observed that, when \mathbf{C}_{MRF} is employed, the graphical Lasso models slightly outperform the proposed approach. This is expected since they are tailored for this specific type of covariance matrices. However, we can also see that the performance of the proposed algorithm is close to that of the alternatives, illustrating the benefits of considering both the joint optimization and the presence of hidden variables. On the other hand, when we focus on the covariance matrices \mathbf{C}_{poly} , it is evident that the proposed method “PGL” clearly outperforms the alternatives, demonstrating that the proposed method is based on more lenient assumptions. Note that the results for “LVGL” for the polynomial covariance

are not included since the error was too high.

Test case 3. Finally, we test the proposed algorithm and the impact of performing the topology inference in a joint fashion using real-world graphs. We employ three graphs defined on a common set of 32 nodes. Nodes represent students from the University of Ljubljana and the different networks encode different types of interactions among the students². The error is computed as detailed in and one hidden variable is considered. The results, illustrated in Fig. 6.2c, show the error of the recovered graphs as the number of samples increases. The displayed error is the mean of 30 realizations of random stationary graph signals and only one hidden variable is considered. Also, for each of the three graphs we include the performance of both the joint and the separate estimation. It can be observed that the recovery of the three graphs improves when a joint approach is followed, showcasing the benefits of exploiting the existing relationship between the different networks. Furthermore, this experiment confirms that the developed method is also suitable for real applications.

6.6 Conclusion

In this chapter, we faced the challenging problem of joint graph topology inference in the presence of hidden nodes. First, we detailed the simpler case where the goal was to learn a single graph from stationary observations, and then, we presented a new method for approaching the joint graph topology inference in the presence of hidden nodes. To tackle this ill-posed inference problem, we assume that (i) the number of hidden nodes H is much smaller than the number of observed nodes O ; (ii) the observed signals are realizations from a random process stationary in $\mathbf{S}^{(k)}$; and (iii) the K graphs are closely related. Furthermore, we exploit the inherent block structure of the matrices $\mathbf{C}^{(k)}$ and $\mathbf{S}^{(k)}$ to solve the joint topology inference problem by solving an optimization framework. A reweighted ℓ_1 norm to promote sparse solutions is employed, and the stationarity assumption is adapted to the presence of hidden nodes by defining the (unknown) low-rank lifting matrices $\mathbf{P}^{(k)}$. Instead of relying on the nuclear norm, low-rank matrices $\mathbf{P}^{(k)}$ are achieved by promoting column-sparsity with the group Lasso penalty. Moreover, the similarity of the K graphs is leveraged in two ways. First, for each pair of graphs, we look for matrices $\mathbf{S}_O^{(k)}$ with a similar edge pattern by minimizing the ℓ_1 penalty, and second, we look for matrices $\mathbf{P}^{(k)}$ with a similar column sparsity pattern. The proposed method is evaluated using synthetic and real-world graphs, and a comparison with other baseline methods based on graph stationarity and on graphical Lasso is provided.

²The original data can be found at <http://vladoviki.fmf.uni-lj.si/doku.php?id=pajek:data:pajek:students>

Chapter 7

Concluding remarks

The encompassing objective of this thesis was to contribute to building the foundations of a robust paradigm to address classical problems within GSP while modeling the detrimental influence of perturbations in the observed data. To that end, we considered several types of perturbations that were classified into two broad classes: (i) perturbations in the graph signals; and (ii) perturbations in the topology of the graph. First, we dealt with imperfections in the observed signals, which often lead to tractable problems and have been more studied in related literature. Since incorporating the influence of imperfections in the signals in GSP tasks is well studied, when the magnitude of the perturbations is small with respect to the original signal (and especially when the perturbations capture the presence of noise), we addressed settings where the magnitude of the perturbations was large. Furthermore, the focus in Chapter 3 and Chapter 4 was on using processing schemes that had not received too much attention in the literature, trying to characterize their estimation performance (including those based on GNN architectures). In the second part of the thesis, we dealt with imperfections in the topology of the graph, which give rise to more challenging problems and have been significantly less studied in the literature. In Chapter 5, the focus was on estimating a graph filter but we also recovered an enhanced estimate of the unperturbed topology as a byproduct. Finally, Chapter 6 tackled directly the problem of learning a graph, but for the challenging setup of network tomography, where hidden (unobservable) nodes are present. Finally, in each chapter we run an extensive battery of experiments to gain additional insights, test the robustness of the methods, compare them to existing state-of-the-art alternatives, and assess their potential applicability in real-world problems.

The remainder of the chapter is devoted to describing the degree of fulfillment of the objectives listed in Section 1.2 (Section 7.1), and propose future lines of work (Section 7.2).

