Die approbierte Originalversion dieser Dissertation ist in der Hauptbibliothek der Technischen Universität Wien aufgestellt und zugänglich.
http://www.ub.tuwien.ac.at

The approved original version of this thesis is available at the main library of the Vienna University of Technology.
http://www.ub.tuwien.ac.at/eng

Smooth Graph Signal Processing: Recovery and Sampling Schemes

Gita Babazadeh Eslamlou
Matrikelnummer: 1329221

Fakultät für Elektrotechnik und Informationstechnik
Technische Universität Wien

Ausgeführt zum Zwecke der Erlangung des akademischen Grades
eines Doktors der technischen Wissenschaften.

February 2018

Advisor

Univ.-Prof. Dr.-Ing. Norbert Görtz
Institute of Telecommunications
Technische Universität Wien
Österreich

Abstract

The amounts of data collected by automated software and hardware in various domains, such as social networks, bioinformatics and wireless sensor networks, are exploding. Beside the sheer *volume* of these data-sets also the *high velocity* (rate of generation) and their *variety* (data composed of mixture of audio video text, only partially labeled) pose big challenges on their processing. A particular useful methodology to cope with big data is provided by graph signal processing (GSP), which models data-sets as signals defined over large graphs (complex networks). The usage of graph models within GSP entails efficient distributed message passing algorithms that are well suited to deal with large volumes of high-speed data. Moreover, graphs allow to organize heterogeneous data by exploiting application specific notions of similarity, thereby addressing the variety of big data. A key problem studied in GSP is the recovery of a graph signal from its noisy samples at few selected nodes. This problem is relevant, e.g., for semi-supervised learning over graphs, where only few training examples (represented by graph nodes) are labeled and most examples are unlabeled. The problem of determining the labels for the unlabeled data is precisely a graph signal recovery problem. The recovery is feasible for the graph signals which are smooth with respect to the graph.

In this work, we investigate the problem of recovering a graph signal from the noisy samples observed at a small number of randomly selected nodes. The signal recovery is formulated as a convex optimization problem. Our approaches exploit the smoothness of typical graph signals occurring in many applications, such as wireless sensor networks or social network analysis. The graph signals are smooth in the sense that the neighboring nodes have similar signal values. In particular, we propose various graph signal recovery methods, which are shown to be particularly well suited for smooth graph signal recovery. Besides, in this dissertation we present a novel and flexible sampling method for signals that are supported on either directed or undirected graphs. The proposed sampling algorithm selects the optimal sampling set from the set of arbitrary weighted graph signal, for any predefined sampling rate, such that the reconstruction (recovery) quality is as high as possible. The algorithm is able to be adaptively adjusted based on the structure of the underlying graph and signal model such as nodes degree and smoothness.

The effectiveness of the proposed recovery and sampling methods is verified by numerical experiments on various random graphs along with a real-world data-set. Numerical evaluations show that the proposed algorithms render a highly efficient performance compared to the state-of-the-art methods.

Declaration

I hereby declare that, except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification at this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the references and acknowledgements.

Gita Babazadeh Eslamlou
Feburary 2018

Table of Contents

Chapter 1

Introduction

During the last couple of years, the massive amount of data and information has exceeded even the most aggressive predictions. Due to the ubiquitous use of the Internet over the new devices and the rapid adoption of novel technologies and applications such as Internet of things (IoT) and social networks, this trend is likely to persist. Such data has generally high complexity in different aspects. First of all, it is extremely high-dimensional which means that it needs a large volume of storage space. Second, the data is usually irregularly structured. For example, in the applications such as wireless sensor networks, the senors are irregularly deployed in the network environment and their collected data highly depends on their geographical positions. Finally, the structure of data is not necessarily obtained from single source of information. For instance, a wide range of entities may affect the data and information extracted from social networks. Hence, in order to represent, compress, recover and analyze such data and efficient usage of its structure, we need the development of the new tools and applications.

Graphs have long been adopted in a broad variety of applications, such as analysis of social networks, machine learning, network protocol optimization, or image processing and have the ability to model such data and their complex interactions. For instance users in the social networks such as Facebook can be considered as the graph nodes and their friendship relationships as the graph edges. Another example is given by 2D images where each pixel corresponds to a node in a grid graph connecting the nearest pixels with each other. A quiet popular approach to handle this data is adding attributes to such nodes and modeling those nodes as signals on a graph. Techniques based on spectral graph theory exploit Fourier transform, filtering and frequency response of the graph to perform a frequency interpretation of graph data. The field which encompasses all these, is commonly known as graph signal processing (GSP) [1]. GSP aims to exploit the well formed tools for representation and

analysis of conventional signals to the new field of signal processing on graphs while using the underlying connectivity information.

One of the important features appears in the signal processing on graphs is smoothness. Smoothness means that the signal coefficients associated with the two end vertices of edges take the similar values. Smoothness appears in many real-world applications. For example, consider the graph signal extracted from social network such as Facebook when the signals on the graph are defined as personal interests. It is generally the case which friends, represented by connected vertices, have similar interests. In GSP literature, different mathematical formulations of graph smoothness are adopted. In this dissertation, we use a widely accepted definition of smoothness which is based on some norm. Smoothness imposes a desirable signal property which can be exploited in many problems involving sampling [2], along with graphs regularization [3] for signal recovery and classification [4].

Sampling and recovery are of immense importance in conventional signal processing, which provide a link between the analog and discrete time signals [5, 6]. Sampling algorithms are a bunch of techniques which make a sequence from a function. Conversely, recovery algorithms provide a function from a sequence. The procedure of sampling a function and its recovery make the foundation of digital signal processing.

One of the main goals of graph signal processing is to derive sampling theorems which are important for the design of graph signal processing systems. Graph signal sampling is known as the mechanism of reducing a graph signal to a small number of measurements. For instance, in the graph signal extracted from Facebook, we select a fraction of users query their interests and then recover the interest of the rest of the users. The questions arise: how many and which users should be selected to have precise prediction of the unsampled users interests. The task of efficient graph signals sampling is, however, not well understood. By paralleling the Nyquist-Shannon theory for the band-limited signals (in the sense of having all signals energy confined to a finite interval in frequency domain), the authors of [7] define notions of (approximately) band-limitedness for the graph signals. For a given bandwidth, they also construct sampling sets of minimum size which guarantee perfect recovery. The real-world signals, however, may not be always band-limited [8]. It is also possible to define the notion of sparse graph signals and apply concepts of compressed sensing (CS) to the graph structured signals. A first application of CS to the graph signals is based on the graph Fourier transform (GFT) which is composed of the eigenvectors of the graph Laplacian. Assuming that the graph signal is sparse in the GFT domain, the authors of [9] propose to subsample the graph signal according to CS theory. However, in this dissertation we consider a notion of sparsity different from [9]. In particular, we assume that the graph signal of interest is sparse in the sense of consisting of few clusters within which the signal is approximately constant.

Another important problem considered in GSP literature is the recovery of graph signals from noisy and incomplete measurements. In particular, we are interested in the reconstruction of a graph signal from a small number of sample values taken on a subset of nodes. The samples of graph signal are generally assumed to be contaminated by additive noise, which captures measurement and modeling errors. The recovery algorithms are based on the assumption of graph signal smoothness with respect to the graph structure. This notion of graph signal smoothness can be interpreted as a constraint on the regularization of the graph signal. Our goal is to reconstruct graph signals under a regularization constraint, i.e., total variation or Tikhonov regularization.

In this dissertation, we pursue our above mentioned goal by addressing, in a mathematically rigorous manner, the following fundamental question: How to properly sample and recover the graph signals?

1.1 Outline

This dissertation is structured as follows. Chapter 2 provides a background on signal processing on graph and compressed sensing. It reviews the existing literature on GSP and CS. The next part of the dissertation, spanning Chapters 3 and 4 present some efficient recovery algorithms on smooth graph signals. In these sections we address both Tikhonov and total variation regularizations. Chapter 5 complements the work by presenting an adaptive sampling algorithm which selects a subset of graph nodes based on the signal and graph structure. The remainder of this chapter provides a short abstract of each chapter and exclusively refers to the papers, which were published by the author.

Chapter 2: This chapter provides a tight definition of GSP and CS in a way that makes the concepts of this dissertation more accessible. It introduces the state-of-the-art methods on sampling and recovery of graph signals. Particular light is shed on approximate message passing (AMP)-based recovery algorithms.

Chapter 3: For discrete time signals, total variation based denoising has been considered in [10], which applies the AMP framework for denoising structured signals. Moreover, the authors of [11] present a widely applicable framework, termed denoising-based approximate message passing (DAMP). This framework is based on combining a given denoiser functions, tailored to a specific signal model, with the AMP rationale of iteratively recovering signals from incomplete random measurements. However, to the best of our knowledge, the use of the DAMP framework for graph signal denoising employing a total variation constraint is novel. Referring to my work in [12], we formalize the DAMP-based graph signal recovery problem via total variation regularization (TV-GSDAMP) and Tikhonov regularization

(Tik-GSDAMP). We show that despite the TV-GSDAMP outperforms the Tik-GSDAMP in terms of recovery performance, it has higher time complexity than Tik-GSDAMP. We apply our recovery method to a set of graph signals including real-world data-set obtained from product rating of a large online retailer.

Chapter 4: Based on our publications in [13, 14] , we formulate the graph signal recovery problem as an optimization problem using Tikhonov regularization to enforce smoothness of the recovered signal. The optimization problem produces a signal balancing two terms: the empirical error, i.e., the deviation of the recovered signal from the observed noisy samples, and the signal smoothness as measured by the graph Laplacian quadratic form. These terms are called fidelity term and smoothness term, respectively. A certain scalar known as the regularization parameter plays a crucial role in controlling the trade-off between fidelity to the data and smoothness of the solution. The optimal signal is characterized by a system of linear equations, which we solve using an iterative Gauss-Seidel (GS) method. We relate the convergence properties of this iterative method to the choice of the sampling set and the graph topology. In order to improve the time complexity of this iterative recovery method, we propose to use block Gauss-Seidel (BGS) which is a variation of Gauss-Seidel (GS) algorithm. The BGS method generalizes the GS algorithm by updating during each iteration whole blocks of the current estimate in one step instead of single entries. Furthermore, we show the optimal setting of regularization parameter is highly influenced by signal structure along with measurement and modeling noise. A very interesting outcome is that the iterative BGS recovery algorithm provides a tight approximation for optimal solution of Tikhonov regularization. Besides, we show that the proposed iterative schema shows better results in terms of recovery performance and time complexity compared to the Tik-GSDAMP algorithm.

Chapter 5: This chapter is based on our work in [15], we extend the theory of graph signal sampling by developing a fast and efficient algorithm for selecting the sampling set of an arbitrary graph signal. The sampling theory deals with measuring a graph signal on a reduced set of nodes with conditions under which the signal has a stable reconstruction. Our goal is to select the minimum number of nodes in a way that it yields the reliable reconstruction of a signal. The process of graph signal sampling is highly dependent on the structure of the graph signal. The smoothness factor, which is defined generally in terms of the signal's Fourier transform, is one of the main players in selecting the efficient sampling set of a graph signal. In a smooth graph signal, the signal values of neighbouring nodes are similar. Hence, the sampling algorithm should be properly adjusted to avoid having a bulk of samples in a certain part of the graph. The degree of the nodes in the graph is another factor which affects the sampling algorithm. In order to obtain the above mentioned goal with

respect to the influence of smoothness and node degree variance we propose a new adaptive graph signal sampling (AGSS) algorithm to sample the nodes such that the recovery error is minimized. We confirm the performance of proposed sampling algorithm by conducting illustrative numerical experiments on various graphs.

Chapter 6: In this chapter we summarize our contributions, discuss open issues and possible directions for future research.

Chapter 2

Background

2.1 Introduction

In this chapter, we discuss about some of the important concepts that are employed throughout this dissertation. We review fundamental GSP methods from the literature, which are related to the problems studied in this dissertation. We give the primary definitions and notations for graphs and signals on graphs, which are used. Then, we present the basics of CS, along with briefly reviewing the AMP framework and its corresponding algorithms such as DAMP, generalized approximate message passing (GAMP), and GAMP for cosparse analysis (GrAMPA) algorithms as well as the efficient fused lasso algorithm (EFLA) that effectively recover signal $\mathbf{x}$ from the noisy measurements. It should be noted that, our main focus resides on presenting the recovery and sampling algorithms which allow us to use the prior knowledge that highly improves the performance.

2.2 Signal processing on graphs

In the last few years, the research in the field of signal processing on graphs has been motivated and influenced multiple disciplines [16–18]. In order to represent the signal processing on graphs efficiently, one requires to consider the intrinsic geometric structure of the underlying graph. Signal models like smoothness depend on the irregular structure of the underlying graph, i.e., the graph on which the signal resides. Therefore, the classical signal processing methods which have been designed for regular signal structures are not suitable for the irregular structures of the graph. Recently, there has been a lot of effort dedicated to designing new algorithms and methods for efficiently handling the new challenges arising from the irregular structure of graphs [19, 20]. Signal processing on graphs has multiple

popular application areas, including approximation, sampling, classification, inpainting, and clustering of signals on graphs [21–26]. We provide an overview of recent work in this area.

2.2.1 Graph signals fundamental definitions

Here, we briefly mention basic definitions for graph signals. Besides, we will recall and add some extra definitions in the system setup section of forthcoming chapters. We generally consider an undirected weighted graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{W})$, with node set $\mathcal{V}$ and edge set $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$. The weights matrix is represented by $\mathbf{W} \in \mathbb{R}^{N \times N}$, where all of its components are positive; $W_{lj} = 0$ if there is no edge between vertices l and j. In particular, $W_{lj} \neq 0$ only if $(l, j) \in \mathcal{E}$. A sample generic graph signal is shown in Figure 2.1.

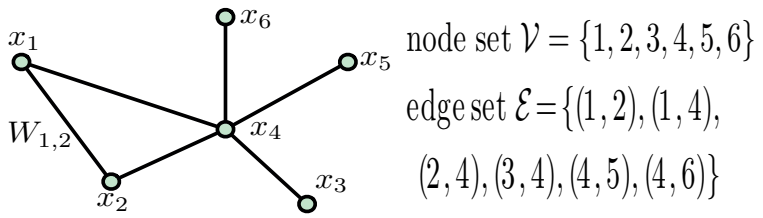


Figure 2.1. Generic graph signal with vertex set $\mathcal{V}$ and edge set $\mathcal{E}$.

We assume that our graph is connected and it consists of N nodes. The δ_{lj} is the distance metric which shows the number of edges in a shortest path (also called a graph geodesic) connecting node l and j. The k-step neighbourhood $\mathcal{N}_l^k = \{j \in \mathcal{V} : \delta_{lj} = k\}$ of the node l is the set of all nodes which are at distance k from the node l. The combinatorial graph Laplacian matrix $\mathbf{L}$ is defined as [19]

$$\mathbf{L} := \mathbf{D} - \mathbf{W}, \tag{2.1}$$

where the degree matrix $\mathbf{D}$ is a diagonal matrix whose ℓ^{th} diagonal element D_ℓ is equal to the sum of the weights of all the edges connected to vertex ℓ, i.e.,

$$D_\ell = \sum_{j \in \mathcal{V}} W_{\ell j}. \tag{2.2}$$

It is a positive semi-definite matrix that has a complete set of real orthonormal eigenvectors, with corresponding nonnegative eigenvalues [27].

For a connected graph, the normalized graph Laplacian is closely related to the combinatorial Laplacian and is defined as

$$\mathcal{L} = \mathbf{D}^{-\frac{1}{2}}\mathbf{L}\mathbf{D}^{-\frac{1}{2}} \tag{2.3}$$

$$= \mathbf{I} - \mathbf{D}^{-\frac{1}{2}}\mathbf{W}\mathbf{D}^{-\frac{1}{2}}, \tag{2.4}$$

where $\mathbf{I}$ is the identity matrix. The combinatorial and the normalized graph Laplacians are both examples of generalized graph Laplacians [28–30] and they are both popular in many graph related applications. Generally, when the graph is regular or semi-regular, the combinatorial Laplacian and the normalized Laplacian have almost identical spectra. But the combinatorial Laplacian has been widely used in the literature, specially in applications that have a random walk, or in applications where weighting vertices by their degrees is more natural. For these reasons, in this dissertation, we use the combinatorial graph Laplacian. We focus only on weighted undirected graphs. For the sake of completeness though, we note that the definition of the Laplacian can be easily extended to directed graphs [31].

2.2.2 Signal representation on graphs

The fundamental similarity between conventional signal processing and graph signal processing is established using spectral graph theory [27]. The classical Fourier transform is generalized to graph settings by using the eigenvectors and the eigenvalues of the graph Laplacian matrix which defines a notion of frequency for graph signals [32]. In particular, the graph Laplacian eigenvectors associated with small eigenvalues can be associated with the low frequency concept. This means that if two nodes are connected with a large weighted edge, the low frequency eigenvector values at those points tend to be similar. The eigenvectors associated with small eigenvalues correspond to signals that vary slowly across the graph. However, the graph Laplacian eigenvectors associated with larger eigenvalues correspond to signals that rapidly change on the graph and have different values on neighboring nodes. Hence, in graph signals, the graph Laplacian eigenvectors characterize a Fourier basis, which can be selected as the eigenvectors of the combinatorial or the normalized graph Laplacian matrices.

A lot of researches have been dedicated to developing methods specifically designed for processing and analyzing of data in graph signals. In particular, the authors in [33] have presented a new discrete signal processing framework for structured datasets and they study the datasets directly. The framework is called discrete signal processing on graphs (DSPG); for the representation, analysis, and processing of data indexed by arbitrary graphs. DSPG extends traditional discrete signal processing theory with linear structure to datasets with

complex structure that can be represented by graphs. The authors demonstrate that, if a graph signal is sparsely represented in the spectral domain, i.e. its frequency content is dominated by few frequencies, then it can be efficiently approximated via only a few spectrum coefficients. They consider three standard orthogonal transforms: the discrete Fourier transform (DFT), discrete cosine transform (DCT), and discrete wavelet transform (DWT). Results show that, the graph Fourier transform leads to smallest errors regardless of the number of the spectrum coefficients applied for approximation [34].

In [35], a new algorithm for denoising of signals residing on arbitrary graphs is presented. The authors derive an exact closed-form solution and an approximate iterative solution based on an inverse and a standard graph filters, respectively. The closed form solution needs the inversion operation, which is expensive and numerically unstable for high-dimensional graph signals. In order to overcome this problem, an approximation of the exact solution (graph filter) is represented. Graph filter produces very similar denoising errors, which highlights its practical usefulness in data denoising. The authors evaluate the derived algorithms, apply them to denoising of measurements from temperature sensors and to combining opinions from multiple experts.

The Nyquist-Shannon theory of sampling to graph signals is extended and a cut-off frequency for all bandlimited graph signal is established in [23]. Besides, the authors provide a novel greedy algorithm to choose the smallest possible sampling set for a given bandwidth. A graph spectral compressed sensing technique is presented in [36]. This technique gathers the measurements from a random subset of nodes and then interpolates with respect to the graph Laplacian eigenbasis, leveraging ideas from compressed sensing. It requires a small portion of the whole sensor nodes to sample and transmit the data, and applies the partial graph Fourier ensemble as the sensing matrix for smooth graph signals. Based on this technique, two algorithms are developed for wireless sensor networks to deal with temporally or spatially correlated signals. Both algorithms demonstrate great improvement in saving the bandwidth resources and the energy consumption.

The authors in [7] propose a new class of graph signals, called approximately bandlimited, along with two graph recovery strategies based on the random sampling and the experimentally designed sampling. They show that for an irregular graph, given that we have an unbiased estimator for the low frequency components, the convergence rate of experimentally designed sampling algorithm outperforms that of random sampling. One of the objectives of this dissertation is exactly to design structured graph sampling algorithm, which can be efficiently implemented and used with suitable graph signal recovery algorithms.

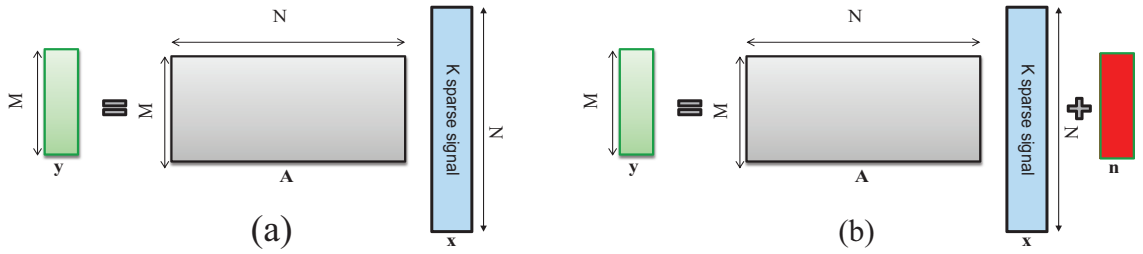


Figure 2.2. Main concept of CS.