7.1 Revisiting the proposed goals

First, the objective **(O1)** considered that the observed graph signals were corrupted with noise and the goal was to develop non-linear algorithms to separate the noise from the signal. This was addressed in Chapter 3. Since linear denoising schemes had been investigated extensively, we designed two (untrained) graph-aware NNs that incorporated the information encoded in the GSO through different strategies. The GCG relied on fixed (non-learnable) GFs to model convolutions

in the vertex domain while the GDec employed a nested collection of graph upsampling operators to progressively increase the input size, limiting the degrees of freedom of the architecture, and providing more robustness to noise. Furthermore, we provided theoretical guarantees on the denoising performance of both architectures when denoising K -bandlimited graph signals under some simplifying assumptions, and then, we numerically illustrated that the proposed architectures were also capable of denoising graph signals in more general settings. Interestingly, the proposed GDec is not limited to the graph signal denoising task. In fact, we successfully leveraged the GDec to learn a mapping from an input graph signal \mathbf{x} defined on some graph \mathcal{G}_1 to a graph signal \mathbf{y} defined on some graph \mathcal{G}_2 by learning a latent space common to \mathbf{x} and \mathbf{y} , a problem intimately related with canonical correlation analysis.

Objective **(O2)**, approached in Chapter 4, considered signals with missing values and the goal was to interpolate the original signals under the assumption that they were DSGS. Interpreting the observed (non-missing) values as samples gathered via an AGSS, first we studied the recovery of the original signal from the local observations when the seeding nodes were known. Then, we considered more challenging sampling configurations where the seeding nodes, the diffusing filter, or both, were unknown.

In the objective **(O3)**, the perturbations represented uncertainty in the edges of the observed graph and the aim was to develop a robust GF identification scheme from input-output observations. To that end, Chapter 5 recast the true graph as an additional estimation variable and formulated an optimization problem that jointly estimated the GF and the true (unknown) GSO. First, we focused on the case where only one GF needed to be estimated and, then, shifted to setups where multiple GFs have to be jointly identified. The optimization problem was formulated completely in the vertex domain bypassing the error propagation associated with high-order matrix polynomials as well as the challenges of dealing with the influence of perturbations in the graph spectrum. Since the optimization problem was non-convex, we blended techniques from alternating optimization and MM to obtain an iterative convex algorithm capable to find a stationary point in polynomial time. This algorithm was later modified so that the scaling of the computational complexity with respect to the number of nodes in the graph is reduced.

Finally, Chapter 6 deals with objective **(O4)**, which focused on developing a joint network topology inference algorithm robust to the presence of hidden nodes from stationary observations. To ensure the tractability of the problems, we assumed that the number of observed nodes was substantially larger than the number of hidden nodes, and formulated constrained optimization problems that accounted for the topological and signal constraints. The cornerstone of the proposed algorithm was to exploit the block-matrix structure resulting from the presence of hidden variables. This structure allowed us to reformulate the classical definition of stationarity to account for the presence of hidden variables and, moreover, it revealed a column sparsity pattern on the matrices $\mathbf{P}^{(k)}$ (associated with hidden nodes) that we exploited to promote graph similarity between edges involved with hidden nodes via a group Lasso regularization.

7.2 Future lines of research

To conclude the document, we present several research directions to strengthen and grow the robust GSP methodology implemented in this thesis. The suggested lines range from considering tractable generalizations of the schemes discussed in the previous chapters to more general and ambitious research directions. The lines in the first group are in general well-defined and likely to be successfully addressed in the short/medium term, while the ones in the second group can be

understood as a research plan for the medium/long term.

Generalizing the theoretical characterization of the denoising with GNNs to more general settings. Enhancing our current understanding of NNs (including GNNs) is a relevant ongoing research problem. In this sense, generalizing the theoretical analysis from Chapter 3 to hold with more lenient assumptions would be beneficial. To be precise, our analysis considered the graph signals being bandlimited, the noise to be white, and the graphs being drawn from an SBM. Considering more general noise distributions is a tractable task. Regarding the assumptions about the signals, dealing with other generative models (such as diffused or smooth signals) requires being able to establish a link between either the spectrum or the vertex distribution of the SBM and the signals at hand. Lastly, considering graphs beyond SBMs requires finding random graph models such that (i) the distance between \mathbf{A} and \mathcal{A} goes to 0 as N grows; and (ii) the sparsity pattern of \mathcal{A} is retained after the operation $\arccos(\mathcal{H}\mathcal{H})$. This is highly non-trivial, but it may be doable in some cases (e.g., graphon models more general than SBMs, or non-graphon graphs with a strong clustered structure). Last but not least, application of our GNN architectures (along with the associated theoretical analysis) to problems other than graph denoising are also worth investigating.

Network topology inference with hidden nodes for multiple graphs. The growing adoption of graph-based approaches has revealed that in many datasets one needs more than one graph to describe the data. When the set of nodes does not change, multi-layer graphs are the preferred way to address this problem. However, the theoretical research in this area has been a bit slow and only recently, generalizations of graphical Lasso to the multi-layer graph case have been proposed. As a result, network topology inference algorithms for multi-layer graphs in the presence of perturbations (including hidden nodes) are mostly missing. To develop such an algorithm in the graphical Lasso case, one should: (i) consider the effect of the hidden nodes in each of the graphs as well as (ii) incorporate a term that promotes similarity among the different graphs (according to the prior information and the application at hand). The challenge to achieve the latter is how to split the similarity metric among observed nodes and hidden nodes. Moreover, the goal would be not only designing an effective algorithm, but also characterizing analytically its performance. In this sense, the works in [125, 128] provide two promising starting points for the case where the observations are either Gaussian or stationary, respectively. Another equally interesting but more challenging line of research is to look at setups where not all the nodes participate in all relations (layers of the graph).