2.3 Compressed sensing

The foundation of the term CS [37–39] lies in a particular interpretation of CS algorithms as a technique to compress the signal **x**. CS is a signal processing method to represent sparse signals in a compressive way and to fully recover sparse signals from much fewer samples than the classical Nyquist rate would suggest. Vast number of practical systems sample a signal at a rate above the Nyquist rate. They transform the signal to a basis which concentrates the signal's energy into a few large components and achieve compression by keeping these dominant coefficients. Since the signal is determined with only a few coefficients, it seems reasonable to ask if fewer measurements could have been taken in the first place. This can be a significant step to assemble the huge volumes of data from social networks, wireless sensors, online retailers and many other applications in the big data millennium.

CS exploits the signal model, like sparsity, to sample the signal more efficiently than the Shannon-Nyquist scheme. CS along with suitable recovery techniques have a huge influence on the scientific community specially by reducing the amount of collected and processed data. Although the CS field has been known for less than a decade, it has already many applications. Sum of the most typical applications of CS are image inpainting [40], magnetic resonance imaging (MRI) [41–43], lens imaging [44] and single pixel camera [45]. Besides, CS has found applications in fields like machine learning [46, 47], astronomy [48, 49], electro-magnetics [50], computational biology [51, 52], multi user detection [53–56], and radar [57, 58].

As we mentioned earliar, CS exploits the signal model, like sparsity. A signal is K-sparse if it has at most K non-zero components, where $K << N$ (N is the signals dimension). A

conventional measure to find the number of non-zero components is the ℓ_0-seminorm [1] [59]

$$\|\mathbf{x}\|_0 = \lim_{p\to 0} \|\mathbf{x}\|_p^p = \lim_{p\to 0} \sum_{n=1}^{N} |x_n|^p = |\{n : x_n \neq 0\}|. \tag{2.5}$$

In order to search for sparse solutions (2.5) is later used to constrain the problem formulation. As depicted in Figure 2.2-(a), CS typically considers linear regression problem of

$$\mathbf{y} = \mathbf{A}\mathbf{x}, \tag{2.6}$$

where $\mathbf{y} \in \mathbb{R}^M$ is the observation or the measurement vector, and $\mathbf{A} \in \mathbb{R}^{M\times N}$ is the measurement matrix. The goal in CS is to reconstruct an N-dimensional signal $\mathbf{x} \in \mathbb{R}^N$ from under-determined systems of the linear equations, i.e., from $M \ll N$ linear measurements. In the under-determined system, regarding the measurement matrix $\mathbf{A}$, there exist infinite solutions for problem (2.6). In order to solve this kind of problems the minimal ℓ_2-norm solution, i.e., the least squares utilization is a standard approach

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} \|\mathbf{x}\|_2 \quad \text{subject to } \mathbf{y} = \mathbf{A}\mathbf{x}. \tag{2.7}$$

It should be noted that the optimization problem in (2.7) does not enforce sparsity. Hence, the basic approach to enforce the sparsity is

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} \|\mathbf{x}\|_0 \quad \text{subject to } \mathbf{y} = \mathbf{A}\mathbf{x}. \tag{2.8}$$

(2.8) is infeasible to solve for a large dimension N, hence in order to make it solvable, ℓ_0-seminorm is replaced with ℓ_1-norm

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} \|\mathbf{x}\|_1 \quad \text{subject to } \mathbf{y} = \mathbf{A}\mathbf{x}. \tag{2.9}$$

This problem is commonly known as basis pursuit (BP). When the data is noisy

$$\mathbf{y} = \mathbf{A}\mathbf{x} + \mathbf{n}, \tag{2.10}$$

as shown in Figure 2.2-(b), we can relax the constraint to some degree. In (2.10), $\mathbf{A} \in \mathbb{R}^{\mathbf{M}\times\mathbf{N}}$ is the sampling or measurement matrix, which has been scaled to unit ℓ_2-column-norm $(M << N)$ and $\mathbf{n} = \{n_\ell, \ell = 1, 2, ..., M\}$, is the noise vector modeled as component-wise

[1] While $\|\cdot\|_0$ does not satisfy the homogeneity condition of the norm so it is not a norm and we call it ℓ_0-seminorm

independent additive white Gaussian noise (AWGN) with zero-mean and variance σ^2, i.e., $n_\ell \sim \mathcal{N}(0, \sigma^2)$. The least squares approach minimizes the error with respect to the ℓ_2-norm and the optimization problem is reformulate as

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} \|\mathbf{x}\|_1 \quad \text{subject to } \|\mathbf{y} - \mathbf{A}\mathbf{x}\|_2^2 \leq \epsilon, \tag{2.11}$$

where $\epsilon \geq \|\mathbf{n}\|_2^2$. This problem is called basis pursuit denoising (BPDN), which is the noise tolerable version of BP. We can reach to the least absolute shrinkage and selection operator (LASSO) formulation [60], by exchanging the constraint and the objective function of BPDN and bringing the sparsity level

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} \left(\frac{1}{2} \|\mathbf{y} - \mathbf{A}\mathbf{x}\|_2^2 + \lambda \|\mathbf{x}\|_1 \right), \tag{2.12}$$

where a Lagrangian penalty λ enforces the sparsity constraint of the solution. Iterative thresholding algorithms [61–63] are very efficient recovery methods that depend on thresholding functions. The iterative hard thresholding (IHT) [64, 65] is based on the ℓ_0-regularized problem

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} \left(\frac{1}{2} \|\mathbf{y} - \mathbf{A}\mathbf{x}\|_2^2 + \lambda \|\mathbf{x}\|_0 \right), \tag{2.13}$$

while iterative soft thresholding (IST) [66, 67] is based on the ℓ_1-regularized problem (2.12). Both IST and IHT methods provide very simple, and fast (suboptimal) solution for pursuit methods. They both alternates between steps of the form

$$\hat{\mathbf{x}}^{(t+1)} = \eta \left(\hat{\mathbf{x}}^{(t)} + \mathbf{A}^T \mathbf{z}^{(t)}; \tau^{(t)} \right), \tag{2.14}$$

$$\mathbf{z}^{(t)} = \mathbf{y} - A\hat{\mathbf{x}}^{(t)}, \tag{2.15}$$

where $\mathbf{z}^{(t)}$ is an estimate of the current residual, η is the thresholding function and $\tau^{(t)}$ is the threshold parameter which adapts during the iterations.

A generalization of LASSO which is called fused lasso [68] intends to penalize the ℓ_1-norm of both the coefficients and their successive differences. This definition contains both the sparsity of coefficients and sparsity of their differences, i.e., local constancy of the coefficient profile. It is defined as

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} \left(\frac{1}{2} \|\mathbf{y} - \mathbf{A}\mathbf{x}\|_2^2 + \lambda_1 \|\mathbf{x}\|_1 + \lambda_2 \sum_{i=1}^{N} |x_i - x_{i-1}| \right). \tag{2.16}$$

Work on CS has driven research into several classes of algorithms for under-determined linear regression, including methods in convex optimizations, heuristic techniques and different Bayesian approaches. Of particular interest in this dissertation are AMP [69–72] and the AMP-based algorithms such as DAMP [11], GAMP [73, 74], the GrAMPA [75], along with EFLA [76] algorithm, which offer the state-of-the-art performance for linear CS problems.

2.3.1 Approximate message passing (AMP)

The AMP algorithm is introduced in [69–71]. AMP is an iterative thresholding algorithm [77, 78, 63] motivated by belief propagation in graphical models. It is a message passing-based framework [79] developed for solving the basis pursuit or the basis pursuit denoising problem [80]. AMP has significantly better sparsity under-sampling trade-off compared to the other iterative thresholding algorithms such as IHT or IST techniques. The measurement vector, $\mathbf{y} \in \mathbb{R}^{\mathbf{M}}$ is assumed to be obtained via an unknown vector (true solution) $\mathbf{x} \in \mathbb{R}^{\mathbf{N}}$. According to (2.10) the sampling rate is denoted by $\delta = \frac{M}{N}$, the number of non-zero elements in $\mathbf{x}$ is denoted by k and the sparsity is shown by $\rho = \frac{K}{M}$. The first order AMP algorithm proceeds iteratively:

$$\hat{\mathbf{x}}^{(t+1)} = \eta\left(\mathbf{A}^T\mathbf{z}^{(t)} + \hat{\mathbf{x}}^{(t)}; \tau^{(t)}\right), \tag{2.17}$$

$$\mathbf{z}^{(t)} = \mathbf{y} - \mathbf{A}\hat{\mathbf{x}}^{(t)} + \frac{1}{\delta}\mathbf{z}^{(t-1)}\left\langle \eta'\left(\mathbf{A}^T\mathbf{z}^{(t-1)} + \hat{\mathbf{x}}^{(t-1)}; \tau^{(t-1)}\right)\right\rangle, \tag{2.18}$$

where, $\eta(\cdot)$ is the soft thresholding function, $\tau^{(t)}$ is the threshold parameter, $\mathbf{A^T}$ denotes the transpose of measurement matrix $\mathbf{A}$. Initially, $\hat{\mathbf{x}}^{(0)} = \mathbf{0}$, $\mathbf{z}^{(0)} = \mathbf{y}$, $\hat{\mathbf{x}}^{(t)}$ and $\mathbf{z}^{(t)}$ are the estimate of $\mathbf{x}$ and the residual at iteration t, respectively. The derivative of thresholding function is shown by η', where $< \cdot >$ is the averaging operator and the term $\frac{1}{\delta}\mathbf{z}^{(t-1)}\eta'\left(A^T\mathbf{z}^{(t-1)} + \hat{\mathbf{x}}^{(t-1)}; \tau^{(t-1)}\right)$ is the Onsager correction term. The Onsager correction term is derived by applying the theory of belief propagation. It substantially improves the sparsity under-sampling trade-off and has a significant influence on the performance of the algorithm [81, 82].

In order to understand how one can design new AMP-based algorithms regarding to new sophisticated priors on the signal, some of the main steps of deriving AMP algorithm are briefly explained. The derivation of AMP is divided into three steps: first derive the exact update rules and then, take the large system and β limits, and finally, reduce the number of messages.

Step 1: Derive update rules using the sum-product method

In the AMP derivation, the prior distribution $p(x_i)$ over each component of $\mathbf{x}$ is assumed to be a Laplace distribution with a common hyper-parameter β.

$$p(x_i) = \frac{\beta\lambda}{2} \exp\left(-\beta\lambda|x_i|\right). \tag{2.19}$$

Since the noise is independent and identically distributed (i.i.d.) Gaussian distributed, the likelihood function is:

$$p(\mathbf{y}|\mathbf{x}) = \mathcal{N}\left(\mathbf{y}|\mathbf{A}\mathbf{x}, \beta^{-1}\mathbf{I}_M\right), \tag{2.20}$$

where β is the noise precision and $\mathbf{I}_M$ is the identity matrix. With respect to these two equations, the joint distribution is written as follows:

$$p(\mathbf{x}, \mathbf{y}) = p(\mathbf{x})p(\mathbf{y}|\mathbf{x}) = \prod_{i=1}^{N} p(x_i) \prod_{a=1}^{M} p(y_a|\mathbf{x}). \tag{2.21}$$

This is equivalent to the factor graph[1] [83, 84] where $\mathbf{x}$ denotes the concealed variables and $\mathbf{y}$ denotes observed variables. The resulting factor graph for the joint distribution given in (2.21) is shown in Figure 2.3. The factor graph consists of multiple loops and it needs to deal

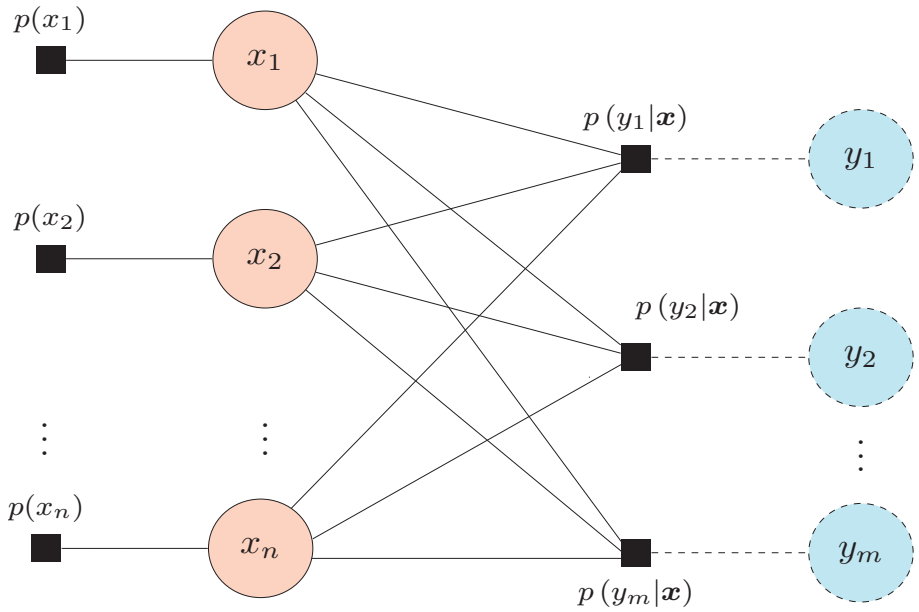


Figure 2.3. Factor graph for (2.21).

with loopy message passing by deriving the loopy sum product messages for the posterior

[1] A factor graph is a bipartite graph consisting of a set of edges and two disjoint sets of nodes: variable nodes and factor nodes. Each variable node is represented by a circle and corresponds to a unique variable in the global function. Each factor node is represented by a filled black square and corresponds to a unique factor function in the decomposition of the global function.

of x_i, which is the product of the incoming messages at variable node x_i.

$$p(x_i|\mathbf{y}) = \mu_{p(x_i)\to x_i}(x_i) \prod_a \mu_{p(y_a|\mathbf{x})\to x_i}(x_i). \tag{2.22}$$

Here, the variables $i, j \in [N]$ are the variable nodes indices, $a, b \in [M]$ are the factor nodes indices, $\mu_{p(x_i)\to x_i}(x_i)$ is the message from $p(x_i)$ to x_i which correspond the prior density (Laplace distribution), and $\mu_{p(y_a|\mathbf{x})\to x_i}(x_i)$ is the message from $p(y_a|\mathbf{x})$ to variable node x_i. To simplify the notations consider $\mu_{p(y_a|\mathbf{x})\to x_i}(x_i) = \mu_{a\to i}(x_i)$ and $\mu_{x_i\to p(y_a|\mathbf{x})}(x_i) = \mu_{i\to a}(x_i)$.

Step 2: Taking the large system and β limit

In this step the goal is to approximate both the $\mu_{a\to i}(x_i)$ and $\mu_{i\to a}(x_i)$ messages by simple parametric densities when $M, N \to \infty$. In particular, using a variant of the Berry-Esseen central limit theorem (CLT), when $N \to \infty$, the messages from factor nodes to variable nodes, $\mu_{a\to i}(x_i)$, can be approximated by a Gaussian distribution. The messages from variable nodes to factor nodes, $\mu_{i\to a}(x_i)$, can be approximated by the product of a Laplace and a Gaussian distribution. To reach this goal a family of densities $f_\beta(\mathbf{x}, s, b)$ are defined as follows:

$$f_\beta(\mathbf{x}, s, b) \equiv \frac{1}{Z_\beta} \exp\left[-\beta|\mathbf{x}| - \frac{\beta}{2b}(s - \mathbf{x})^2\right]. \tag{2.23}$$

This family of densities is a product of a Laplace distribution and a Gaussian distribution with varying parameters s, β, and b. In the second part of this step, it can be shown that in limit $\beta \to \infty$, the mean and variance of f_β distribution are soft thresholding and its derivative, respectively. By substituting these concepts of mean and variance into the update equations, the message passing schemes can be significantly simplified.

Step 3: Reduce the number of messages

As we mentioned before, AMP is an iterative algorithm. By some mathematical manipulations [71], messages can be updated according to the following rules

$$\hat{x}_{i\to a}^{(t+1)} = \eta_t\left(\sum_{b\in[m]\setminus a} A_{bi} z_{b\to i}^{(t)}; \tau^{(t)}\right), \tag{2.24}$$

$$z_{a\to i}^{(t)} = y_a - \sum_{j\in[n]\setminus i} A_{ja}\hat{x}_{j\to a}^{(t)}, \tag{2.25}$$

where in each iteration $2MN$ messages are propagated. In order to reduce the number of messages Donoho et al.[70] introduced another approximation. The idea is to approximate

$\hat{x}_{i\to a}^{(t)}$ and $z_{a\to i}^{(t)}$ as follows

$$\hat{x}_{i\to a}^{(t)} = \hat{x}_i^{(t)} + \epsilon.\hat{x}_{i\to a}^{(t)} + O\left(\frac{1}{M}\right), \tag{2.26}$$

$$z_{a\to i}^{(t)} = z_a^{(t)} + \epsilon.z_{a\to i}^{(t)} + O\left(\frac{1}{M}\right), \tag{2.27}$$

the reason to do that is in (2.24) it is seen that $\hat{x}_{i\to a}^{(t)}$ only depends weakly on index a, since the right hand side only depends on index a, through the missing term in the sum. Analogously, the same is true in (2.25) for $z_{a\to i}^{(t)}$ and index i. By substituting these two expressions in (2.24) and (2.25) and using first order approximation along with the law of large numbers, the update equations forms as follows

$$\hat{\mathbf{x}}^{(t+1)} = \eta\left(\mathbf{A}^T\mathbf{z}^{(t)} + \hat{\mathbf{x}}^{(t)}; \tau^{(t)}\right), \tag{2.28}$$

$$\mathbf{z}^{(t)} = \mathbf{y} - \mathbf{A}\hat{\mathbf{x}}^{(t)} + \frac{1}{\delta}\mathbf{z}^{(t-1)}\left\langle \eta'\left(\mathbf{A}^T\mathbf{z}^{(t-1)} + \hat{\mathbf{x}}^{(t-1)}; \tau^{(t-1)}\right)\right\rangle. \tag{2.29}$$

This algorithm only requires propagating $M + N$ messages in each iteration. Thus, the dominating operations in each iteration is the two matrix multiplications, $\mathbf{A}\hat{\mathbf{x}}^{(t)}$ and $\mathbf{A}^T\mathbf{z}^{(t)}$, which both scale as $O(MN)$. Therefore, each iteration has complexity $O(MN)$.

2.3.2 Denoising-based approximate message passing (DAMP)

A new CS recovery framework named DAMP has been designed in [11]. DAMP uses a denoising algorithm to recover signals from compressive measurements. Many applications like images do not have a sparse representation in a well-known basis such as wavelet and DCT. For this reason the authors designed an improved and more general CS recovery framework based on AMP algorithm.

DAMP has special advantages and can be applied to many different signal models such as sparse, smooth, or low-rank signals. This framework is capable of high performance reconstruction and overcomes the drawbacks of AMP and other AMP-based algorithms, e.g., DAMP offers state-of-the-art CS recovery performance for natural images, for an appropriate choice of denoiser. Depending on the signal model, the main goal of the DAMP framework is to use appropriate denoiser which leads to near optimum recovery. Depend on the specific signal model, family of denoisers (e.g non-local means (NLM) [85], block matching 3D (BM3D) [86]) are used to achieve a good estimate of the true signal $\mathbf{x} \in C$, where C is class of signals from (2.10).

To avoid the non-Gaussianity of the estimated noise, a proper Onsager correction term is employed in the DAMP iterations. The Onsager correction term forces the distribution of the

signal error at each iteration to be very close to white Gaussian noise which denoisers are typically designed to remove. We will discuss DAMP in more detail in the chapter 3.

2.3.3 Generalized approximate message passing (GAMP)

The GAMP algorithm is introduced in [73]. Rangan generalized the AMP algorithm to deal with arbitrary noise and prior distributions. Hence, GAMP is a generalization of AMP, where AMP is applicable only for Gaussian channels. The GAMP analysis follows the AMP analysis of [87] and it is also very related to [88, 89], but states a precise analysis with a dense matrix $\mathbf{A}$, along with a simpler implementation.

GAMP's flexibility permits to do efficient inference via sparsity supporting prior distributions like the spike and slab prior [90]. Besides, GAMP could be used for classification when the noise follows the binomial distribution [91]. Although GAMP's flexibility is significantly increased, the computational complexity remains the same as AMP, i.e., $O(NM)$.

It should be noted that the GAMP derivation is mainly based on the Taylor series and an application of CLT, which is more straightforward than the derivation of AMP. The GAMP framework is configured not only to perform maximum a-posteriori probability (MAP) estimation using max-sum message passing but also to perform minimum mean squared error (MMSE) estimation using sum-product message passing. The authors provide two versions of GAMP for both MMSE and MAP estimations. The simplified version of the GAMP framework corresponds to the AMP algorithm, if the prior and noise distribution are chosen to be Laplace and Gaussian, respectively.

2.3.4 GAMP for cosparse analysis (GrAMPA)

The GrAMPA framework [75] is based on a Bayesian approach to cosparse analysis CS which leverages GAMP algorithm. In cosparse analysis CS, we aim to estimate a non-sparse signal from a noisy under-determined system where the signal is cosparse for a certain linear transform. The GrAMPA has very good universality to signal models and noise channels. It provides a novel approach to cosparse analysis CS.

This Bayesian approach works with a generic analysis operator and vast range of signal priors. It encapsulates both MAP and MMSE estimations of true signal $\mathbf{x}$ efficiently to decrease the computational complexity. Furthermore, in this work, a sparse non-informative parameter estimator (SNIPE) denoiser algorithm is represented, which supports Bernoulli-Uniform and Bernoulli-Gaussian priors. Numerical results show improved recovery performance and an excellent runtime of GrAMPA-SNIPE approach compared with other recent analysis-CS algorithms.

2.3.5 Efficient fused lasso algorithm (EFLA)

The authors in [76] focused on optimizing a class of problems with the fused lasso penalty. The problem in (2.16) is challenging to solve as it leads to a class of non-smooth and non-separable optimization problems. The fused lasso penalty imposes sparsity in the coefficients and their successive differences, which is desirable for applications with features ordered in some meaningful way. Most of the presented techniques have high computational complexity and do not scale well to large-size problems.

For optimizing this class of problems, the objective function is considered the fused function as a mixture of with the smooth part and the other non-smooth part. The authors proposed to apply the Nesterov's method [92] to develop EFLA. Nesterov's method is a first order black-box method for smooth convex optimization. The EFLA method leverages the special structure of defined problem (2.16) to achieve the optimal first-order black-box methods convergence rate of $O\left(\frac{1}{k^2}\right)$ for k iterations. Applying the first-order black-box method to solve the non-smooth problem (2.16) leads to the convergence rate of $O\left(\frac{1}{\sqrt{k}}\right)$ which is much slower than $O\left(\frac{1}{k^2}\right)$. Using empirical evaluations, the authors have shown that EFLA significantly outperform existing solvers, and claimed that it is applicable for large-scale problems.

2.3.6 Final remarks

Developing new recovery algorithms for different signal models specially for sparse recovery is a very active field of research and new algorithms are proposed monthly. In addition to the algorithms that we discussed above there are still many high qualified research works in this field. Here, we focused on the state-of-the-art algorithms which we want to exploit in our work.

Chapter 3

Graph signal recovery using approximate message passing

3.1 Introduction

As mentioned briefly in the previous chapters, in this dissertation we mainly focus on the problem of recovering a graph signal from noisy and incomplete information. Particularly in this chapter, we propose an AMP flavored method for graph signal recovery. The recovery of the graph signal is based on noisy signal values at a small number of randomly selected nodes. We exploit the smoothness of typical graph signals occurring in many applications, such as wireless sensor networks or social network analysis. The graph signals are smooth in the sense that neighbouring nodes have similar signal values.

Based on our work in [12], the contribution of the chapter is summarized as follows. We design an AMP-based graph signal recovery method which is able to cope with incomplete and noisy measurements. Methodologically, this algorithm can be regarded as an instance of the denoising framework in [11] for the special case of graph signals having small total variation, i.e., which consist of few clusters within which the signal values do not vary significantly. The signal recovery is based on a small number of noisy samples of the smooth graph signal. We conduct illustrative numerical experiments which validate the performance of the proposed recovery method and reveal superiority of our approach against existing methods in some relevant scenarios.

The rest of this chapter is organized as follows. First, in Section 3.2, we formalize the problem of graph signal recovery from incomplete and noisy measurements. The novel DAMP-based recovery method is presented in Section 3.3. Finally, some numerical results are presented and discussed in Section 3.4.

3.2 Problem setup

In the Section 3.2.1 we focus on the GSP concept in more detail and afterwards in Section 3.2.2, we will describe the recovery problem which we are interested in.

3.2.1 Elements of graph signal processing

The emerging field of GSP [1, 19] aims at dealing efficiently with decentralized, graph-structured data as encountered in modern information networks. GSP is a generalization of discrete time signal processing. Specifically, discrete time signals may be interpreted as graph signals defined over a chain graph whose nodes represent the discrete time instants. A general graph signal is obtained by allowing for general graph structures (see Figure 3.1).

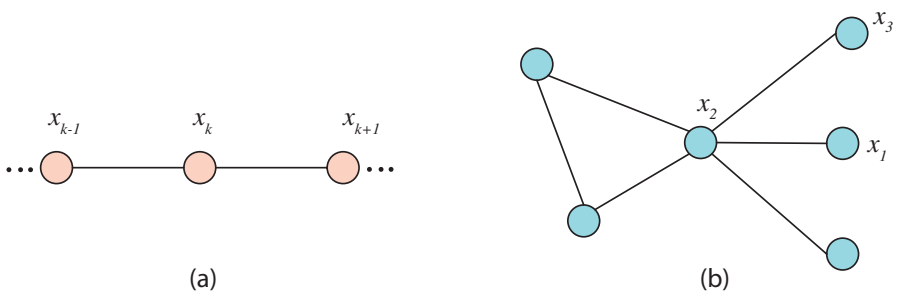


Figure 3.1. Chain graph underlying discrete time signal processing (a) and generic graph signal (b).

More formally, we consider the undirected weighted graphs $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{W})$ with node set $\mathcal{V} = \{1, \ldots, N\}$ and the edge set $\mathcal{E}$ consisting of unordered node pairs (r, s) for which $W_{rs} \neq 0$. For a given graph $\mathcal{G}$, a graph signal $\mathbf{x}$ is a mapping from the node set into the reals. We can represent a graph signal conveniently as a vector $\mathbf{x} \in \mathbb{R}^N$ by defining x_r to be the value of the graph signal at node $r \in \mathcal{V}$.

For the undirected weighted graphs like the graph in Figure 3.2, $\mathbf{W} \in \mathbb{R}^{N \times N}$ is symmetric, i.e., $\mathbf{W} = \mathbf{W}^T$. A non-zero entry of the weight matrix W_{rs} represents the strength of the correlation between signal values x_r and x_s. In a wireless sensor network application, the entry W_{rs} could reflect the distance between sensor nodes r and s. It is reasonable to assume that the sensor values x_r and x_s of nearby sensor nodes to be strongly correlated. Another important matrix associated with a graph is the graph Laplacian matrix $\mathbf{L}$, defined as

$$\mathbf{L} = \mathbf{D} - \mathbf{W}. \tag{3.1}$$

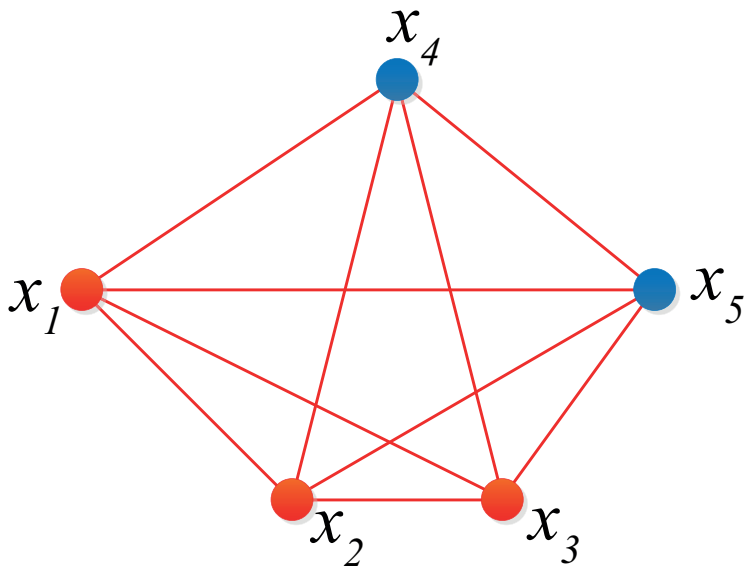


Figure 3.2. The sample undirected weighted graph, and its weighted adjacency matrix, where the signal values are 1 (red) and -1 (blue).

Here, $\mathbf{D}$ denotes the diagonal matrix with the r^{th} diagonal element given by

$$D_{rr} = \sum_{r' \in \mathcal{V}} D_{rr'}, \tag{3.2}$$

i.e., the sum of the weights of all the edges connected to node r. Many methods of discrete time signal processing (e.g., denoising), rely on smoothness of the signal with respect to the underlying graph. In order to make the notion of graph signal smoothness precise, we follow [19] and introduce the graph gradient

$$\|\nabla_r \mathbf{x}\|_2 := \left[\sum_{r' \in \mathcal{N}(r)} W_{rr'} (x_{r'} - x_r)^2 \right]^{1/2}. \tag{3.3}$$

The norm $\|\nabla_r \mathbf{x}\|$ of the local gradient is termed local variation and measures the variability of the graph signal at a given node r. Here, $\mathcal{N}(r) := \{r' : W_{rr'} \neq 0\}$ is the neighbourhood of node $r \in \mathcal{V}$. A global measure of the graph signal smoothness is then obtained by

$$S_p(\mathbf{x}) = (1/p) \sum_{r \in \mathcal{V}} \|\nabla_r \mathbf{x}\|_p^p, \tag{3.4}$$

for some $p \in [1, \infty)$. In order to compare the smoothness of various graph signals with each other, we define a new smoothness factor $S'_p(\mathbf{x})$ which can be obtained by normalizing the value of $S_p(\mathbf{x})$:

$$S'_p(\mathbf{x}) = \frac{S_p(\mathbf{x})}{\|\mathbf{x}\|_p^p}. \tag{3.5}$$

In what follows we will consider only two specific choices for p, i.e., $p = 1$ and $p = 2$. For $p = 1$ the measure $S_p(\mathbf{x})$ is termed the *graph total variation* [93] and when $p = 2$ the measure $S_p(\mathbf{x})$ reduces to the *graph Laplacian form* [19]

$$S_2(\mathbf{x}) = (1/2) \sum_{r \in \mathcal{V}} \sum_{r' \in \mathcal{N}(r)} W_{rr'}(x_{r'} - x_r)^2 = \mathbf{x}^T \mathbf{L} \mathbf{x}. \tag{3.6}$$

We have now the tools at hand to formalize the graph signal recovery problem that is considered in this chapter.

3.2.2 The recovery problem

Our approach is based on the hypothesis that the true graph signal $\mathbf{x}$ is smooth, i.e., the measure $S_p(\mathbf{x})$ ($p \in \{1, 2\}$) is small. We have access to the graph signal $\mathbf{x}$ only via its values at a randomly selected small subset $\mathcal{S} = \{i_1, \ldots, i_M\} \subseteq \mathcal{V}$ of graph nodes. Moreover, the observed signal values are corrupted by measurement noise. Thus, the observation is given by

$$\mathbf{y} = \mathbf{x}\big|_{\mathcal{S}} + \sigma \mathbf{n}, \tag{3.7}$$

where the restriction $\mathbf{x}\big|_{\mathcal{S}}$ is obtained from $\mathbf{x}$ by selecting the entries of $\mathbf{x}$ with indices in $\mathcal{S}$. The noise vector $\mathbf{n}$ is assumed to be white Gaussian noise with zero-mean and unit variance, i.e., $\mathbf{n} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. Alternatively, we can represent the vector $\mathbf{y}$ as

$$\mathbf{y} = \mathbf{A}\mathbf{x} + \sigma \mathbf{n}. \tag{3.8}$$

The measurement matrix $\mathbf{A} \in \{0, 1\}^{M \times N}$ models the selection of the subset $\mathcal{S}$: It contains exactly one non-zero element in each row. Since the subset $\mathcal{S}$ is chosen randomly, the matrix $\mathbf{A}$ is random as well.

While the recovery of graph signals from incomplete noisy measurements has been considered already in [35], the application of AMP to the recovery of smooth graph signals recovery based on constraining the graph total variation seems to be new. The subsampling exploits the intrinsic low-dimensional structure inherent to graph signals which are smooth,

i.e., when the quantity $S_p(\mathbf{x})$ ($p \in \{1, 2\}$) is small. Since our recovery problem can be interpreted as a structured signal recovery problem using incomplete information, we will now propose a recovery method based on the DAMP framework which is well suited for such recovery problems to deal with subsampled structured signals.

3.3 Graph signal denoising via DAMP

In this section, first we recall the DAMP method then we state a detailed description of our proposed algorithm named graph signal DAMP (GSDAMP).

3.3.1 Review of DAMP

Consider a signal $\mathbf{x} \in \mathbb{R}^N$ which is known to belong to some signal class C, e.g., graph signals with small total variation. For many important signal classes C, one can find efficient denoising functions $\mathbf{D}_\sigma(\cdot)$ which operate on the noisy signal

$$\mathbf{y} = \mathbf{x} + \sigma\mathbf{n}, \tag{3.9}$$

where $\mathbf{n}$ is modeled as zero-mean white Gaussian noise with unit variance, i.e., $\mathbf{n} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. The denoising mapping $\mathbf{D}_\sigma(\cdot)$ typically depends on the variance σ^2 of the additive noise in (3.9). However, the notation $\mathbf{D}_\sigma(\cdot)$ does not make explicit that the denoiser also depends on the signal model C. The output $\mathbf{D}_\sigma(\mathbf{y})$ of the denoiser, when applied to the noisy signal $\mathbf{y}$, is an estimate for the true signal $\mathbf{x}$. For C being the set of sparse signals, i.e., $C = \mathcal{X}_s := \{\mathbf{s} \in \mathbb{R}^N : \|\mathbf{s}\|_0 \leq s\}$ it is known that an efficient denoising mapping is obtained by retaining the s largest (magnitude) entries of $\mathbf{y}$ and zeroing the rest.

However, as opposed to the signal in noise model (3.9), recently much interest has been devoted to the problem of recovering a structured signal $\mathbf{x}$ from incomplete information given by noisy low-dimensional random projections (3.8), where $\mathbf{A} \in \mathbb{R}^{M \times N}$, with $M \ll N$, denotes a random projection matrix. A widely used choice for $\mathbf{A}$ is the Gaussian ensemble which is a matrix consisting of i.i.d. zero-mean Gaussian variables with variance $1/M$ (which ensures column normalization). By leveraging the principles behind AMP, which considers the special case of sparse signals $C = \mathcal{X}_s$, the authors of [11] propose an efficient iterative method, termed DAMP, for recovering a structured signal $\mathbf{x} \in C$ from the measurements $\mathbf{y}$ in (3.8) for a general signal class C with associated denoiser $\mathbf{D}_\sigma(\cdot)$.

In particular, DAMP constructs a sequence $\hat{\mathbf{x}}^{(t)}$, $t = 1, 2, \ldots$, of signal estimates by iterating the following steps [11, Eq. (4)]

$$\hat{\mathbf{x}}^{(t+1)} = \mathbf{D}_{\hat{\sigma}^{(t)}}\left(\mathbf{A}^T\mathbf{z}^{(t)} + \hat{\mathbf{x}}^{(t)}\right), \tag{3.10}$$

$$\mathbf{z}^{(t)} = \mathbf{y} - \mathbf{A}\hat{\mathbf{x}}^{(t)} + \frac{1}{M}\mathbf{z}^{(t-1)}\nabla\cdot\mathbf{D}_{\hat{\sigma}^{(t-1)}}\left(\mathbf{A}^T\mathbf{z}^{(t-1)} + \hat{\mathbf{x}}^{(t-1)}\right), \tag{3.11}$$

$$\hat{\sigma}^{(t)} = \sqrt{(1/M)\|\mathbf{z}^{(t)}\|_2^2}, \tag{3.12}$$

the initial choice for the estimate $\hat{\mathbf{x}}$ and residual $\mathbf{z}$ is $\hat{\mathbf{x}}^{(0)} = \mathbf{0}$ and $\mathbf{z}^{(0)} = \mathbf{y}$, respectively.

The DAMP iterations (3.10)–(3.12) are able to accurately recover the structured signal $\mathbf{x} \in C$ from the noisy measurements $\mathbf{y}$ in (3.8). The correction term in (3.11), i.e., $\frac{1}{M}\mathbf{z}^{(t-1)}\nabla\cdot\mathbf{D}_{\hat{\sigma}^{(t-1)}}(\mathbf{A}^T\mathbf{z}^{(t-1)} + \hat{\mathbf{x}}^{(t-1)})$ is crucial for the success of the DAMP algorithm, where $\nabla\cdot\mathbf{D}_{\hat{\sigma}^{(t-1)}}(\cdot)$ is the derivative of the denoiser $\mathbf{D}_{\hat{\sigma}^{(t-1)}}(\cdot)$. The effect of including this term in (3.11) is that the equivalent estimation noise $\mathbf{n}^{(t)} := \mathbf{A}^T\mathbf{z}^{(t)} + \hat{\mathbf{x}}^{(t)} - \mathbf{x}$ behaves nearly like a multivariate normal random vector [11]. The (approximate) Gaussianity of the effective noise vector $\mathbf{n}^{(t)}$ is clearly desirable since the denoiser $\mathbf{D}_{\hat{\sigma}^{(t)}}(\cdot)$ is typically trimmed to remove additive Gaussian noise and the first step (3.10) of DAMP just amounts to the denoising operation

$$\hat{\mathbf{x}}^{(t+1)} = \mathbf{D}_{\hat{\sigma}^{(t)}}\left(\mathbf{x} + \mathbf{n}^{(t)}\right). \tag{3.13}$$

3.3.2 Graph signal DAMP (GSDAMP)

Let us now specialize the generic DAMP algorithm, given by the iterations (3.10)–(3.12), to the problem of graph signal denoising. We assume that the true signal belongs to the class C of smooth graph signals, given explicitly by $C = \{\mathbf{x} : S_p(\mathbf{x}) \leq \tau\}$. A natural choice for the corresponding denoiser, which is to be applied to a noisy graph signal $\mathbf{s} = \mathbf{x} + \sigma\mathbf{n}$, would be given by the minimizer of the following problem:

$$\min_{\mathbf{x}\in\mathbb{R}^N} \|\mathbf{s} - \mathbf{x}\|_2^2 \quad \text{s.t.} \quad S_p(\mathbf{x}) \leq \tau. \tag{3.14}$$

However, we will find it more convenient to use the "penalized version" of (3.14), i.e.,

$$\mathbf{D}_{\lambda,W}(\mathbf{s}) := \arg\min_{\mathbf{x}\in\mathbb{R}^N} \|\mathbf{s} - \mathbf{x}\|_2^2 + \lambda S_p(\mathbf{x}). \tag{3.15}$$

For convex $S_p(\mathbf{x})$, which is the case for $p \geq 1$, the two problems (3.14) and (3.15) are equivalent by Lagrangian duality [94]. In fact, for each choice of τ there exists a choice for λ such that a minimizer (3.14) is simultaneously a solution to (3.15) and vice versa.

The objective function in (3.15) is a weighted sum of two convex terms: the regularization term and the data fidelity term. The regularization parameter λ plays a significant role in the accurate recovery of graph signals. Choosing a proper value for λ, leads to attain a compromise to suppress the noise and preserve the original graph signal. If we use a very small value for λ, then the solution will be under-smooth, it will fit the given data properly but it will preserve the noise in homogeneous areas. However, if we use a very large value for λ, then the solution will not fit the measured noisy graph signal values accurately while it will be over smooth and remove not only the noise but also the important details of the data [95]. Obviously, choosing a proper value for λ is really tricky and most importantly depends on the signal structure. To have a best recovery solution we should be able to set a value in a way that leads to a good compromise of the above mentioned outcomes.

In order to deploy DAMP for graph signal denoising, we require an efficient implementation of the denoiser mapping $\mathbf{D}_{\lambda,W}(\cdot)$ and its divergence $\nabla\cdot\mathbf{D}_{\lambda,W}(\mathbf{x}) := \sum_{k=1}^{N} \frac{\partial}{\partial x_k}\mathbf{D}_{\lambda,W_k}(\mathbf{x})$, where x_k is the k^{th} element of signal $\mathbf{x}$ and W_k is the k^{th} row of the symmetric weight matrix $\mathbf{W}$. Note that the denoiser amounts to solving a convex optimization problem allowing for efficient numerical implementations. In particular, we rely on the freely available software package GSPBox [96]. In order to evaluate the divergence $\nabla\cdot\mathbf{D}_{\lambda,W}(\mathbf{x})$, we follow [11]: An approximation of the divergence can be obtained by [97]

$$\nabla\cdot\mathbf{D}_{\lambda,W}(\mathbf{x}) \approx \mathbb{E}_{\mathbf{b}}\left\{(1/\varepsilon)\mathbf{b}^T\left(\mathbf{D}_{\lambda,W}(\mathbf{x}+\varepsilon\mathbf{b}) - \mathbf{D}_{\lambda,W}(\mathbf{x})\right)\right\}, \tag{3.16}$$

for some small $\varepsilon > 0$. However, in the numerical implementation we will make a further approximation by replacing the expectation in (3.16) with a sample mean, i..e, we use

$$\tilde{d}(\mathbf{x}) := \frac{1}{L}\sum_{l=1}^{L}(1/\varepsilon)\mathbf{b}_l^T\left(\mathbf{D}_{\lambda,W}(\mathbf{x}+\varepsilon\mathbf{b}_l)-\mathbf{D}_{\lambda,W}(\mathbf{x})\right), \tag{3.17}$$

where $\mathbf{b}_1,\ldots,\mathbf{b}_L$ are i.i.d. realizations of the random vector $\mathbf{b}\sim\mathcal{N}(\mathbf{0},\mathbf{I})$ and L is the number of the realizations. One can choose a very large value for L, e.g., $L = N$, where N is number of the graph signal components.