Accounting for perturbations in the observed topology in other GSP problems. The presence of perturbations in the observed topology is critical in most GSP approaches, and hence, generalizing the ideas from Chapter 5 to other GSP problems like (blind) deconvolution, denoising, or sampling and reconstruction constitutes an interesting line of research. In this sense, while the general approach in Chapter 5 can be preserved (i.e., defining the true GSO as an explicit estimation variable, formulate a problem that solves jointly over the variables of interest and the true GSO, and link the true GSO and the variables of interest via tractable constraints), the specific formulation and algorithmic approach will depend on the problem at hand. On top of this, even for the GF identification task investigated in Chapter 5 (as well as for the generalizations suggested above), one can enhance the problem formulation by incorporating additional information of the GSO and/or its perturbations. A first avenue to pursue is considering alternative types of link perturbations as well as the presence of hidden nodes. A second avenue is to consider additional information about the true GSO (either in the form of a statistical prior or, e.g., having access to other graphs that we know to belong to the same class/family than the true one). The

previous discussion illustrates that, by strengthening the estimation/denoising of GSO in the formulation, the GSP paradigm shifts from a two-step approach where first one observes/estimates a graph and then uses the graph to solve SP tasks to an encompassing methodology where the estimation/denoising of graph and the GSP task of interest are solved jointly.

Exploiting prior information about the graph topology. When dealing with a perturbed GSO $\bar{\mathbf{S}}$, we have assumed that the magnitude of the perturbations was small, so that we could augment our formulation with a term that promotes similarity between \mathbf{S} and $\bar{\mathbf{S}}$. However, there may be practical settings where leveraging only this information may not be enough. Examples include setups where the magnitude of the perturbations in $\bar{\mathbf{S}}$ is large, or when $\bar{\mathbf{S}}$ corresponds to a (possibly perturbed) subgraph of the whole \mathcal{G} , so there is no information about the topology of the remaining graph. In these cases, having access to prior information about the underlying graph is key to obtaining an accurate estimate of \mathbf{S} that can be exploited. As briefly pointed out in this chapter, two potential avenues to achieve this are: (i) viewing the graph as a realization of a random graph model and incorporating some (tractable) statistical prior to the formulation; and (ii) assuming that we have access to other graphs that are similar / related / belong to the same family as the graph at hand. Regarding (i), postulation of meaningful and tractable probabilistic graphs is an entire area of research, so that the efforts there should be on identifying models that are particularly suited for the (vertex based, spectral based, polynomial based...) approach exploited by the GSP task at hand. Regarding (ii), our work in [73] (which finds a graph with a density of motifs similar to that of another given graph and is able to deal with graphs with different numbers of nodes) provides an early example of how to achieve that. Intuitively, prior information of this sort could be included in our robust GSP approaches to enhance the resilience of the algorithms to perturbations and, moreover, the proposed similarity metric can be used in joint (multi-layer) network topology inference problems even when the sought graphs have different sizes. Although interesting, employing this prior information about the density of motifs is a challenging task, because the similarity metric from [73] is formulated in the spectral domain. Since most of the algorithms presented in this thesis were formulated in the node domain, they will need to be carefully redesigned to obtain convex solutions exploiting the similarity of motifs.

The lines above represent a few examples of open problems that can be addressed using the results of this thesis as starting point. There are a number of emerging areas in GSP (tensor models, time-varying graph signals, link-based signals and GSOs, categorical graph signals, Montecarlo graph and graph-signal based estimation schemes...) that are at their infancy and, as a result, are currently ignoring the effect of perturbations. Carefully updating and coming up with new formulations to render those novel schemes robust to perturbations in the data and the supporting graph will certainly be a problem of interest that will receive substantial attention in future years.

Bibliography

- [1] D. Easley and J. Kleinberg, *Networks, crowds, and markets: Reasoning about a highly connected world*. Cambridge University Press, 2010.
- [2] M. E. J. Newman, "The structure and function of complex networks," *SIAM rev.*, vol. 45, no. 2, pp. 167–256, 2003.
- [3] R. Castro, M. Coates, G. Liang, R. Nowak, and B. Yu, "Network tomography: Recent developments," *Statistical Science*, vol. 19, no. 3, pp. 499–517, 2004.
- [4] J. Leskovec, A. Rajaraman, and J. D. Ullman, *Mining of massive data sets*. Cambridge University Press, 2020.
- [5] K. Nodop, R. Connolly, and F. Girardi, "The field campaigns of the european tracer experiment (etex): Overview and results," *Atmospheric Environment*, vol. 32, no. 24, pp. 4095–4108, 1998.
- [6] E. D. Kolaczyk, *Statistical Analysis of Network Data: Methods and Models*. New York, NY: Springer, 2009.
- [7] O. Sporns, *Discovering the Human Connectome*. Boston, MA: MIT Press, 2012.
- [8] D. Shuman, S. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains," *IEEE Signal Process. Mag.*, vol. 30, no. 3, pp. 83–98, 2013.
- [9] P. Djuric and C. Richard, *Cooperative and Graph Signal Processing: Principles and Applications*. Academic Press, 2018.
- [10] A. Ortega, P. Frossard, J. Kovačević, J. M. F. Moura, and P. Vandergheynst, "Graph signal processing: Overview, challenges, and applications," *Proc. IEEE*, vol. 106, no. 5, pp. 808–828, 2018.
- [11] A. G. Marques, N. Kiyavash, J. M. F. Moura, D. V. D. Ville, and R. Willett, "Graph signal processing: Foundations and emerging directions (editorial)," *IEEE Signal Process. Mag.*, vol. 37, Nov. 2020.
- [12] A. Sandryhaila and J. M. F. Moura, "Discrete signal processing on graphs," *IEEE Trans. Signal Process.*, vol. 61, no. 7, pp. 1644–1656, 2013.
- [13] —, "Discrete signal processing on graphs: Frequency analysis," *IEEE Trans. Signal Process.*, vol. 62, no. 12, pp. 3042–3054, 2014.