The summary of our method, which we term graph signal denoising using AMP (GSDAMP), is given in Algorithm 1.

There are various possibilities for the stopping criterion in Line 10 of Algorithm 1, e.g. a maximum number of iterations. For the numerical experiments discussed in Section 3.4, we used as convergence criteria a maximum number of iterations, along with the relative progress $\frac{\|\hat{\mathbf{x}}^{(t)}-\hat{\mathbf{x}}^{(t-1)}\|_2}{\|\hat{\mathbf{x}}^{(t)}\|_2}$ and Algorithm 1 stopped if it was below a given threshold ϵ.

Algorithm 1: Graph signal denoising AMP (GSDAMP)

1 **Input:** given the noisy graph signal samples $\mathbf{y}$ (cf. (3.8)), sampling pattern $\mathcal{S}$, and denoising parameter p (cf. (3.15)) perform the following:
2 **Initialization:** set $t = 0$, $\hat{\mathbf{x}}_0 = \mathbf{0}$, $\hat{\mathbf{z}}_0 = \mathbf{y}$,
3 **Implement:** a DAMP iteration via
4 $\tilde{\mathbf{x}}^{(t)} = \mathbf{A}^T\mathbf{z}^{(t)} + \hat{\mathbf{x}}^{(t)}$
5 $\hat{\sigma}^{(t)} = \sqrt{(1/M)\|\mathbf{z}^{(t)}\|_2^2}$ (compute $\hat{\sigma}^{(t)}$ by (3.12))
6 $\hat{\lambda}^{(t)} = 1 - \exp\left(-\hat{\sigma}^{(t)}\right)$
7 $\hat{\mathbf{x}}^{(t+1)} = \mathbf{D}_{\hat{\lambda}^{(t)},W}(\tilde{\mathbf{x}}^{(t)})$ (using the denoiser (3.15))
8 $\mathbf{z}^{(t)} = \mathbf{y} - \mathbf{A}\hat{\mathbf{x}}^{(t)} + \frac{1}{M}\mathbf{z}^{(t-1)}\tilde{d}(\tilde{\mathbf{x}}^{(t-1)})$ (using approximation (3.17))
9 $t := t + 1$
10 **Output:** final estimate $\hat{\mathbf{x}}^{(t)}$ if stopping criterion satisfied, otherwise go back to Line 3.

3.4 Numerical results

In this section, we present the results of the numerical experiments validating the performance of the proposed method using various undirected graphs such as the undirected random irregular (RIR) graph [96], the undirected Bunny graph [96], the Minnesota road graph (network) [98], as well as a graph of real-world data-set, i.e., the Amazon product rating graph [99]. In the first part of the performance validation, we analyze normalized mean squared error (NMSE) for varying sampling rate M/N, and noise standard deviation σ. In the next part, we focus on analyzing the label recovery ratio (LRR) with varying sampling rate M/N, and noise standard deviation σ.

Here, we use Algorithm 1 with total variation ($p = 1$ in (3.4)) [93] and Tikhonov ($p = 2$ in (3.4)) [100] denoisers. We refer to these two instances of Algorithm 1 as total variation GSDAMP (TV-GSDAMP) and Tikhonov GSDAMP (Tik-GSDAMP) respectively, and compare their efficiency with EFLA [76] and GrAMPA [75] the state-of-the-art graph signal recovery algorithms. A rigorous description regarding to the EFLA and GrAMPA algorithms is provided in Chapter 2. In order to implement EFLA and GrAMPA, we use the online available codes in [101] and [102] respectively.

3.4.1 Undirected smooth graph signals

Using the GSPBox software [96], we generate three distinct undirected graphs, i.e., the RIR graph with size $N = 1000$, $\mathcal{E} = 3362$ edges and the average node degree of 6.7, the Bunny graph with size $N = 2503$, $\mathcal{E} = 13726$ edges and the average node degree of 11, as well as the

Minnesota road graph with size $N = 2642$ and $\mathcal{E} = 3304$ edges and the average node degree of 3. In addition to these three graphs, we applied our algorithm along with the other recovery algorithms to the real-world data-set of the Amazon product rating graph [99].
The Amazon product rating data-set was collected by crawling the website of the Amazon Internet-based retailer [99]. This data-set consists of rating information of four different product categories: books, music CDs, DVDs and video tapes. The products are represented by the nodes of the graph. The nodes representing two particular products are connected by an edge if they are co-purchased often. Each product is assigned ratings by the users who bought it, taking values in the set $1/2\{0, 1, \ldots, 9, 10\}$. We then obtain a graph signal $\mathbf{x}_0$ by setting its value at node i to the average of all ratings for the product i. The graph of the raw data contained isolated nodes and several small components. We select the largest connected subgraph $\mathcal{G}$ for our numerical experiments. In this graph, there are $N = 290744$ nodes, $|\mathcal{E}| = 729048$ edges and the average degree of each node is 5. In order to obtain the undirected Amazon product rating graph the edge directions were ignored.

We randomly selected M noise-contaminated signal samples x_i and this way obtained the measurement vector $\mathbf{y}$. For sufficient statistical significance of the results we ran the recovery method for 500 times and each time with different noise realizations. We set the stopping criteria in a way that the algorithm either reaches the maximum number of 50 iterations or the relative progress saturates according to $\frac{\|\hat{\mathbf{x}}^{(t)} - \hat{\mathbf{x}}^{(t-1)}\|_2}{\|\hat{\mathbf{x}}^{(t)}\|_2} \leq \epsilon$, where the threshold $\epsilon = 10^{-2}$. The Algorithm 1 stops when it meets either one of the mentioned criteria. The final result is averaged over the outcomes of the individual runs of the recovery scheme. All of the graph signals are smooth and, except for the Amazon product rating graph signal, the rest of the graph signals take values 1 and -1 coded by red and blue colours in Figures 3.3 to 3.5.

In particular, Figures 3.3 to 3.5 show the original graph signal $\mathcal{G}$ and four recovered graph signals of the mentioned recovery algorithms. As evident, the graph signals recovered by TV-GSDAMP and Tik-GSDAMP are much more similar to the original graph signals compare to the results of EFLA and GrAMPA recovery algorithms.

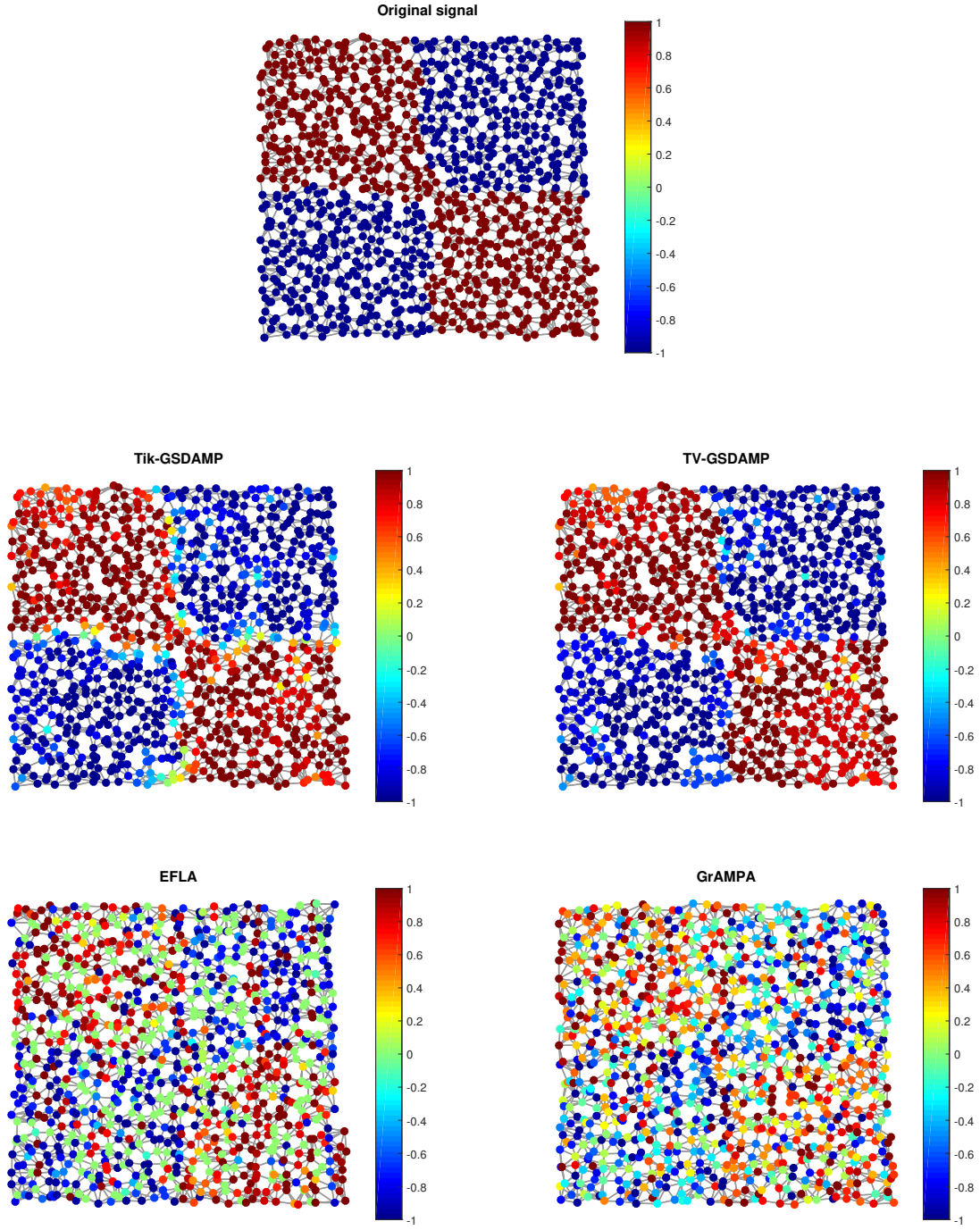


Figure 3.3. The original RIR graph $\mathcal{G}$ and its recovery with four different recovery algorithms for a sampling rate $M/N = 0.3$ and a noise standard deviation $\sigma = 0.3$.

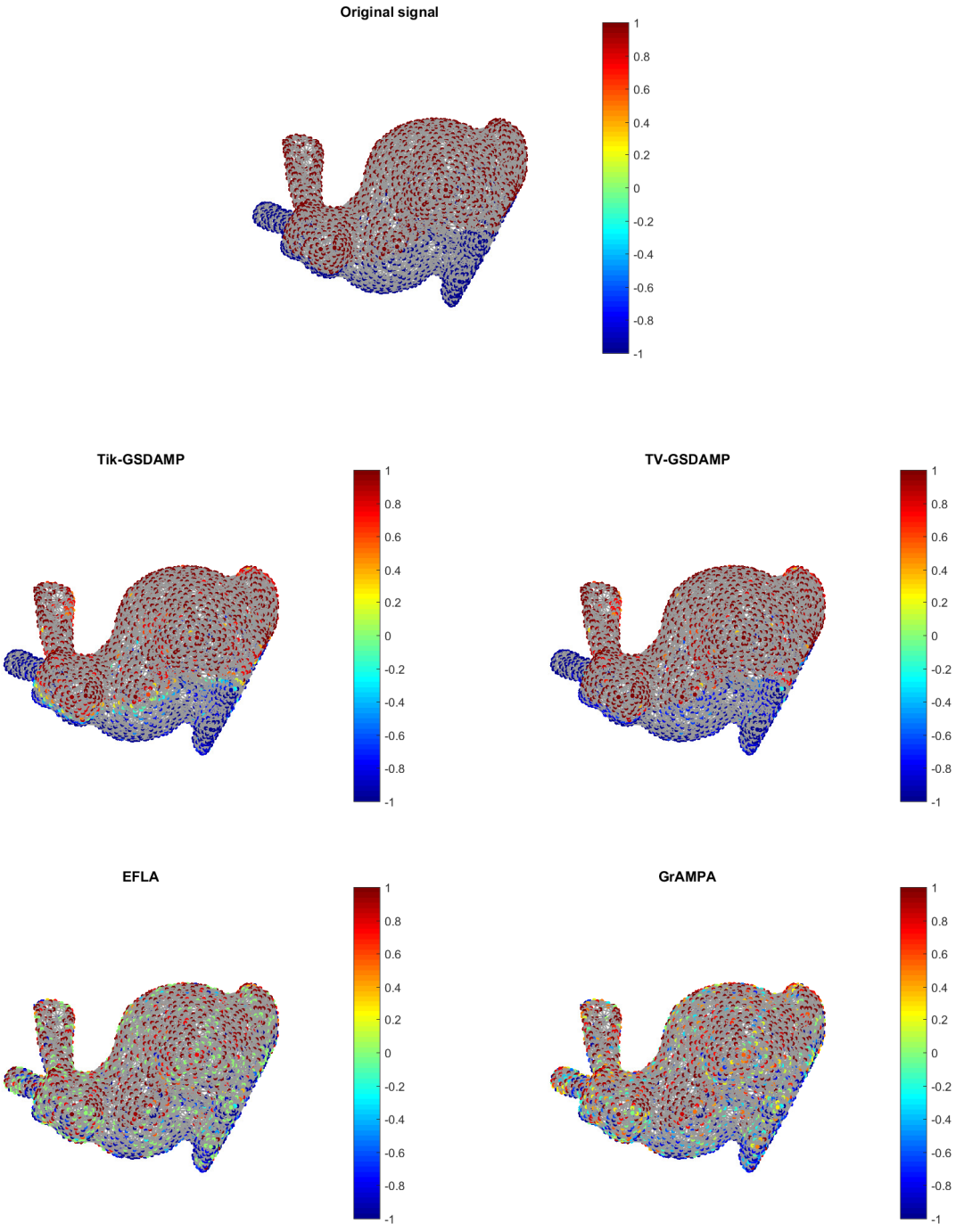


Figure 3.4. The original Bunny graph $\mathcal{G}$ and its recovery with four different recovery algorithms for a sampling rate $M/N = 0.3$ and noise standard deviation $\sigma = 0.3$.

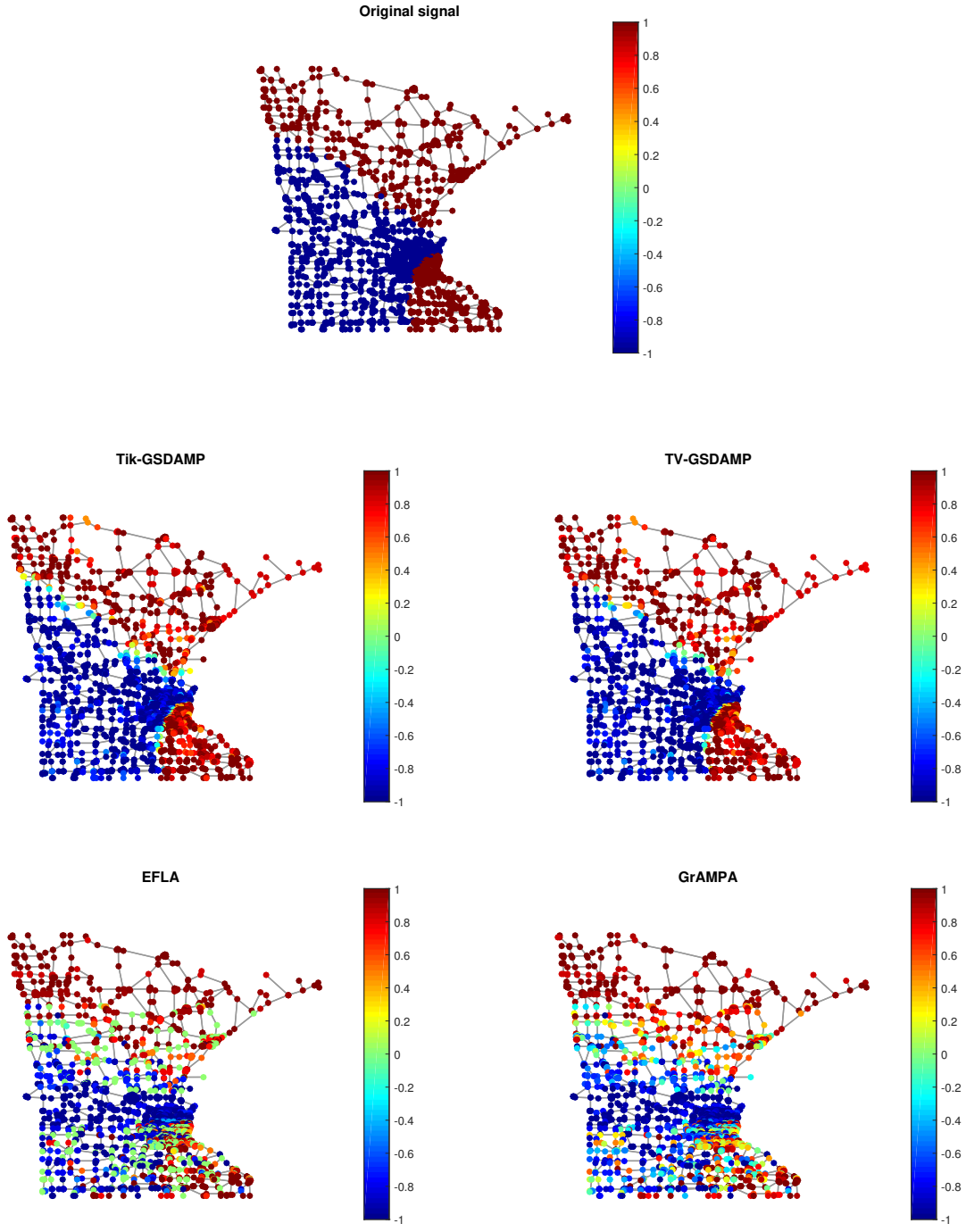


Figure 3.5. The original Minnesota road graph $\mathcal{G}$ and its recovery with four different recovery algorithms for a sampling rate $M/N = 0.3$ and a noise standard deviation $\sigma = 0.3$.

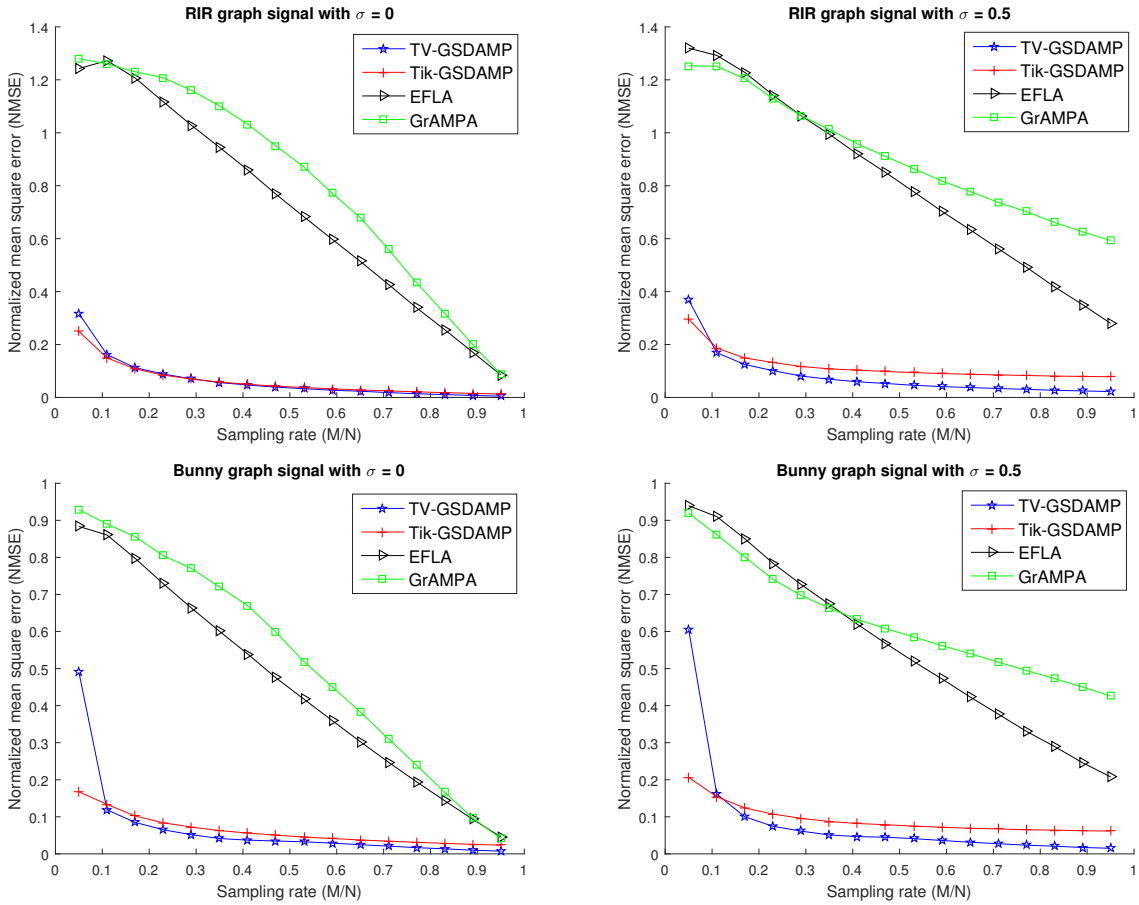


Figure 3.6. NMSE vs sampling rate M/N for RIR and Bunny graphs, where the noise standard deviation σ is set to 0 and 0.5.