- [14] E. Isufi, A. Loukas, A. Simonetto, and G. Leus, "Autoregressive moving average graph filtering," *IEEE Trans. Signal Process.*, vol. 65, no. 2, pp. 274–288, 2016.
- [15] S. Sardellitti, S. Barbarossa, and P. D. Lorenzo, "On the graph Fourier transform for directed graphs," *IEEE J. Sel. Topics Signal Process.*, vol. 11, no. 6, pp. 796–811, 2017.
- [16] S. Segarra, G. Mateos, A. G. Marques, and A. Ribeiro, "Blind identification of graph filters," *IEEE Trans. Signal Process.*, vol. 65, no. 5, pp. 1146–1159, 2017.
- [17] S. Segarra, A. G. Marques, and A. Ribeiro, "Optimal graph-filter design and applications to distributed linear network operators," *IEEE Trans. Signal Process.*, vol. 65, no. 15, pp. 4117–4131, 2017.
- [18] F. J. Iglesias, S. Segarra, S. Rey, A. G. Marques, and D. Ramírez, "Demixing and blind deconvolution of graph-diffused sparse signals," in *IEEE Int. Conf. Acoustics, Speech Signal Process. (ICASSP)*. IEEE, 2018, pp. 4189–4193.
- [19] J. Liu, E. Isufi, and G. Leus, "Filter design for autoregressive moving average graph filters," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 5, no. 1, pp. 47–60, 2018.
- [20] N. Tremblay, P. Gonçalves, and P. Borgnat, "Design of graph filters and filterbanks," in *Cooperative Graph Signal Process.* Elsevier, 2018, pp. 299–324.
- [21] S. Chen, R. Varma, A. Sandryhaila, and J. Kovačević, "Discrete signal processing on graphs: Sampling theory," *IEEE Trans. Signal Process.*, vol. 63, no. 24, pp. 6510–6523, 2015.
- [22] A. Anis, A. Gadde, and A. Ortega, "Towards a sampling theorem for signals on arbitrary graphs," in *IEEE Int. Conf. Acoustics, Speech Signal Process. (ICASSP)*. IEEE, 2014, pp. 3864–3868.
- [23] A. G. Marques, S. Segarra, G. Leus, and A. Ribeiro, "Sampling of graph signals with successive local aggregations," *IEEE Trans. Signal Process.*, vol. 64, no. 7, pp. 1832–1843, 2015.
- [24] G. Puy, N. Tremblay, R. Gribonval, and P. Vandergheynst, "Random sampling of bandlimited signals on graphs," *Appl. Comput. Harmonic Anal.*, vol. 44, no. 2, pp. 446–475, 2018.
- [25] S. Chen, A. Sandryhaila, J. M. F. Moura, and J. Kovačević, "Signal denoising on graphs via graph filtering," in *Global Conf. Signal Info. Process. (GlobalSIP)*. IEEE, 2014, pp. 872–876.
- [26] S. Chen, A. Sandryhaila, G. Lederman, Z. Wang, J. M. F. Moura, P. Rizzo, J. Bielak, J. H. Garrett, and J. Kovačević, "Signal inpainting on graphs via total variation minimization," in *IEEE Int. Conf. Acoustics, Speech Signal Process. (ICASSP)*. IEEE, 2014, pp. 8267–8271.
- [27] M. Onuki, S. Ono, M. Yamagishi, and Y. Tanaka, "Graph signal denoising via trilateral filter on graph spectral domain," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 2, no. 2, pp. 137–148, 2016.
- [28] S. Chen, A. Sandryhaila, J. M. F. Moura, and J. Kovačević, "Signal recovery on graphs: Variation minimization," *IEEE Trans. Signal Process.*, vol. 63, no. 17, pp. 4609–4624, 2015.
- [29] V. Kalofolias, "How to learn a graph from smooth signals," in *Intl. Conf. Artif. Intel. Statist. (AISTATS)*. J. Mach. Learn. Res., 2016, pp. 920–929.

-
- [30] E. Pavez and A. Ortega, "Generalized Laplacian precision matrix estimation for graph signal processing," in *IEEE Int. Conf. Acoustics, Speech Signal Process. (ICASSP)*, 2016, pp. 6350–6354.
- [31] X. Dong, D. Thanou, P. Frossard, and P. Vandergheynst, "Learning Laplacian matrix in smooth graph signal representations," *IEEE Trans. Signal Process.*, vol. 64, no. 23, pp. 6160–6173, 2016.
- [32] S. Segarra, A. G. Marques, G. Mateos, and A. Ribeiro, "Network topology inference from spectral templates," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 3, no. 3, pp. 467–483, 2017.
- [33] S. Segarra, A. G. Marques, M. Goyal, and S. Rey, "Network topology inference from input-output diffusion pairs," in *IEEE Wrkshp. Statistical Signal Process. (SSP)*. IEEE, 2018, pp. 508–512.
- [34] G. Mateos, S. Segarra, A. G. Marques, and A. Ribeiro, "Connecting the dots: Identifying network structure via graph signal processing," *IEEE Signal Process. Mag.*, vol. 36, no. 3, pp. 16–43, 2019.
- [35] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.
- [36] F. Gama, A. G. Marques, G. Leus, and A. Ribeiro, "Convolutional neural network architectures for signals supported on graphs," *IEEE Trans. Signal Process.*, vol. 67, no. 4, pp. 1034–1049, 2019.
- [37] S. Sakhavi, C. Guan, and S. Yan, "Learning temporal information for brain-computer interface using convolutional neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 11, pp. 5619–5629, 2018.
- [38] Z. Cui, K. Henrickson, R. Ke, and Y. Wang, "Traffic graph convolutional recurrent neural network: A deep learning framework for network-scale traffic learning and forecasting," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 11, pp. 4883–4894, 2019.
- [39] H. Wang, J. Wang, J. Wang, M. Zhao, W. Zhang, F. Zhang, X. Xie, and M. Guo, "GraphGAN: graph representation learning with generative adversarial nets," in *AAAI Conf. Artificial Intell.*, vol. 32, 2018.
- [40] P. Giménez-Febrer and A. Pages-Zamora, "Matrix completion of noisy graph signals via proximal gradient minimization," in *IEEE Int. Conf. Acoustics, Speech Signal Process. (ICASSP)*. IEEE, 2017, pp. 4441–4445.
- [41] Y. Wang, J. Sharpnack, A. Smola, and R. Tibshirani, "Trend filtering on graphs," in *Artif. Intell. Statistics*. PMLR, 2015, pp. 1042–1050.
- [42] J. Pang and G. Cheung, "Graph Laplacian regularization for image denoising: Analysis in the continuous domain," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 26, no. 4, pp. 1770–1785, 2017.
- [43] D. B. Tay and J. Jiang, "Time-varying graph signal denoising via median filters," *IEEE Trans. Circuits Syst., II, Exp. Briefs*, 2020.
- [44] T. H. Do, D. M. Nguyen, and N. Deligiannis, "Graph auto-encoder for graph signal denoising," in *IEEE Int. Conf. Acoustics, Speech Signal Process. (ICASSP)*, 2020, pp. 3322–3326.

- [45] S. Chen, Y. C. Eldar, and L. Zhao, "Graph unrolling networks: Interpretable neural networks for graph signal denoising," *IEEE Trans. Signal Process.*, 2021.
- [46] E. Acuna and C. Rodriguez, "The treatment of missing values and its effect on classifier accuracy," in *Classification, Clustering, Data Mining Appl.* Springer, 2004, pp. 639–647.
- [47] M. Tsitsvero, S. Barbarossa, and P. D. Lorenzo, "Signals on graphs: Uncertainty principle and sampling," *IEEE Trans. Signal Process.*, vol. 64, no. 18, pp. 4845–4860, 2016.
- [48] A. Anis, A. Gadde, and A. Ortega, "Efficient sampling set selection for bandlimited graph signals using graph spectral proxies," *IEEE Trans. Signal Process.*, vol. 64, no. 14, pp. 3775–3789, 2016.
- [49] S. Chen, R. Varma, A. Singh, and J. Kovačević, "Signal recovery on graphs: Fundamental limits of sampling strategies," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 2, no. 4, pp. 539–554, 2016.
- [50] E. Isufi, A. Loukas, A. Simonetto, and G. Leus, "Filtering random graph processes over random time-varying graphs," *IEEE Trans. Signal Process.*, vol. 65, no. 16, pp. 4406–4421, 2017.
- [51] J. Friedman, T. Hastie, and R. Tibshirani, "Sparse inverse covariance estimation with the graphical lasso," *Biostatistics*, vol. 9, no. 3, pp. 432–441, 2008.
- [52] E. Ceci and S. Barbarossa, "Graph signal processing in the presence of topology uncertainties," *IEEE Trans. Signal Process.*, vol. 68, pp. 1558–1573, 2020.
- [53] J. Miettinen, S. A. Vorobyov, and E. Ollila, "Graph error effect in graph signal processing," in *IEEE Int. Conf. Acoustics, Speech Signal Process. (ICASSP)*. IEEE, 2018, pp. 4164–4168.
- [54] —, "Modelling graph errors: Towards robust graph signal processing," *arXiv preprint arXiv:1903.08398*, 2019.
- [55] E. Ceci, Y. Shen, G. B. Giannakis, and S. Barbarossa, "Graph-based learning under perturbations via total least-squares," *IEEE Trans. Signal Process.*, vol. 68, pp. 2870–2882, 2020.
- [56] A. Natali, M. Coutino, and G. Leus, "Topology-aware joint graph filter and edge weight identification for network processes," in *IEEE Intl. Wrkshp. Mach. Learn. Signal Process. (MLSP)*. IEEE, 2020, pp. 1–6.
- [57] V. Chandrasekaran, P. A. Parrilo, and A. S. Willsky, "Latent variable graphical model selection via convex optimization," *Annu. Allerton Conf. Commun., Control, Comput.*, vol. 40, no. 4, pp. 1935–1967, 2012.
- [58] X. Yang, M. Sheng, Y. Yuan, and T. Q. S. Quek, "Network topology inference from heterogeneous incomplete graph signals," *IEEE Trans. Signal Process.*, vol. 69, pp. 314–327, 2020.
- [59] A. Anandkumar, D. Hsu, A. Javanmard, and S. Kakade, "Learning linear Bayesian networks with latent variables," in *Int. Conf. Mach. Learn. (ICML)*, 2013, pp. 249–257.
- [60] J. Mei and J. M. F. Moura, "SILVar: Single index latent variable models," *IEEE Trans. Signal Process.*, vol. 66, no. 11, pp. 2790–2803, 2018.