The influence of sampling rate M/N and noise standard deviation σ on recovery performance of the mentioned graph signals is investigated in the Section 3.4.2 and Section 3.4.3, respectively.

3.4.2 Recovery performance regarding to NMSE

In Figures 3.6 and 3.7, we plot the NMSE over sampling rate M/N for TV-GSDAMP and Tik-GSDAMP. We show the corresponding plots for the EFLA and GrAMPA recovery methods as well. In these figures, the x-axis is sampling rate $M/N \in [0.05, 0.95]$ and the y-axis is the NMSE. As clear from Figures 3.6 and 3.7, for all four graph signals the NMSE of TV-GSDAMP and Tik-GSDAMP is always smaller compared to other solvers, for both noiseless and noisy graph signals.

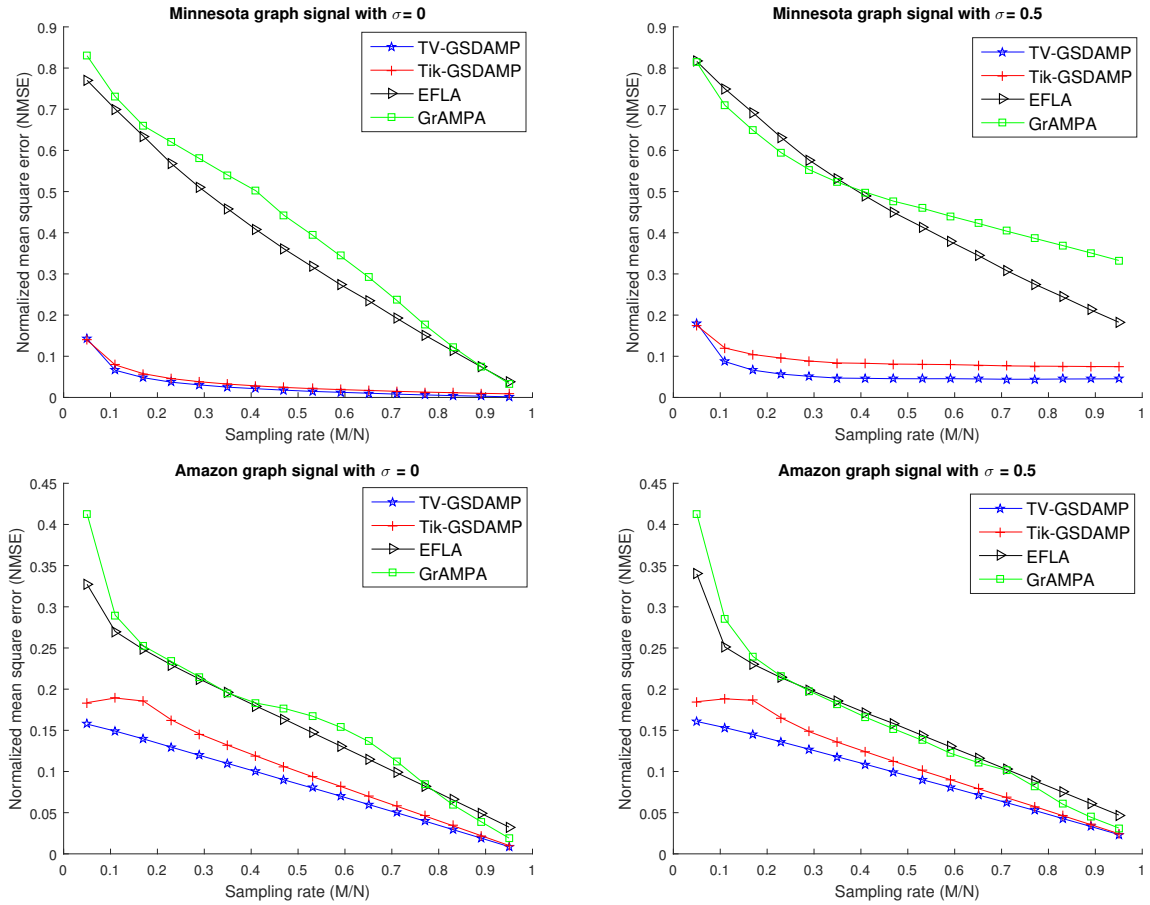


Figure 3.7. NMSE vs sampling rate M/N for Minnesota and Amazon graphs, where the noise standard deviation σ is set to 0 and 0.5.

Apart from the impact of sampling rate on NMSE, we also investigate the dependence of NMSE on the noise standard deviation σ for sampling rates of $M/N \in \{0.2, 0.4\}$. In Figures 3.8 and 3.9, the x-axis is the noise standard deviation $\sigma \in [0, 1]$ and the y-axis is the NMSE. As evident from Figures 3.8 and 3.9, for all scrutinized graph signals the proposed methods outperform the other algorithms, particularly in low sampling rates regime.

With respect to Figures 3.6 to 3.9, both Tik-GSDAMP and TV-GSDAMP show significantly better recovery performance, on average more than 2 times better, than GrAMPA and EFLA, specially for low sampling rates, i.e., when $M/N \leq 0.1$, generally fail to recover the graph signal.

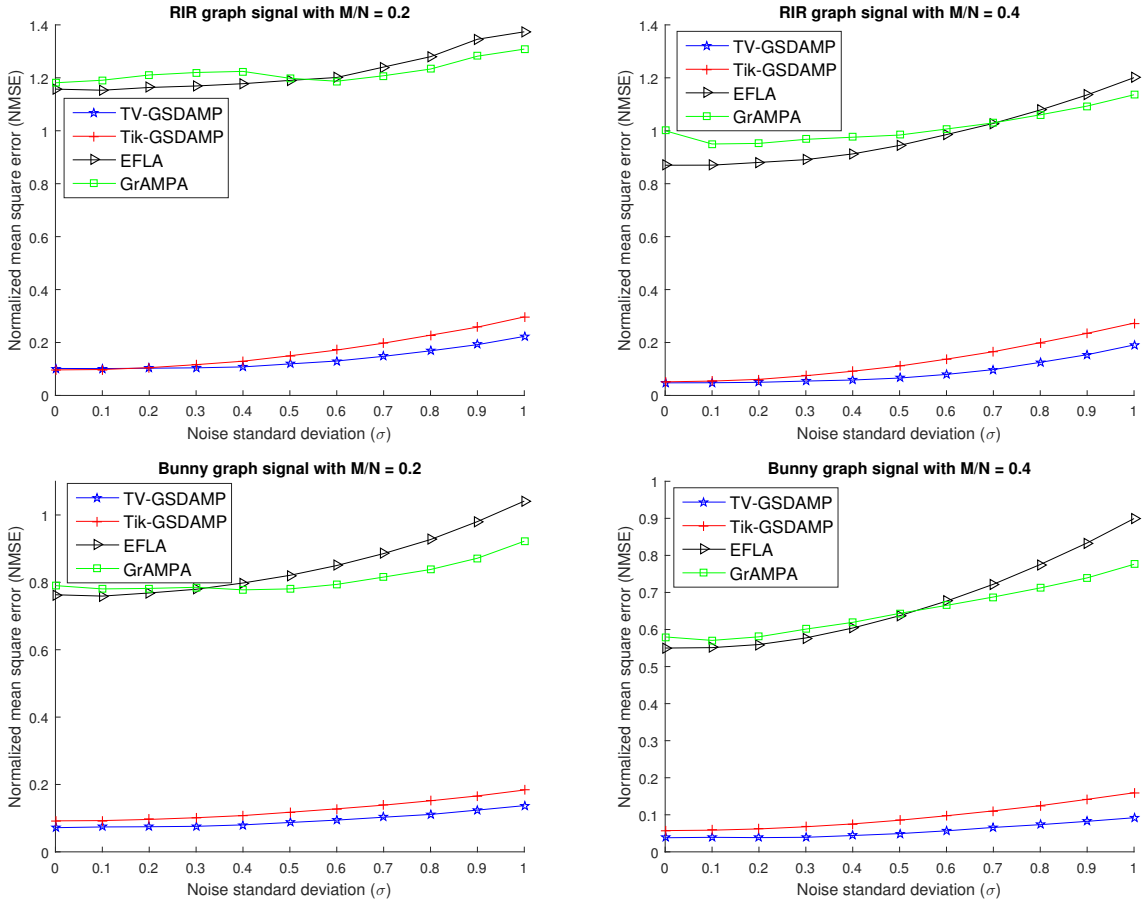


Figure 3.8. NMSE vs noise standard deviation σ for RIR and Bunny graphs, where the sampling rate M/N is set to 0.2 and 0.4.

3.4.3 Recovery performance regarding to LRR

Another figure of merit, beside the NMSE, is the LRR a_η (3.18) defined as the fraction of nodes $i \in \mathcal{V}$, for which the recovery error $|x_i - \hat{x}_i|$ does not exceed the threshold η, i.e.,

$$a_\eta = \mid \{i \in \mathcal{V}, \mid x_i - \hat{x}_i \mid \leq \eta\} \mid /N, \tag{3.18}$$

where, x_i is the original signal value at node i, and $\hat{x}_i$ is the recovered signal value.

To obtain the value of a_η, first we round the recovered signal value to the nearest signal value in the signal value set. In the Amazon product rating data-set, since the rating of the products come from the set $x_i \in 1/2\,\{0, 1, 2, \ldots, 10\}$, we have $\eta \in 1/2\,\{0, 1, 2, \ldots, 10\}$. However, in RIR, Bunny and Minnesota road graphs the signals come from the set $x_i \in \{-1, 1\}$, hence we have $\eta \in \{0, 2\}$. In this chapter, we compare the exact recovery performance of the proposed algorithms with GrAMPA and EFLA algorithms for all investigated graph signals

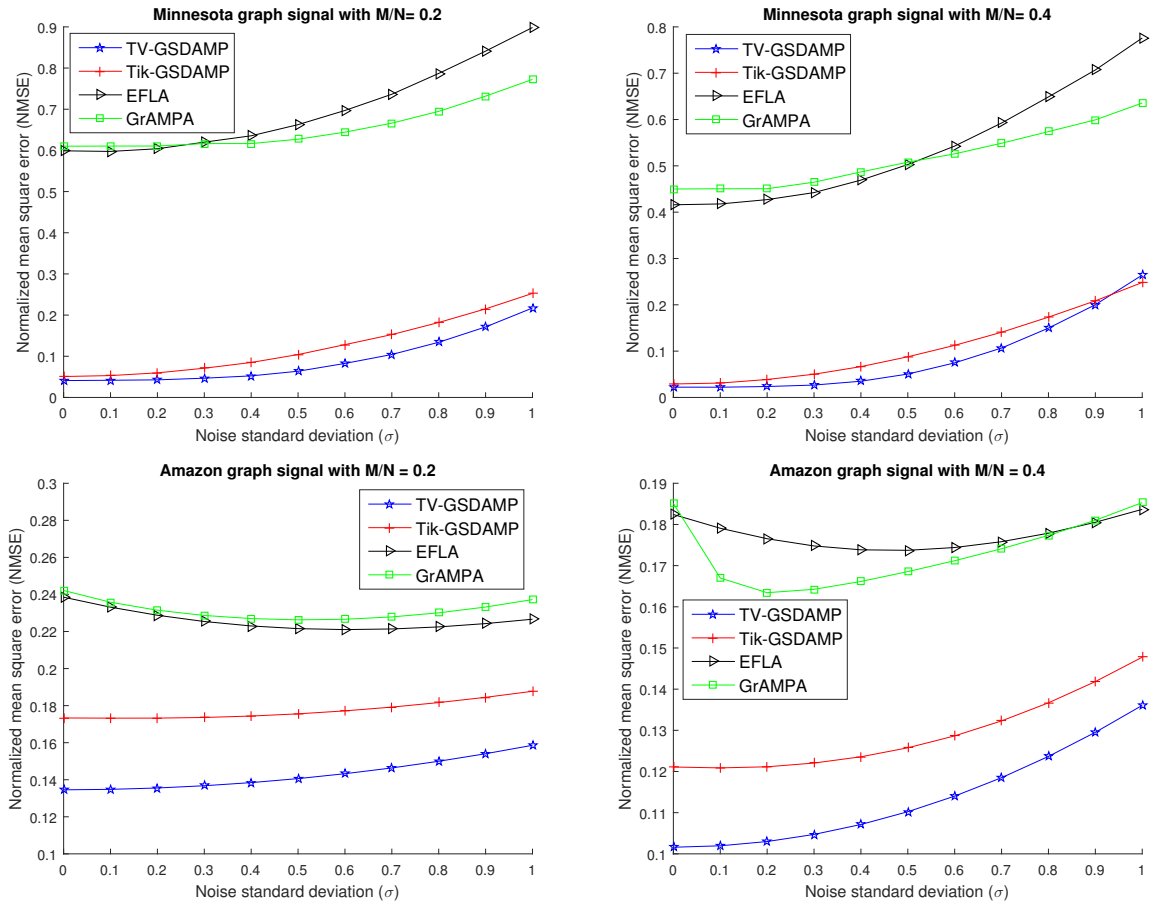


Figure 3.9. NMSE vs noise standard deviation σ for Minnesota and Amazon graphs, where the sampling rate M/N is set to 0.2 and 0.4.

where the value for η is set to zero, i.e., we compare the number of the samples that exactly recovered exploiting each recovery algorithm.

The LRR of TV-GSDAMP, Tik-GSDAMP, EFLA and GrAMPA recovery methods over varying noise standard deviations σ is illustrated in Figure 3.10, where the sampling rate M/N is set to 0.2. As shown in Figure 3.10, for RIR, Bunny and Minnesota graph signals with the same graph value set i.e., $x_i \in \{1, -1\}$, even with low sampling rate $M/N = 0.2$, and high noise standard deviation $\sigma = 1$, TV-GSDAMP and Tik-GSDAMP recovery algorithms can recover more than 97% of the graph values ($a_{0\{TV-GSDAMP,Tik-GSDAMP\}} \geq 0.97$) which is at least 29% higher than the recovery performance of GrAMPA ($a_{0\{GrAMPA\}} \leq 0.75$) and more than 53% higher than the recovery performance of EFLA ($a_{0\{EFLA\}} \leq 0.63$) with the same settings. For noise standard deviation $\sigma = 0$ and sampling rate ($M/N = 0.2$), the proposed algorithms recover 98% to 99% ($a_{0\{TV-GSDAMP,Tik-GSDAMP\}} \geq 0.98$) of the graph

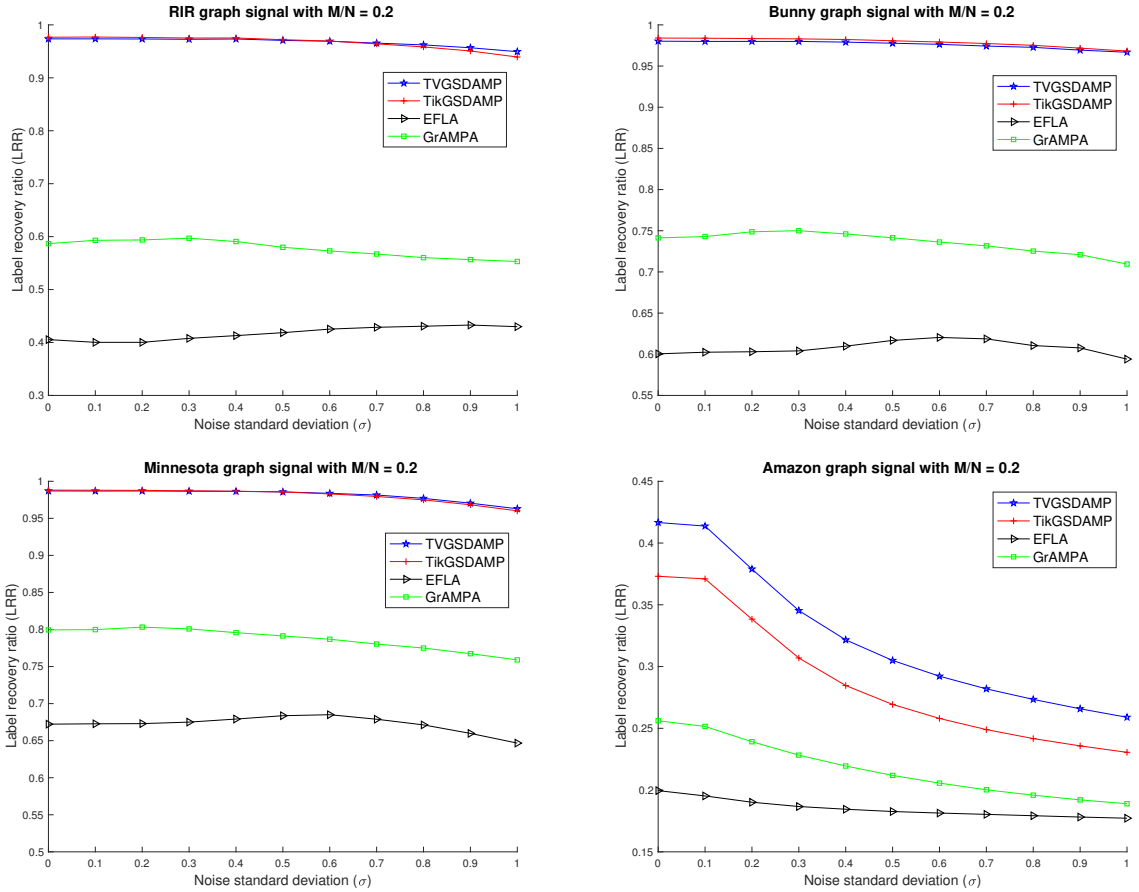


Figure 3.10. LRR vs noise standard deviation σ, where the sampling rate M/N is set to 0.2.

signals values which are again superior to the recovery performance of GrAMPA and EFLA recovery algorithms. One can also see from Figure 3.10 the superiority of the proposed algorithms in recovery of Amazon graph signals values, where the TV-GSDAMP recovery rate is 7% higher than the LRR of Tik-GSDAMP, 33% more than LRR of GrAMPA and 64% more than LRR of EFLA algorithm for the same sampling rate and noise standard deviation $\sigma = 1$. For the noiseless scenario the recovery percentage of the proposed algorithms is even much higher (at least twice) compared to the rest of the algorithms.