-
- [61] A. Chang, T. Yao, and G. I. Allen, "Graphical models and dynamic latent factors for modeling functional brain connectivity," *2019 IEEE Data Science Wrksp. (DSW)*, pp. 57–63, 2019.
- [62] S. Rey, A. G. Marques, and S. Segarra, "An underparametrized deep decoder architecture for graph signals," in *IEEE Intl. Wrksp. Computat. Advances Multi-Sensor Adaptive Process. (CAMSAP)*. IEEE, 2019, pp. 231–235.
- [63] S. Rey, S. Segarra, R. Heckel, and A. G. Marques, "Untrained graph neural networks for denoising," *arXiv preprint arXiv:2109.11700*, 2021.
- [64] S. Rey, V. M. Tenorio, S. Rozada, L. Martino, and A. G. Marques, "Deep encoder-decoder neural network architectures for graph output signals," in *Conf. Signals, Syst., Computers (Asilomar)*. IEEE, 2019, pp. 225–229.
- [65] S. Rey, V. M. Tenorio, S. Rozada, L. Martino, and A. G. Marques, "Overparametrized deep encoder-decoder schemes for inputs and outputs defined over graphs," in *European Signal Process. Conf. (EUSIPCO)*. IEEE, 2021, pp. 855–859.
- [66] S. Rey, F. J. I. Garcia, C. Cabrera, and A. G. Marques, "Sampling and reconstruction of diffused sparse graph signals from successive local aggregations," *IEEE Signal Process. Lett.*, vol. 26, no. 8, pp. 1142–1146, 2019.
- [67] S. Rey and A. G. Marques, "Robust graph-filter identification with graph denoising regularization," in *IEEE Int. Conf. Acoustics, Speech Signal Process. (ICASSP)*. IEEE, 2021, pp. 5300–5304.
- [68] S. Rey, V. M. Tenorio, and A. G. Marques, "Robust graph filter identification and graph denoising from signal observations," *arXiv preprint arXiv:2210.08488*, 2022.
- [69] V. M. Tenorio, S. Rey, F. Gama, S. Segarra, and A. G. Marques, "A robust alternative for graph convolutional neural networks via graph neighborhood filters," in *Conf. Signals, Syst., Computers (Asilomar)*. IEEE, 2021, pp. 1573–1578.
- [70] A. Buciualea, S. Rey, C. Cabrera, and A. G. Marques, "Network reconstruction from graph-stationary signals with hidden variables," in *Conf. Signals, Syst., Computers (Asilomar)*. IEEE, 2019, pp. 56–60.
- [71] S. Rey, A. Buciualea, M. Navarro, S. Segarra, and A. G. Marques, "Joint inference of multiple graphs with hidden variables from stationary graph signals," in *IEEE Int. Conf. Acoustics, Speech Signal Process. (ICASSP)*. IEEE, 2022, pp. 5817–5821.
- [72] N. Simon, J. Friedman, T. Hastie, and R. Tibshirani, "A sparse-group lasso," *J. Comput. Graphical Statist.*, vol. 22, no. 2, pp. 231–245, 2013.
- [73] S. Rey, T. M. Roddenberry, S. Segarra, and A. G. Marques, "Enhanced graph-learning schemes driven by similar distributions of motifs," *arXiv preprint arXiv:2207.04747*, 2022.
- [74] A. G. Marques, S. Segarra, G. Leus, and A. Ribeiro, "Stationary graph processes and spectral estimation," *IEEE Trans. Signal Process.*, vol. 65, no. 22, pp. 5911–5926, 2017.
- [75] M. Yuan and Y. Lin, "Model selection and estimation in the Gaussian graphical model," *Biometrika*, vol. 94, no. 1, pp. 19–35, 2007.
- [76] D. Ulyanov, A. Vedaldi, and V. Lempitsky, "Deep image prior," in *IEEE Conf. Comput. Vision Pattern Recog.*, 2018, pp. 9446–9454.

- [77] R. Heckel and P. Hand, "Deep decoder: Concise image representations from untrained non-convolutional networks," in *Intl. Conf. Learn. Repr.*, 2018.
- [78] A. K. Jain and R. C. Dubes, *Algorithms for Clustering Data*. Prentice Hall Englewood Cliffs, NJ, 1988, vol. 6.
- [79] G. Carlsson, F. Memoli, A. Ribeiro, and S. Segarra, "Hierarchical clustering of asymmetric networks," *Adv. Data Anal. Classification*, vol. 12, no. 1, pp. 65–105, Mar 2018.
- [80] G. Carlsson, F. Mémoli, A. Ribeiro, and S. Segarra, "Axiomatic construction of hierarchical clustering in asymmetric networks," in *IEEE Int. Conf. Acoustics, Speech Signal Process. (ICASSP)*, 2013, pp. 5219–5223.
- [81] D. Chen, Y. Lin, W. Li, P. Li, J. Zhou, and X. Sun, "Measuring and relieving the over-smoothing problem for graph neural networks from the topological view," in *AAAI Conf. Artificial Intell.*, vol. 34, no. 04, 2020, pp. 3438–3445.
- [82] R. Heckel and M. Soltanolkotabi, "Denoising and regularization via exploiting the structural bias of convolutional generators," *arXiv preprint arXiv:1910.14634*, 2019.
- [83] A. Daniely, R. Frostig, and Y. Singer, "Toward deeper understanding of neural networks: The power of initialization and a dual view on expressivity," in *Adv. Neural Inf. Proc. Syst.*, 2016, pp. 2253–2261.
- [84] M. Newman, *Networks*. Oxford University Press, 2018.
- [85] M. T. Schaub, S. Segarra, and J. N. Tsitsiklis, "Blind identification of stochastic block models from dynamical observations," *SIAM J. Math. Data Sc.*, vol. 2, no. 2, pp. 335–367, 2020.
- [86] D. J. Watts and S. H. Strogatz, "Collective dynamics of 'small-world' networks," *Nature*, vol. 393, no. 6684, pp. 440–442, 1998.
- [87] W. W. Zachary, "An information flow model for conflict and fission in small groups," *J. Anthropol. Res.*, vol. 33, no. 4, pp. 452–473, 1977.
- [88] D. J. Watts, "Networks, dynamics, and the small-world phenomenon," *Amer. J. Sociology*, vol. 105, no. 2, pp. 493–527, 1999.
- [89] P. Holme and B. J. Kim, "Growing scale-free networks with tunable clustering," *Physical review E*, vol. 65, no. 2, p. 026107, 2002.
- [90] S. Segarra, A. G. Marques, G. R. Arce, and A. Ribeiro, "Design of weighted median graph filters," in *IEEE Intl. Wrksp. Computat. Advances Multi-Sensor Adaptive Process. (CAMSAP)*, 2017, pp. 1–5.
- [91] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," *arXiv preprint arXiv:1710.10903*, 2017.
- [92] F. Dorfler and F. Bullo, "Kron reduction of graphs with applications to electrical networks," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 60, no. 1, pp. 150–163, 2012.
- [93] J. Cardoso, V. D. M., J. Ying, and D. P. Palomar, "Algorithms for learning graphs in financial markets," *arXiv preprint arXiv:2012.15410*, 2020.

-
- [94] Y. Yu, T. Wang, and R. J. Samworth, "A useful variant of the Davis–Kahan theorem for statisticians," *Biometrika*, vol. 102, no. 2, pp. 315–323, 2015.
- [95] D. Ramirez, A. G. Marques, and S. Segarra, "Graph-signal reconstruction and blind deconvolution for diffused sparse inputs," in *IEEE Int. Conf. Acoustics, Speech Signal Process. (ICASSP)*, March 2017, pp. 4104–4108.
- [96] S. M. Kay, *Fundamentals of Statistical Signal Processing: Estimation Theory*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1993.
- [97] F. Pukelsheim, *Optimal Design of Experiments*. SIAM, 1993, vol. 50.
- [98] L. F. O. Chamon and A. Ribeiro, "Greedy sampling of graph signals," *IEEE Trans. Signal Process.*, vol. 66, no. 1, pp. 34–47, Jan. 2018.
- [99] D. L. Donoho and M. Elad, "Optimally sparse representation in general (nonorthogonal) dictionaries via ℓ^1 minimization," *Proc. Nat. Academy Sciences*, vol. 100, no. 5, pp. 2197–2202, 2003.
- [100] A. Decelle, F. Krzakala, C. Moore, and L. Zdeborová, "Asymptotic analysis of the stochastic block model for modular networks and its algorithmic applications," *Physical Review E*, vol. 84, no. 6, p. 066106, Dec. 2011.
- [101] P. D. Dobson and A. J. Doig, "Distinguishing enzyme structures from non-enzymes without alignments," *J. molecular biology*, vol. 330, no. 4, pp. 771–783, 2003.
- [102] D. Thanou, X. Dong, D. Kressner, and P. Frossard, "Learning heat diffusion graphs," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 3, no. 3, pp. 484–499, 2017.
- [103] Y. Zhu, F. J. Iglesias-Garcia, A. G. Marques, and S. Segarra, "Estimating network processes via blind identification of multiple graph filters," *IEEE Trans. Signal Process.*, vol. 68, pp. 3049–3063, 2020.
- [104] Y. He and H. Wai, "Detecting central nodes from low-rank excited graph signals via structured factor analysis," *IEEE Trans. Signal Process.*, 2022.
- [105] R. Levie, E. Isufi, and G. Kutyniok, "On the transferability of spectral graph filters," in *Int. Conf. Sampling Theory Appl. (SampTA)*. IEEE, 2019, pp. 1–5.
- [106] R. Levie et al., "Transferability of spectral graph convolutional neural networks," *J. Mach. Learn. Res.*, vol. 22, pp. 272–1, 2021.
- [107] L. Ruiz, F. Gama, and A. Ribeiro, "Graph neural networks: Architectures, stability, and transferability," *Proc. IEEE*, vol. 109, no. 5, pp. 660–682, 2021.
- [108] H. Zou and T. Hastie, "Regularization and variable selection via the elastic net," *J. Roy. Statistical Soc.: Ser. B (Statistical Methodology)*, vol. 67, no. 2, pp. 301–320, 2005.
- [109] B. Dai, S. Ding, and G. Wahba, "Multivariate Bernoulli distribution," *Bernoulli*, vol. 19, no. 4, pp. 1465–1483, 2013.
- [110] S. Segarra and A. Ribeiro, "Stability and continuity of centrality measures in weighted graphs," *IEEE Trans. Signal Process.*, vol. 64, no. 3, pp. 543–555, 2015.
- [111] E. J. Candes, M. B. Wakin, and S. Boyd, "Enhancing sparsity by reweighted ℓ_1 minimization," *J. Fourier Anal. Appl.*, vol. 14, no. 5-6, pp. 877–905, 2008.