The LRR of TV-GSDAMP, Tik-GSDAMP, EFLA and GrAMPA recovery methods over varying sampling rate M/N is illustrated in Figure 3.11, where the noise standard deviation σ is set to 0. Intuitively, the LRR of all the recovery algorithms improve for higher sampling rates. In the noiseless setting with the maximum error threshold $\eta = 0$ and sampling rate $M/N = 0.95$, for all scrutinized graph signals the proposed algorithms are able to recover the correct ratings of more than 99% of the graph values. The other recovery algorithms are able to give the same result for the RIR, Bunny and Minnesota graphs. However,

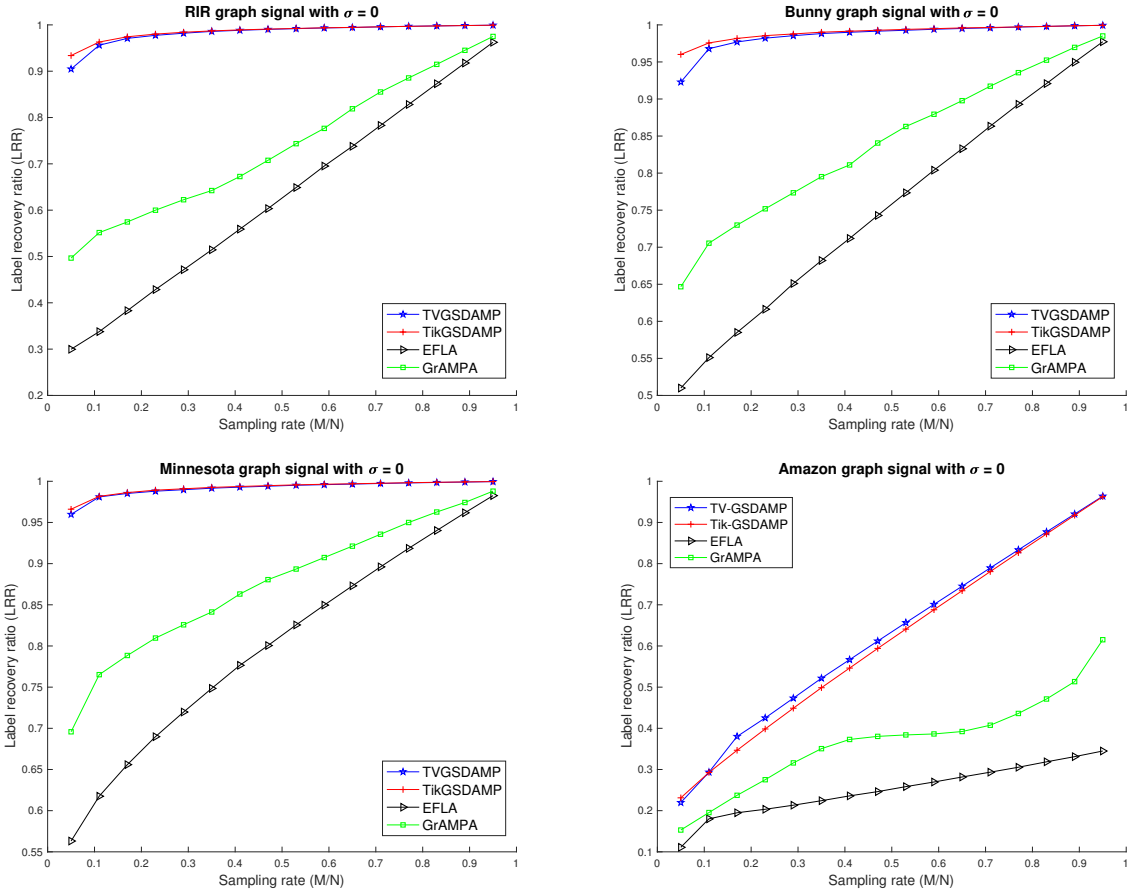


Figure 3.11. LRR vs sampling rate M/N, where the noise standard deviation σ is set to 0.

for Amazon product rating graph the LRR of GrAMPA is less than ($a_{0\{GrAMPA\}} \leq 0.65$) which is at least 85% higher than the LRR of EFLA ($a_{0\{EFLA\}} \leq 0.35$). Besides, in Amazon product rating graph when we set the sampling rate to 0.05, the TV-GSDAMP and Tik-GSDAMP algorithms are able to recover the correct ratings of more than 22% of the products ($a_{0\{TV-GSDAMP,Tik-GSDAMP\}} \geq 0.22$) which is 46% higher than LRR of GrAMPA ($a_{0\{GrAMPA\}} \leq 0.15$). The EFLA recovery algorithm is able to recover the correct ratings of 10% of graph values ($a_{0\{EFLA\}} = 0.1$).

Again incrementing the noise level harms the recovery performance, e.g., for noise standard deviation $\sigma = 0.5$, the recovery performance decreases for all of the compared algorithms (see Appendix B for more results). By increasing the noise standard deviation, the recovery slopes show that the TV-GSDAMP and Tik-GSDAMP algorithms are more robust to the noise compared to EFLA and GrAMPA algorithms. Even in the high noise standard

deviation regime and for all graph signals the proposed algorithms outperform both EFLA and GrAMPA in all scrutinized scenarios.

3.5 Conclusion

In this chapter we present a new AMP-based method for recovering smooth graph signals based on a preferably small number of noisy signal samples. We exploit the smoothness of typical graph signals occurring in many applications, such as Internet-based retailers or social network analysis. For the graph signal recovery, we combine graph signal denoisers i.e., total variation and Tikhonov denoisers, with the AMP framework. Supported by illustrative numerical experiments, we show that our proposed algorithms significantly outperform existing methods particularly for very low sampling rates in several relevant regimes.

Chapter 4

Graph signal recovery via iterative solvers

4.1 Introduction

This chapter focuses on the concept of graph signal recovery in more depth. Like previous chapter here in the considered system set up the graph signal recovery is formulated as a CS measurement, the graph signal values are obtained using CS recovery by Laplacian iterative methods. For the sake of comprehensiveness, some basic concepts of GSP are restated in this chapter.

As mentioned in the previous chapter, we are interested to reconstruct a graph signal from partially observed samples (possibly noisy), which is a key problem studied in GSP. Recovery algorithm uses the prior knowledge that the true graph signal is smooth with respect to the graph structure. A graph signal is called smooth, when signal values of connected nodes do not vary significantly. To indicate the smoothness of the graph signal, we use the notion of graph gradients. Several approaches to the graph signal recovery problem have been put forward. Recently, there have been some attempts to apply the AMP framework to the problem of the graph signal recovery like total variation AMP (TV-AMP) [103], GSDAMP [12], and GrAMPA [75]. Another wide family of graph recovery algorithms is obtained by convex optimization methods [104–106]. Within this class, methods based on Tikhonov regularization using the graph Laplacian quadratic form are appealing, since they amount to solve systems of linear equations involving the graph Laplacian [29].

Based on our work in [13], the contribution of this chapter is summarized as follows. We formulate the graph signal recovery problem as an optimization problem using Tikhonov regularization of noisy graph signals. The optimization problem consists of two quadratic

terms. The first term is to measure the fidelity of the recovery and the second term, which is the graph Laplacian quadratic form, measures the smoothness of the reconstructed graph signal. To solve the optimization problem, we first derive a closed-form solution that leads to a system of linear equations, which can be solved, e.g., by iterative techniques known from the literature such as the GS [107] or the BGS [108, 109] methods. We use those iterative techniques as recovery methods to solve our graph signal recovery problem and we also provide convergence criteria. Computationally more efficient methods exist (e.g. [110, 30]) to solve the linear systems we consider[1], but we don't use those advanced techniques in this work for simplicity of the implementation and because the conventional BGS method was sufficiently efficient for the simulations we conducted.

After we explain the system set up in Section 4.2. In the interest of clarity, we first explain the sampling of graph signal and formalize the graph signal recovery problem in Section 4.3. We will then discuss about iterative solvers in Section 4.4 and finally in Section 4.5 we will be discussing the results of numerical experiments.

4.2 System setup

We consider an undirected weighted graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{W})$, as illustrated in Figure 4.1, with node set $\mathcal{V} = \{1, \ldots, N\}$ and edge set $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$. We assume the graph to be simple, i.e., it does not have any self-loops. Thus, $(\ell, j) \in \mathcal{E}$ implies $\ell \neq j$ and $(j, \ell) \in \mathcal{E}$ (since the edges are undirected). The entries $W_{\ell j}$ (all non-negative) of the symmetric weight matrix $\mathbf{W} \in \mathbb{R}^{N \times N}$ quantify the strength of the connections in the graph. The neighbourhood of the node $\ell \in \mathcal{V}$ is defined as $\mathcal{N}_\ell := \{\, j \in \mathcal{V} \mid W_{\ell j} \neq 0 \,\}$. In particular, $W_{\ell j} \neq 0$ only if $(\ell, j) \in \mathcal{E}$, i.e., the support of the matrix $\mathbf{W}$ reflects the edge structure of the graph $\mathcal{G}$.

A graph signal $\mathbf{x} : \mathcal{V} \to \mathbb{R}^N$ defined on the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{W})$ is a labeling of the graph nodes with real numbers, i.e., each node $\ell \in \mathcal{V}$ is assigned a graph signal value $x_\ell \in \mathbb{R}$. We stack the graph signal values into a vector $\mathbf{x} \in \mathbb{R}^N$, whose ℓ^{th} entry is the graph signal value x_ℓ at node $\ell \in \mathcal{V}$. The graph signals arising in many important applications are smooth in the sense that the signal values x_ℓ, x_j for neighbouring nodes $\ell, j \in \mathcal{V}$ are similar in general, i.e., for $(\ell, j) \in \mathcal{E}$, $x_\ell \approx x_j$. We use the gradient of a graph signal at node ℓ as a smoothness metric [19] and define the global smoothness of the graph signal $\mathbf{x}$ around node ℓ

[1] Any linear system $\mathbf{Bx} = \mathbf{b}$ with the system matrix $\mathbf{B}$, symmetric and diagonally dominant (SDD) system can be transformed [111, 112] into a Laplacian system (of twice the size) for which very efficient solutions are discussed, some with a complexity that is only linear in the number of edges of the underlying graph [110, 30].

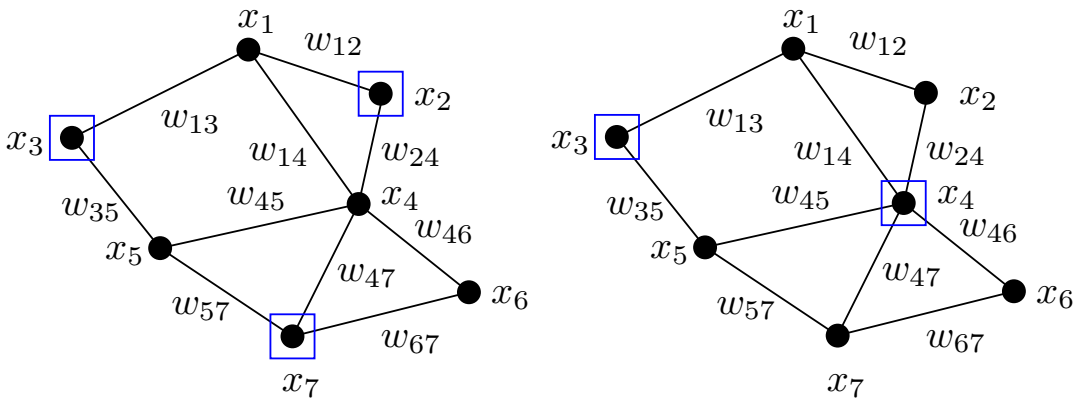


Figure 4.1. Undirected graph with signal components x_i and weights $\{w_{\ell j} = w_{j\ell}, \ell, j = 1, 2, ..., N\}$.

as

$$||\nabla_\ell \mathbf{x}||_2 = \left[\sum_{j\in\mathcal{N}_\ell} W_{\ell j}\left(x_j - x_\ell\right)^2\right]^{\frac{1}{2}}. \tag{4.1}$$

Here, $\nabla_\ell \mathbf{x}$ is the graph gradient of $\mathbf{x}$ at node ℓ. This measure is small when $\mathbf{x}$ has similar values at ℓ and its neighbouring nodes. The smoothness of graph signal $\mathbf{x}$ can be quantified by

$$S(\mathbf{x}) = \frac{1}{2}\sum_{\ell\in\mathcal{V}} ||\nabla_\ell \mathbf{x}||_2^2 = \frac{1}{2}\sum_{\ell\in\mathcal{V}}\sum_{j\in\mathcal{N}_\ell} W_{\ell j}\left(x_j - x_\ell\right)^2 \tag{4.2}$$

$$= \sum_{(\ell,j)\in\mathcal{E}} W_{\ell j}\left(x_j - x_\ell\right)^2 = \mathbf{x}^T\mathbf{L}\mathbf{x}\,, \tag{4.3}$$

where the combinatorial graph Laplacian matrix $\mathbf{L}$ is defined as

$$\mathbf{L} := \mathbf{D} - \mathbf{W}, \tag{4.4}$$

and the degree matrix $\mathbf{D}$ is a diagonal matrix whose ℓ^{th} diagonal element $D_{\ell\ell}$ is equal to the sum of the weights of all the edges connected to node ℓ, i.e.,

$$D_{\ell\ell} = \sum_{j\in\mathcal{V}} W_{\ell j}. \tag{4.5}$$

We normalize the value of $S(\mathbf{x})$ in order to compare the smoothness of various graph signals with each other

$$S'(\mathbf{x}) = \frac{S(\mathbf{x})}{\|\mathbf{x}\|_2^2}. \tag{4.6}$$

4.3 Recovery problem

The graph signal recovery problem arises, when we assume that only a few components x_j of the signal $\mathbf{x}$ are observed. In the next subsections we explain the concept of sampling of graph signal and then we will consider the problem of recovering a graph signal $\mathbf{x}$ with respect to the prior information, i.e., signal model and noisy or noiseless samples, that we already have.

4.3.1 Graph signal sampling

In this chapter, we consider the problem of recovering a smooth graph signal $\mathbf{x} = \{x_j, j = 1, 2, ..., N\}$ (true graph signal) from its noisy samples

$$y_k = x_{s_k} + n_k, \quad k \in \{1, 2, ..., M\}, \tag{4.7}$$

where

$$s_k \in \{1, 2, ..., N\} \quad \text{and} \quad s_k \neq s_{k'} \quad \text{for} \quad k \neq k', \tag{4.8}$$

and the set

$$\mathcal{S} = \{s_1, s_2, ..., s_M\}, \tag{4.9}$$

is, for $M < N$, a subset of the nodes set $\mathcal{V}$. We assume the number M of samples to be much smaller than the graph signal size N. By placing the observations y_k into the measurement vector $\mathbf{y} = \{y_k, k = 1, 2, ..., M\} \in \mathbb{R}^M$, we obtain a linear measurement model

$$\mathbf{y} = \mathbf{A}\mathbf{x} + \mathbf{n}, \tag{4.10}$$

where $\mathbf{n} = \{n_k, k = 1, 2, ..., M\}$, is the noise vector modeled as component-wise independent AWGN with zero-mean and variance σ^2, i.e., $n_k \sim \mathcal{N}(0, \sigma^2)$. The noise vector is to encompass the effects of measurement and modeling errors. The measurement matrix $\mathbf{A} \in \{1, 0\}^{M \times N}$ represents the sampling process. It mostly consists of zeros, with at most one non-zero entry in each column and exactly one non-zero entry in each row. Each row of $\mathbf{A}$ corresponds to selecting a graph signal value x_ℓ for some $\ell \in \mathcal{S}$. For example the measurement matrix $\mathbf{A}$ for the graph depicted in the left-side of Figure 4.1 is

$$\mathbf{A} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}, \tag{4.11}$$

where signal components x_2, x_3, x_7 are the selected (sampled) nodes of the graph (in Figure 4.1 sampled nodes are marked by squares around them). It should be noted that, *any* set of M (out of N) graph signal components can be chosen; moreover, the measurement matrix $\mathbf{A}$ is not unique, even if the same nodes are sampled, as, rows of $\mathbf{A}$ can be flipped.

4.3.2 Graph signal recovery

The graph signal recovery from the samples in (4.10) is (for $M < N$) an under-determined problem, but it can be solved by exploiting the weight matrix enforcing a smoothness criterion. The idea is to search for a graph signal vector $\hat{\mathbf{x}}$ that reproduces the observations "as accurate as possible" while at the same time the information from the weight matrix is used to infer the missing components.

Since we assume that the true graph signal $\mathbf{x}$ is smooth with respect to the underlying graph $\mathcal{G}$, we can use this prior information using regularization by (4.2) and (4.3). Our approach for recovering the true graph signal $\mathbf{x}$ from the noisy samples y_k, $k = 1, 2, ..., M$, from (4.7) is based on balancing the empirical error

$$E(\mathbf{x}) = \sum_{k=1}^{M} (y_k - x_{s_k})^2 = \|\mathbf{y} - \mathbf{A}\mathbf{x}\|_2^2, \tag{4.12}$$

with the smoothness $S(\mathbf{x})$ (defined in (4.2) and (4.3)) of the recovered signal $\mathbf{x}$. Thus, a graph signal recovery strategy is given by the optimization problem

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}\in\mathbb{R}^N} \underbrace{E(\mathbf{x}) + \lambda S(\mathbf{x})}_{\doteq F(\mathbf{x})} . \tag{4.13}$$

The first term in (4.13) requires the reconstructed signal $\mathbf{A}\mathbf{x}$ to be close to $\mathbf{y}$ and the second term enforces the smoothness of the solution. To determine the trade-off between accuracy $E(\mathbf{x})$ and smoothness $S(\mathbf{x})$ of the reconstruction $\hat{\mathbf{x}}$, the tuning parameter $\lambda > 0$ is introduced; a small parameter λ puts more emphasis on accuracy and less on smoothness (and vice versa). In order to recover the noisy graph signal from a small number of noisy samples we have to minimize the objective function $F(\mathbf{x})$ in (4.13), which is a linear combination of two quadratic functions of $\mathbf{x}$. The optimization in (4.13) has to be conducted over N-dimensions (the dimensionality of the graph signal vector $\mathbf{x}$).

In what follows, we first use a "scalar" approach to derive a solution, which is based on taking partial derivatives for the components; this reveals the structure of the problem and motivates an efficient method for recovery. A second vector-based approach allows for

compact statements, under which conditions a unique solution for the recovery problem exists.

A. **Scalar approach**

We write $F(\mathbf{x})$ in (4.13) according to

$$F(\mathbf{x}) = \sum_{k=1}^{M}(y_k - (\mathbf{A}\mathbf{x})_k)^2 + \lambda\,\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1,j\neq i}^{N} W_{ij}(x_i - x_j)^2\,, \tag{4.14}$$

with $(\mathbf{A}\mathbf{x})_k$ denoting the k^{th} component of the vector $\mathbf{A}\mathbf{x}$. The summation in the right double-sum was expanded over all nodes of the set $\mathcal{V} = \{1, 2, ..., N\}$; only for the nodes in the neighbourhood $\mathcal{N}_i$ of node i, the entries W_{ij} of the weight matrix will be non-zero, so the result is equivalent to (4.2). We take partial derivatives for the graph signal components x_ℓ, $\ell = 1, 2, ..., N$

$$\frac{\partial F(\mathbf{x})}{\partial x_\ell} = -2\sum_{k=1}^{M}(y_k - (\mathbf{A}\mathbf{x})_k)A_{k\ell} \tag{4.15}$$

$$+\,\frac{\lambda}{2}\Big(\sum_{j=1,j\neq l}^{N} 2W_{\ell j}(x_\ell - x_j) + \sum_{i=1,i\neq \ell}^{N}(-2)W_{i\ell}(x_i - x_\ell)\Big)$$

$$= -2\sum_{k=1}^{M}\big(y_k - \sum_{n=1}^{N} A_{kn}x_n\big)A_{k\ell} + 2\,\lambda\sum_{j=1,j\neq l}^{N} W_{\ell j}(x_\ell - x_j), \tag{4.16}$$

where the symmetry $W_{\ell j} = W_{j\ell}$ was used. Then we set the right-side of (4.16) to zero to find the extremum:

$$\begin{aligned}
\sum_{k=1}^{M} A_{k\ell}\,y_k &= \sum_{n=1}^{N} x_n \sum_{k=1}^{M} A_{kn}A_{k\ell} \;+\; \lambda \sum_{j=1,j\neq l}^{N} W_{\ell j}(x_\ell - x_j),\\
&= \sum_{n=1,n\neq \ell}^{N} x_n \sum_{k=1}^{M} A_{kn}A_{k\ell} \;+\; x_\ell \sum_{k=1}^{M} A_{k\ell}A_{k\ell} \;+\; \lambda \sum_{j=1,j\neq l}^{N} W_{\ell j}(x_\ell - x_j),\\
&= \sum_{n=1,n\neq \ell}^{N} x_n \sum_{k=1}^{M} A_{kn}A_{k\ell} \;+\; x_\ell \sum_{k=1}^{M} A_{k\ell}A_{k\ell} \;+\; x_\ell\,\lambda \sum_{j=1,j\neq l}^{N} W_{\ell j} \;-\; \lambda \sum_{j=1,j\neq l}^{N} W_{\ell j}x_j,\\
&= \sum_{j=1,j\neq \ell}^{N} x_j \sum_{k=1}^{M} A_{kj}A_{k\ell} \;-\; \lambda \sum_{j=1,j\neq l}^{N} W_{\ell j}x_j \;+\; x_\ell \sum_{k=1}^{M} A_{k\ell}A_{k\ell} \;+\; x_\ell\,\lambda \sum_{j=1,j\neq l}^{N} W_{\ell j},
\end{aligned}$$

$$= \sum_{j=1,j\neq\ell}^{N} x_j \sum_{k=1}^{M} A_{kj}A_{k\ell} + \sum_{j=1,j\neq l}^{N} x_j\big(-\lambda W_{\ell j}\big) + x_\ell \sum_{k=1}^{M} A_{k\ell}A_{k\ell} + x_\ell\lambda \sum_{j=1,j\neq l}^{N} W_{\ell j},$$
$$= \sum_{j=1,j\neq\ell}^{N} x_j\Big(\sum_{k=1}^{M} A_{kj}A_{k\ell} + \big(-\lambda W_{\ell j}\big)\Big) + x_\ell \sum_{k=1}^{M} A_{k\ell}A_{k\ell} + x_\ell\lambda \sum_{j=1,j\neq l}^{N} W_{\ell j}.$$

We obtain the system of linear equations

$$\underbrace{\sum_{k=1}^{M} A_{k\ell}\, y_k}_{b_\ell} = \sum_{j=1,j\neq\ell}^{N} x_j \underbrace{\Big(\sum_{k=1}^{M} A_{kj}A_{k\ell} - \lambda W_{\ell j}\Big)}_{C_{\ell j}} + x_\ell \underbrace{\Big(\sum_{k'=1}^{M} A_{k'\ell}^2 + \sum_{j'=1,j'\neq l}^{N} \lambda W_{\ell j'}\Big)}_{C_{\ell\ell}}, \tag{4.17}$$

for $\ell = 1, 2, ..., N$, with the coefficients b_ℓ and $C_{\ell j}$ that can be computed from the given measurements y_k, the measurement matrix coefficients A_{kj} and the weight matrix coefficients $W_{\ell j}$. It should be noted that the system of equations (4.17) applies for *any* sampling matrix $\mathbf{A} \in \mathbb{R}^{M\times N}$. The system may not have a unique solution (which would mean unique recovery is impossible), so it needs to be clarified when a solution is possible.