- [112] Y. Sun, P. Babu, and D. P. Palomar, "Majorization-minimization algorithms in signal processing, communications, and machine learning," *IEEE Trans. Signal Process.*, vol. 65, no. 3, pp. 794–816, 2016.
- [113] J. Gorski, F. Pfeuffer, and K. Klamroth, "Biconvex sets and optimization with biconvex functions: A survey and extensions," *Math. Methods Oper. Res.*, vol. 66, no. 3, pp. 373–407, 2007.
- [114] M. Hong, M. Razaviyayn, Z. Luo, and J. Pang, "A unified algorithmic framework for block-structured optimization involving big data: With applications in machine learning and signal processing," *IEEE Signal Process. Mag.*, vol. 33, no. 1, pp. 57–77, 2015.
- [115] R. Shafipour and G. Mateos, "Online topology inference from streaming stationary graph signals with partial connectivity information," *Algorithms*, vol. 13, no. 9, p. 228, 2020.
- [116] A. Buciualea, S. Rey, and A. G. Marques, "Learning graphs from smooth and graph-stationary signals with hidden variables," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 8, pp. 273–287, 2022.
- [117] G. C. Reinsel, *Elements of multivariate time series analysis*. Springer Science & Business Media, 2003.
- [118] J. Mei and J. M. F. Moura, "Signal processing on graphs: Causal modeling of unstructured data," *IEEE Trans. Signal Process.*, vol. 65, no. 8, pp. 2077–2092, 2017.
- [119] E. Isufi, A. Loukas, N. Perraudin, and G. Leus, "Forecasting time series with VARMA recursions on graphs," *IEEE Trans. Signal Process.*, vol. 67, no. 18, pp. 4870–4885, 2019.
- [120] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.
- [121] D. Coppersmith and S. Winograd, "Matrix multiplication via arithmetic progressions," in *Proc. ACM Symp. Theory Comput.*, 1987, pp. 1–6.
- [122] T. Hastie, R. Tibshirani, and M. Wainwright, "Statistical learning with sparsity," *Monographs Statistics Appl. Probability*, vol. 143, p. 143, 2015.
- [123] Y. Murase, J. Török, H. H. Jo, K. Kaski, and J. Kertész, "Multilayer weighted social network model," *Physical Review E*, vol. 90, no. 5, p. 052810, 2014.
- [124] P. Danaher, P. Wang, and D. M. Witten, "The joint graphical lasso for inverse covariance estimation across multiple classes," *J. Roy. Statistical Soc.: Ser. B (Statistical Methodology)*, vol. 76, no. 2, pp. 373–397, 2014.
- [125] M. Navarro, Y. Wang, A. G. Marques, C. Uhler, and S. Segarra, "Joint inference of multiple graphs from matrix polynomials," *J. Mach. Learn. Res.*, 2020.
- [126] J. Arroyo, A. Athreya, J. Cape, G. Chen, C. E. Priebe, and J. T. Vogelstein, "Inference for multiple heterogeneous networks with a common invariant subspace," *J. Mach. Learn. Res.*, vol. 22, no. 142, pp. 1–49, 2021.
- [127] 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.
- [128] P. Ravikumar, M. J. Wainwright, G. Raskutti, and B. Yu, "High-dimensional covariance estimation by minimizing ℓ_1 -penalized log-determinant divergence," *Electron. J. Statistics*, vol. 5, pp. 935–980, 2011.