B. **Vector-based approach**

The objective function in (4.13) can (see (4.3)) be equivalently written as

$$F(\mathbf{x}) = \|\mathbf{y} - \mathbf{A}\mathbf{x}\|_2^2 + \lambda\, \mathbf{x}^T\mathbf{L}\mathbf{x} \tag{4.18}$$
$$= \big(\mathbf{y} - \mathbf{A}\mathbf{x}\big)^T\big(\mathbf{y} - \mathbf{A}\mathbf{x}\big) + \lambda\, \mathbf{x}^T\mathbf{L}\mathbf{x} \tag{4.19}$$
$$= \mathbf{y}^T\mathbf{y} - 2\,\mathbf{y}^T\mathbf{A}\,\mathbf{x} + \mathbf{x}^T\Big(\mathbf{A}^T\mathbf{A} + \lambda\mathbf{L}\Big)\mathbf{x}\,. \tag{4.20}$$

The gradient, taken for the unknown vector $\mathbf{x}$, set to zero leads to

$$\underbrace{\mathbf{A}^T\mathbf{y}}_{\mathbf{b}} = \underbrace{(\mathbf{A}^T\mathbf{A} + \lambda\mathbf{L})}_{\mathbf{C}}\mathbf{x}, \tag{4.21}$$

with the definitions of the column vector $\mathbf{b} = (b_1, b_2, ..., b_N)^T$ and the matrix $\mathbf{C} = \{C_{\ell j}, \ell, j = 1, 2, ..., N\}$ inserted, the solution (4.21) equals the solution in (4.17).

The solution of linear system (4.21) which is equivalent to the optimal solution of (4.13) is

$$\hat{\mathbf{x}} = \mathbf{C}^{-1}\mathbf{b}. \tag{4.22}$$

It is clear from (4.22) that the matrix $\mathbf{C}$ has to be invertible, for a unique solution of (4.21) (and, hence, of (4.17)) to exist.

C. Invertibility and positive definiteness of the system matrix

Since we consider high-dimensional problems with large N to solve the systems of linear equations, an inversion of the matrix $\mathbf{C}$ as well as standard approaches like Gaussian elimination or QR decomposition are not preferred due to the complexity. Therefore, here we consider an iterative approach – the well-known Gauss-Seidel method – that is immediately suggested by (4.17). The idea is to solve the problem for $\hat{x}_\ell$ and compute an update for it while all the other components $\hat{x}_j$, $j \neq \ell$ are kept fixed. This approach can be done alternatively for all of the components. The question is if this process converges to a unique solution, and in Section 4.4 convergence statements for this iterative process known from the literature are given. They require such a unique solution to exist, which in turn means that the system matrix $\mathbf{C}$ has to be invertible. Therefore, we now consider our specific sampling matrices $\mathbf{A}$ which consists mostly of zeros and have at most one "1"-entry in each column and exactly one "1"-entry in each row; an example is given in (4.11).

The $N \times N$ matrix $\mathbf{A}^T\mathbf{A}$ then consists almost entirely of zero-entries, with "1"-components in those locations on the main diagonal which correspond to the index of the nodes that are sampled. The matrix $\mathbf{A}^T\mathbf{A}$ stays the same, when the rows of $\mathbf{A}$ are flipped, this can be easily verified by inspection of examples. Hence, in the system matrix

$$\mathbf{C} = \mathbf{A}^T\mathbf{A} + \lambda\mathbf{L}, \tag{4.23}$$

the selected sampling matrix adds the values of "1" to the main diagonal of the $\lambda\mathbf{L}$ matrix in the locations that correspond to the sampled graph signal components. As an example in (4.11), we have

$$\mathbf{A}^T\mathbf{A} = \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix},$$

$$\mathbf{A}^T\mathbf{A} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}. \tag{4.24}$$

The graph Laplacian $\mathbf{L}$ itself is a singular matrix, as by definition at least one arbitrary row can be made "all-zero" by adding all the other rows to it. Hence, it is clear that $\mathbf{L}$ has at least one eigenvalue that is "zero" and it is also known [19] that $\mathbf{L}$ has only non-negative eigenvalues. However, "zero" appears as an eigenvalue with a multiplicity that equals the number of separable subgraphs [19]. These are the subsets of the nodes in the graph that are intra-connected but have no connections (through non-zero entries in the weight matrix) to other parts of the graph. In the sampling matrix selection, we have to make sure that all of the zero eigenvalues of $\mathbf{L}$ disappear by adding "1"-values to the main diagonal in the right places while with the non-zero eigenvalues, system matrix $\mathbf{C}$ will be invertible.

If we assume that $\mathbf{L}$ corresponds to a connected graph the matrix has exactly one eigenvalue that is zero. As the nodes can be arbitrarily labeled, without loss of generality we assume that only node $\ell = 1$ is sampled. Then for $\lambda > 0$ we have

$$\frac{\mathbf{C}}{\lambda} = \begin{pmatrix} \frac{1}{\lambda} + D_{11} & -W_{12} & -W_{13} & \dots & -W_{1N} \\ -W_{12} & D_{22} & -W_{23} & \dots & -W_{2N} \\ \dots & \dots & \dots & \dots & \dots \\ -W_{1N} & -W_{2N} & -W_{3N} & \dots & D_{NN} \end{pmatrix}, \tag{4.25}$$

where the diagonal terms $D_{\ell\ell}$, $\ell = 1, 2, ..., N$, are defined in (4.5). When we add the rows $2, 3, ..., N$ to the first, we obtain

$$\frac{1}{\lambda}\mathbf{C} = \begin{pmatrix} \frac{1}{\lambda} & 0 & 0 & \dots & 0 \\ -W_{12} & D_{22} & -W_{23} & \dots & -W_{2N} \\ \dots & \dots & \dots & \dots & \dots \\ -W_{1N} & -W_{2N} & -W_{3N} & \dots & D_{NN} \end{pmatrix}. \tag{4.26}$$

Now we can make the matrix "upper triangular" by successively scaling (with strictly positive factors) the elements $D_{\ell\ell} > 0$ for $\ell = 2, 3, ..., N$ on the main diagonal appropriately to cancel all elements of the columns below the main diagonal in the ℓ^{th} column. This is possible, because the terms $D_{\ell\ell}$ on the main diagonal are by definition in (4.5) guaranteed to be positive, as they contain the sum of all the weights in a column of $\mathbf{W}$ and we assumed the graph is connected. The determinant of $\mathbf{C}$ (using the constructed upper triangular matrix) equals the product of the elements on the main diagonal, i.e.,

$$\det(\mathbf{C}) = \lambda\big(\frac{1}{\lambda}\big)\big(D_2\big)\prod_{\ell=3}^{N}\big(\alpha_\ell D_{\ell\ell}\big), \tag{4.27}$$

where we have $0 < \alpha_\ell < \infty$. The sum of the columns of a connected weight matrix must contain at least one positive component $W_{\ell j}$ and all the elements off the main diagonal are negative. Hence, a positive factor must be chosen to cancel those elements; the factor α_ℓ is the result of such repeated scaling operations. As all the terms in (4.27) are positive, $\det(\mathbf{C}) > 0$ is guaranteed. So, we need to sample at least one component in a connected graph to make $\mathbf{C}$ invertible. Sampling more nodes does no harm, as then the positive main diagonal elements are made even larger.

In the more general case, when the graph consists of several subgraphs , the problem can be decomposed into several independent problems. Hence, at least one signal component must be sampled from each of the disconnected subgraphs, and then (and only then) the matrix $\mathbf{C}$ invertibility is guaranteed[2].

For the convergence of the iterative methods discussed in the next section, results are available from the literature that the system matrix $\mathbf{C}$ need to be positive definite. The matrix $\mathbf{C}$ is real and symmetric by definition according to (4.23) and by the definition of the graph Laplacian in (4.4). But it should be noted that $\mathbf{C}$ is not a Laplacian matrix

[2]If there is no sampled component in a disconnected subgraph, a row in $\mathbf{C}$ can be made zero (as the $1/\lambda$-term would be missing, see (4.25)) and then $\mathbf{C}$ would not be invertible.

as the row- and column-sums are non-zero. From Sylvester's criterion [113], [114, Section 4.2.1], it is known that a real and symmetric matrix is positive definite if and only if all its leading principal minors are positive. This property can be shown for the matrix (4.25) to hold by considering the upper triangular form of the system matrix $\mathbf{C}$. Since for a triangular matrix, the ℓ^{th} leading principal minor is the product of the first ℓ elements on the main diagonal and all the elements on the main diagonal of the upper triangular form of $\mathbf{C}$ are positive, the matrix $\mathbf{C}$ in (4.25) must be positive definite. If more than one measurement is taken, there will be more diagonal elements in the matrix $\mathbf{C}$ that a "1" is added to, so the elements will remain positive and the same argumentation as above will apply. Hence, if at least one measurement is taken and the graph does not contain disconnected subgraphs, then the matrix $\mathbf{C}$ will be positive definite.

4.4 Iterative graph signal recovery

In what follows, we consider graph signals defined over the connected graphs. This assumption does not incur any loss of generality. Indeed, if the graph is composed of several not interconnected subgraphs, the recovery problem (4.13) and the associated linear system of equation (4.21) would split into independent subproblems, one for each subgraph.

The general system of linear equations in (4.17) can be specialized to the specific sampling matrices. As there is at most one "1" in each column of $\mathbf{A}$, we have $A_{kj}A_{k\ell} = 0$ for $j \neq \ell$, so

$$C_{\ell j} = -\lambda W_{\ell j}\,, \tag{4.28}$$

for $\ell, j = 1, 2, ..., N$.

If there is *no* measurement of the component x_ℓ, i.e. $\ell \notin \mathcal{S}$, we have $\sum_{k'=1}^{M} A_{k'\ell}^2 = 0$ because there is no non-zero element in column ℓ in any row of $\mathbf{A}$. If there is a non-zero element in some row of the column ℓ, i.e., $\ell \in \mathcal{S}$, regarding to the construction of $\mathbf{A}$ it will only be one (non-zero), and its value will be "1". Hence,

$$C_{\ell\ell} = \begin{cases} 1 + \lambda D_{\ell\ell} & \text{for } \ell \in \mathcal{S} \\ \lambda D_{\ell\ell} & \text{for } \ell \notin \mathcal{S} \end{cases}, \tag{4.29}$$

for $\ell = 1, 2, ..., N$, where

$$D_{\ell\ell} = \sum_{j=1, j\neq\ell}^{N} W_{\ell j}\,. \tag{4.30}$$

For the left-side of the linear system in (4.17), we obtain

$$b_\ell = \sum_{k-1}^{M} A_{k\ell}\, y_k = \begin{cases} y_k & \text{for } \ell \in \mathcal{S} \text{ with } A_{k\ell} = 1 \\ 0 & \text{for } \ell \notin \mathcal{S} \end{cases} . \tag{4.31}$$

4.4.1 Gauss-Seidel iterative solver

In order to obtain the recovered signal $\hat{\mathbf{x}}$, we have to solve the system of linear equations in (4.17). An iterative method is suggested to compute an update for $\hat{x}_\ell$ and keep the other components $\hat{x}_j$, $j \neq \ell$ fixed: a refinement, which is used below and is known as the GS method in the literature (e.g. [107, 109]), is to immediately use the recently updated values of nodes when other nodes are updated. The GS method then constructs from (4.17) a sequence $\hat{\mathbf{x}}^{(t)}$ by iterating, for $t = 1, 2, \ldots$, the node-wise updates

$$\hat{x}_\ell^{(t)} = \frac{1}{C_{\ell\ell}}\big(b_\ell - \sum_{j=1}^{\ell-1} C_{\ell j}\hat{x}_j^{(t)} - \sum_{j=\ell+1}^{N} C_{\ell j}\hat{x}_j^{(t-1)}\big)\,, \tag{4.32}$$

where for $\ell = 1$ the first sum is not computed, because (as a convention) the upper sum-index $l - 1$ is smaller than the lower one, and similarly for $\ell = N$, the second sum is not computed. Now we use (4.28), (4.29); this leads to

$$\hat{x}_\ell^{(t)} = \begin{cases} \dfrac{1}{1+\lambda D_{\ell\ell}}\Big(y_\ell + \lambda\big(\sum_{j=1}^{\ell-1} W_{\ell j}\hat{x}_j^{(t)} + \sum_{j=\ell+1}^{N} W_{\ell j}\hat{x}_j^{(t-1)}\big)\Big) & \ell \in \mathcal{S} \\ \dfrac{1}{D_{\ell\ell}}\big(\sum_{j=1}^{\ell-1} W_{\ell j}\hat{x}_j^{(t)} + \sum_{j=\ell+1}^{N} W_{\ell j}\hat{x}_j^{(t-1)}\big) & \ell \notin \mathcal{S} \end{cases} . \tag{4.33}$$

As a stopping criterion for the iterations, we use a pre-specified threshold ε on the squared error between the left- and right-side of (4.17), i.e.,

$$E = \sum_{\ell=1}^{N}\Big(b_\ell - \sum_{j=1}^{N} C_{\ell j}\,\hat{x}_j^{(t)}\Big)^2,$$

into which (4.28) and (4.29) are inserted. The result can be written as given in Line 9 of Algorithm 2, which summarizes the iterative scheme.

Algorithm 2 does *not* assume that the graph is sparse. If it is, the scheme in Algorithm 2 can be made much faster, because the sums in Line 5 and Line 7 can be limited to the neighbourhood $\mathcal{N}_\ell$ of the node x_ℓ (see 4.2) in which the weights $W_{\ell j}$ are non-zero: for a sparse weight matrix, that neighbourhood set will be small.

Algorithm 2: Graph signal recovery via GS method

Input: symmetric weighted $N \times N$ adjacency matrix $\mathbf{W}$, $M \times N$ sampling matrix $\mathbf{A}$, measurement vector $\mathbf{y} = (y_1, ..., y_M)^T$, parameter $\lambda > 0$ to balance accuracy and smoothness of the solution, error threshold $\varepsilon > 0$ for the stopping criterion

Initialization: set $t = 0$, compute $D_{\ell\ell} = \sum_{j=1}^{N} W_{\ell j}$, $b_\ell = \sum_{k=1}^{M} A_{k\ell} y_k$ and set $\hat{x}_\ell^{(0)} = 0$, for $\ell = 1, 2, ..., N$.

repeat

  $t := t + 1$

  for $\ell \in \mathcal{S}$ **do**

$$\hat{x}_\ell^{(t)} = \frac{y_\ell + \lambda \sum_{j=1}^{\ell-1} W_{\ell j}\, \hat{x}_j^{(t)} + \lambda \sum_{j=\ell+1}^{N} W_{\ell j}\, \hat{x}_j^{(t-1)}}{1+\lambda D_{\ell\ell}}$$

  end for

  for $\ell \notin \mathcal{S}$ **do**

$$\hat{x}_\ell^{(t)} = \frac{\sum_{j=1}^{\ell-1} W_{\ell j}\, \hat{x}_j^{(t)} + \sum_{j=\ell+1}^{N} W_{\ell j}\, \hat{x}_j^{(t-1)}}{D_{\ell\ell}}$$

  end for

(Note: if $\ell = 1$ the sum $\sum_{j=1}^{\ell-1}$ is not computed; and if $\ell = N$ the sum $\sum_{j=\ell+1}^{N}$ is not computed)

  Compute error

$$E = \sum_{\ell \in \mathcal{S}} \left((b_\ell - \hat{x}_\ell^{(t)}) - \lambda q_\ell \right)^2 + \sum_{\ell \notin \mathcal{S}} \left(\lambda q_\ell \right)^2$$

  with

$$q_\ell = D_{\ell\ell}\, \hat{x}_\ell^{(t)} - \sum_{j=1, j\neq\ell}^{N} W_{\ell j}\, \hat{x}_j^{(t)}$$

until $E < \varepsilon$

Output: recovered graph signal $\hat{\mathbf{x}} = \left(\hat{x}_1^{(t)}, \hat{x}_2^{(t)}, ..., \hat{x}_N^{(t)}\right)^T$.

4.4.2 Block Gauss-Seidel iterative solver

The *Block* GS (BGS) method [108, Chap 10] generalizes the GS method by updating during each iteration whole blocks of the current estimate $\hat{\mathbf{x}}^{(t)}$ in one step instead of single entries (as in (4.33)). This allows for parallel computations and, hence, significant increase in speed, even though the complexity of one update step may appear to be higher for BGS in general. However, a major advantage of BGS is that it typically requires fewer iterations to reach a given solution accuracy. The BGS method is based on partitioning the system matrix $\mathbf{C}$, solution vector $\mathbf{x}$ and vector $\mathbf{b}$ of (4.21), into p blocks according to

$$\mathbf{C} = \begin{pmatrix} \mathbf{C}_{11} & \mathbf{C}_{12} & \dots & \mathbf{C}_{1p} \\ \mathbf{C}_{21} & \mathbf{C}_{22} & \dots & \mathbf{C}_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{C}_{p1} & \mathbf{C}_{p2} & \dots & \mathbf{C}_{pp} \end{pmatrix}, \ \mathbf{x} = \begin{pmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_p \end{pmatrix}, \ \mathbf{b} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_p \end{pmatrix}. \tag{4.34}$$

The iterations in the BGS method for solving (4.21) are defined by generalizing (4.33); the scheme is stated in Algorithm 3.

Algorithm 3: Smooth graph signal recovery via BGS

Input: $\mathbf{x}^0 \leftarrow 0$, $\mathbf{C} = \mathbf{A}^T\mathbf{A} + \lambda\mathbf{L}$, $t = 0$, number of partitions p, noisy samples $\{y_\ell\}_{\ell\in\mathcal{S}}$, sampling set $\mathcal{S}$, and parameter λ
repeat
  for $\ell = 1 : p$ **do**
    $\tilde{\mathbf{b}} \leftarrow (\mathbf{b})_\ell - \sum_{j=1}^{\ell-1} \mathbf{C}_{\ell j}\hat{\mathbf{x}}_j^{(t)} - \sum_{j=\ell+1}^{N} \mathbf{C}_{\ell j}\hat{\mathbf{x}}_j^{(t-1)}$
    Solve $\mathbf{C}_{\ell\ell}\hat{\mathbf{x}}_\ell = \mathbf{b}_\ell$ using (4.33) to obtain $\hat{x}_\ell^{(t)}$
  end for
  $t \leftarrow t + 1$
until the stopping criterion is met.
Output: recovered graph signal $\hat{\mathbf{x}} = \hat{\mathbf{x}}_\ell^{(t-1)}$.

The convergence of Algorithm 2 and Algorithm 3 will be discussed in Section 4.4.3.

4.4.3 Convergence criteria

In order to proof the convergence of the sequence $\hat{\mathbf{x}}^{(t)}$ $t = 1, 2, \ldots$, generated by both the point-wise and the block GS Algorithm, we consider the coefficient matrix $\mathbf{C}$ in (4.21). The diagonal entries of coefficient matrix $\mathbf{C}$ are obtained from (4.29) and its off diagonal entries

are given by $C_{\ell j} = -\lambda W_{\ell j}$. With the definition of irreducibly diagonally dominant matrices from [107, Def 4.5], the ℓ^{th} row of the matrix $\mathbf{C}$ is either strictly dominant for $\ell \in \mathcal{S}$, as

$$\sum_{j \in V, \ell \neq j} \mid \lambda W_{\ell j} \mid < 1 + \lambda D_{\ell \ell}, \tag{4.35}$$

or weakly dominant for $\ell \notin \mathcal{S}$, as

$$\sum_{j \in V, \ell \neq j} \mid \lambda W_{\ell j} \mid \leq \lambda D_{\ell \ell}. \tag{4.36}$$

This makes the matrix $\mathbf{C}$ irreducibly diagonally dominant. From [107, Theorem 4.9], if the matrix $\mathbf{C}$ is an irreducibly diagonally dominant matrix, then $\hat{\mathbf{x}}$ generated by Algorithm 2 or Algorithm 3 converges to the unique solution.

4.5 Numerical results

In order to assess the accuracy of the Laplacian solvers, we compare the efficiency of BGS-recovery algorithm with Tik-GSDAMP [12] graph signal recovery method along with the OPT-recovery method which is obtained by solving (4.22) using matrix inversion. We apply these recovery methods to various connected smooth graph signals, such as signals on a RIR graph, signals on a Bunny graph and etc. The detailed description of the compared graphs is in Section 3.4.

In the simulations, we exploit a random sampling method for selecting M signal samples x_ℓ and add zero-mean AWGN noise with variance σ^2. Thus, we obtain a measurement vector $\mathbf{y}$ conforming to the model (4.10). In order to accelerate the recovery of $\mathbf{x}$ from the noisy measurements (4.7), we apply Algorithm 3 using a partitioning of the graph into $p = 40$ blocks of equal size for the Amazon product rating graph signal. However, due to the small size of the RIR, Bunny and Minnesota graph signal we set the number of partitions to $p = 4$. The stopping criteria used for Algorithm 3 is either it reaches the maximum number of 50 iterations or the relative progress saturates according to $\frac{\|\hat{\mathbf{x}}^{(t)} - \hat{\mathbf{x}}^{(t-1)}\|_2}{\|\hat{\mathbf{x}}^{(t)}\|_2} \leq 10^{-2}$. To have an accurate and reliable comparison of the recovery schemes, we use exactly the same settings for all of the recovery schemes i.e, the same number of samples M and noise standard deviation σ.

In Figure 4.2 and Figure 4.3, we investigate the optimal value of the regularization parameter λ, i.e., λ_{opt}. We run both OPT- and BGS-recovery methods for different λ values (cf. (4.13)). Our goal is to find the optimum value of λ by adopting a two-fold search strategy which yields the best performance in terms of the NMSE over different noise standard

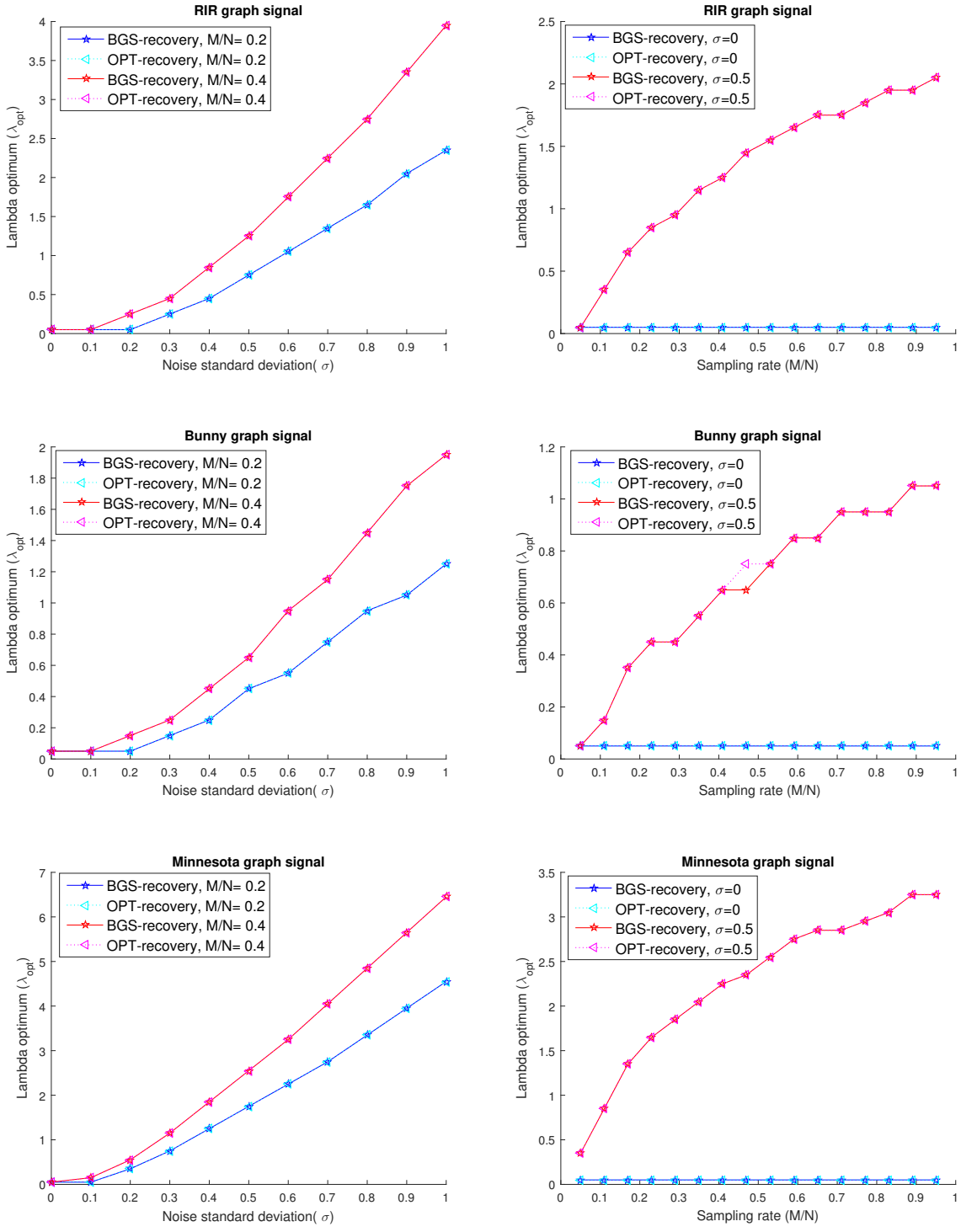


Figure 4.2. The optimum value of λ over varying noise standard deviations σ (left), and varying sampling rates M/N (right) for both BGS- and OPT-recovery algorithms.

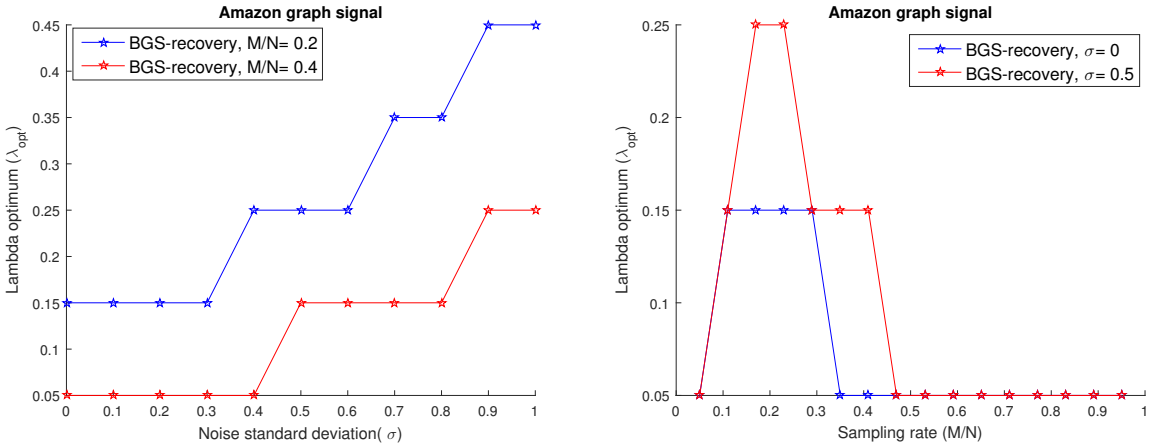


Figure 4.3. The optimum value of λ for Amazon product rating graph over varying noise standard deviations σ (left), and varying sampling rates M/N (right).

deviations σ and various sampling rates M/N. Note that, for RIR, Bunny and Minnesota graph signals we check the optimum value of λ for both BGS- and OPT-recovery algorithms. However, due to the large value of N (graph signals dimension) and high complexity of matrix inversion in Amazon product rating graph signal it is not feasible to use the OPT-recovery algorithm to reconstruct the graph signal.

Figure 4.2 illustrates that the BGS- and OPT-recovery algorithms deliver exactly the same values for the λ_{opt}. It also shows that the value of λ_{opt} increases for a growing value of noise standard deviation σ. In a high noise standard deviation regime, the value of sampled signals are unreliable and the recovery algorithms exhibit better performance by putting higher emphasize on the smoothness term of (4.13). In a signal with $\sigma = 0$, the sampling is the main cause of the observed signal imperfection. In such case, since the sampled signal values are equal to the original signal values the fidelity term of (4.13) plays more on the recovery of signal. In signal with $\sigma = 0.5$, the sampled signal values deviate from the original ones due to the effect of the noise. In this case, the difference in the signal values of two adjacent nodes is considered as the effect of noise and is removed by the smoothness term. Hence, in a smooth graph signal by increasing the value of sampling rate the algorithms employ a higher value for λ_{opt} to enforce the recovered signal to have similar values for neighbouring nodes.

Interestingly, the values of λ_{opt} in Figure 4.3 show a different trend for the Amazon product rating graph signal compared to the other graph signals. The reason for such trend is that compare to the other graph signals the Amazon dataset is less smooth, i.e., $S'_{Amazon}(\mathbf{x}) = 0.54$, while the smoothness factor of the the rest of the graph signals are $S'_{RIR}(\mathbf{x}) = 0.08$, $S'_{Bunny}(\mathbf{x}) = 0.18$, and $S'_{Minnesota}(\mathbf{x}) = 0.06$. When we have a highly smooth graph signal like RIR, Bunny or Minnesota graph signals, we expect that signal values do

not differ to much in the neighbouring nodes. Hence, in case $\sigma = 0.5$ by increasing the sampling rate the value of λ_{opt} will increases as well. In a less smooth signal like Amazon by increasing the sampling rate at first (when we have very low sampling rate), we observe an increase in the value of λ_{opt} until a certain point which further increment of sampling rate will decrease that. At this point, we have sufficient amount of samples in which recovery of signal without considering the effect of smoothness term will cause a higher performance. Hence, in the non-smooth graph in case $\sigma = 0.5$ and even for larger values of sampling rate it makes sense to not put high emphasize on the smoothness term.

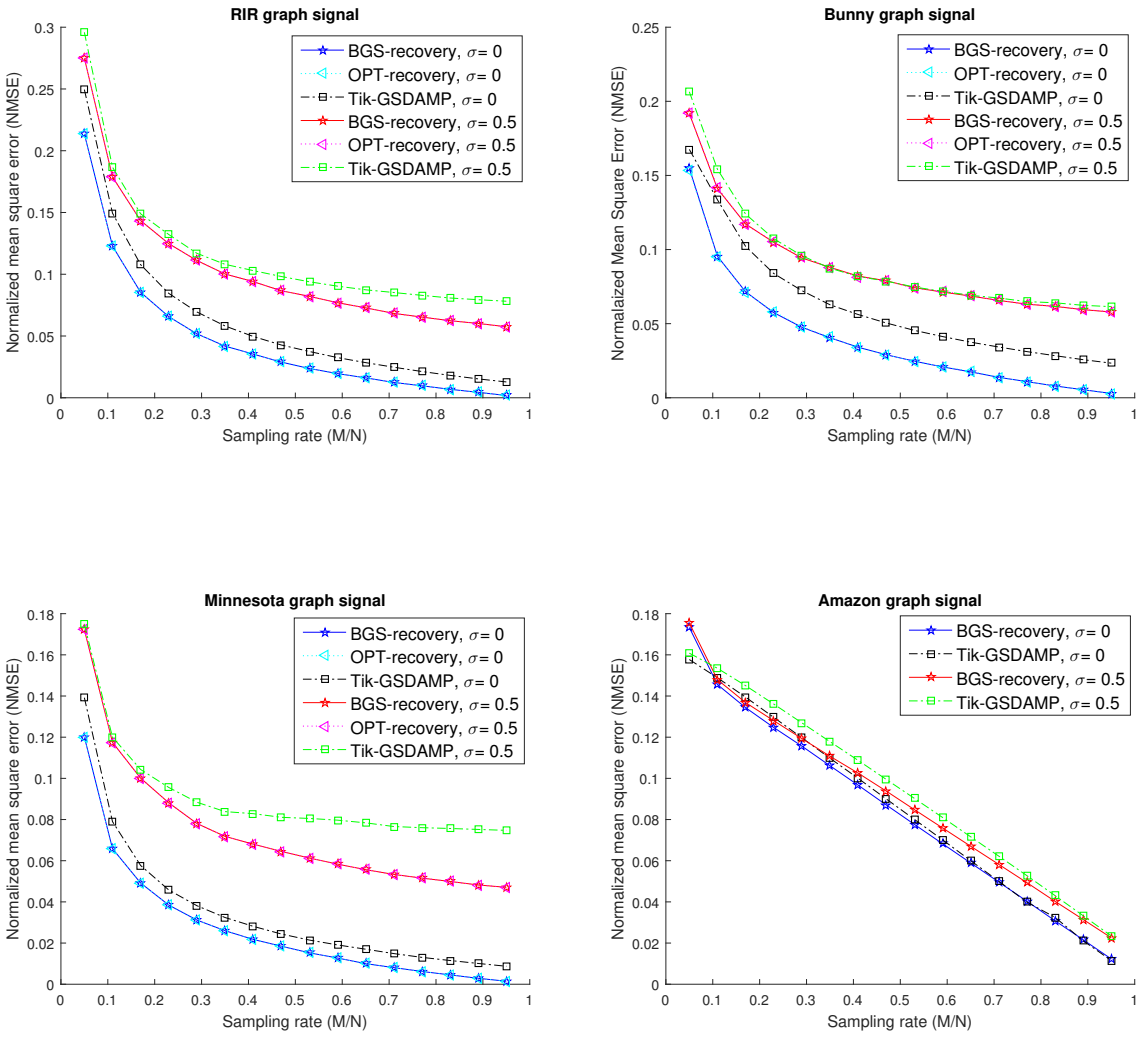


Figure 4.4. NMSE over varying sampling rates M/N.

Using the value of λ_{opt}, we analyze the effect of different noise standard deviations σ along with different sampling rates M/N on the NMSE of recovered signal. In order to validate

the performance of Algorithm 3, we compare the recovery performance of BGS-recovery algorithm with the optimal recovery method (OPT-recovery) along with the Tik-GSDAMP [12] graph signal recovery algorithm. In Figure 4.4, we plot the NMSE over sampling rate M/N for Tik-GSDAMP, BGS- and if feasible for OPT-recovery. In this figure, the x-axis is sampling rate $M/N \in [0.05, 0.95]$, and the y-axis is the NMSE. Intuitively, the value of NMSE decreases for a growing value of sampling rate. Besides, for all of the graph signals the NMSE of BGS-recovery algorithm is always smaller compared to Tik-GSDAMP, for both noiseless and noisy graph signals.

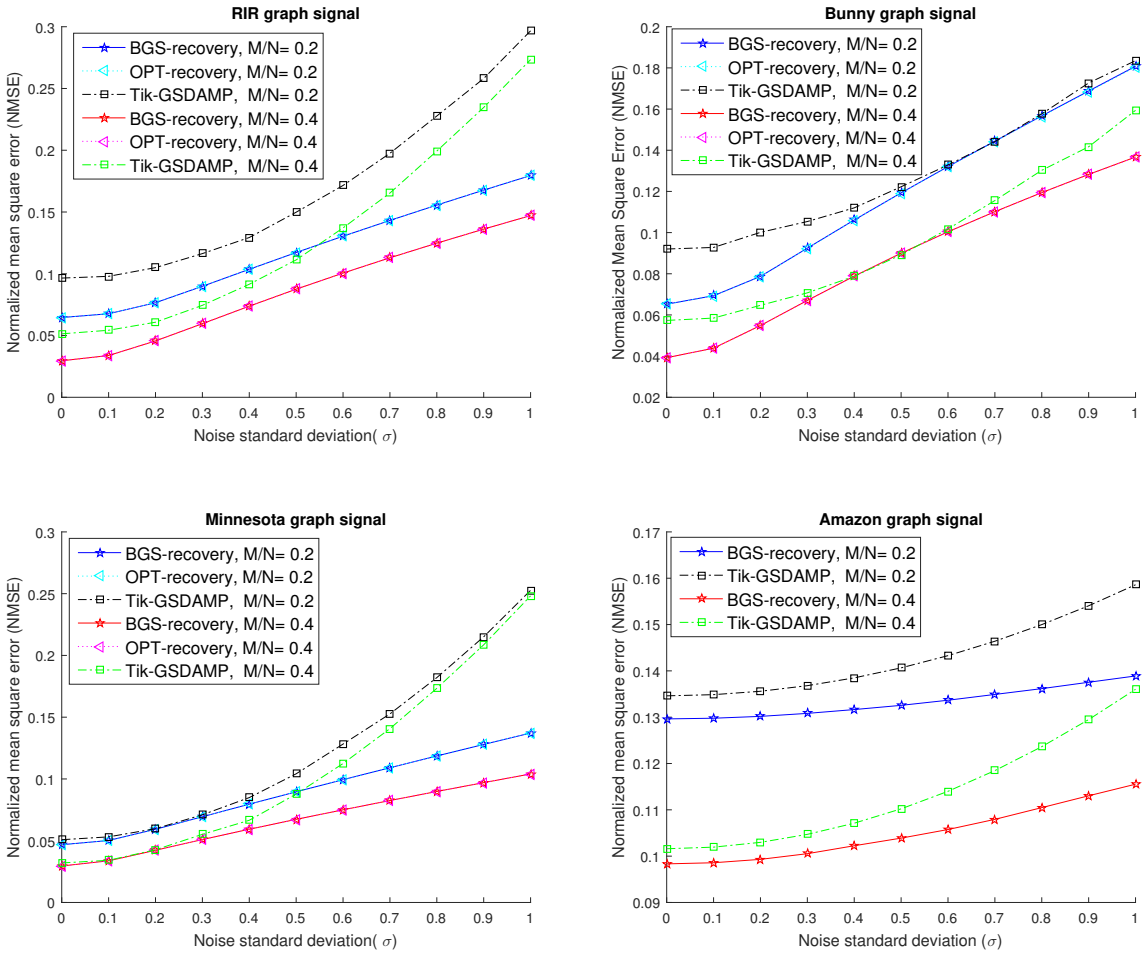


Figure 4.5. NMSE over varying noise standard deviations (σ).

Apart from the impact of sampling rate on NMSE, we also investigate the dependence of NMSE on the noise standard deviation σ for sampling rates of $M/N \in \{0.2, 0.4\}$. In Figure 4.5, the x-axis is the noise standard deviation $\sigma \in [0, 1]$, and the y-axis is the NMSE.

Table 4.1. Comparision of the recovery algorithms in terms of the simulation run time.

Recovery algorithm	Graph type	Simulation run time (sec.)
BGS- recovery	RIR	0.0867
	Minnesota	0.3104
	Bunny	0.2707
	Amazon p=4	111.6195
	Amazon p=40	19.1961
OPT- recovery	RIR	0.0826
	Minnesota	0.5039
	Bunny	2.0205
	Amazon	-
Tik-GSDAMP	RIR	0.0243
	Minnesota	0.0851
	Bunny	0.0924
	Amazon	29.9726

As evident from Figure 4.5, for all scrutinized graph signals the proposed method outperforms the Tik-GSDAMP recovery algorithm, particularly in higher noise standard deviation regime.

As illustrated in Figure 4.4 and Figure 4.5 in certain scenarios, the recovery performance of BGS recovery algorithm is exactly the same as the recovery performance of OPT-recovery method which outperforms the Tik-GSDAMP recovery algorithm. The OPT-recovery suffers from the high complexity and it is not feasible to use it for high-dimensional graph signal recoveries, while BGS-recovery has the same recovery performance as OPT-recovery and is applicable for high-dimensional graph signal recoveries due to the lower complexity.

Table 4.1 shows the simulation time of compared recovery algorithms on a system with the same settings. As it is obvious from the table the OPT-recovery algorithm shows the worst run-time performance compared to the two other algorithms particularly for the graphs with higher number of nodes and edges. We can also observe that the partitioning number has a non-negligible impact on the time complexity of BGS-recovery algorithm. While the run time of BGS-recovery algorithm with $p = 4$ on the Amazon graph is 111.6195 seconds, however, when we set $p = 40$ its run time reduces to 19.1961 seconds. Finally, we observe that in the case of large dimensional graph signals the BGS-recovery also outperforms Tik-GSDAMP recovery algorithm not only in terms of recovery performance but also of time complexity.

4.6 Conclusion

We formulated the problem of recovering a smooth graph signal from an incomplete noise-contaminated samples as a convex optimization problem. The optimization problem was

reduced to a system of linear equations involving the graph Laplacian. An efficient recovery method for smooth graph signals was then obtained by applying the BGS method, in addition to the optimum recovery solution. The effectiveness of the proposed recovery method was verified by numerical experiments on different graph signals including a real-world dataset containing product ratings of the Amazon Internet-based shop. According to the numerical results, in several scenarios, the proposed algorithm shows the exact results like the optimum solution and it outperforms a state-of-the-art recovery method, particularly for higher noise standard deviations.

Chapter 5

Adaptive graph signal sampling

5.1 Introduction

The previous chapters are concerned with noisy and under-determined graph signal recovery where the samples of graphs are chosen randomly with the uniform distribution. We shift gears in this chapter to consider the state-of-the-art and more accurate sampling techniques than a random sampling. Unlike classical signal processing, for even fundamental signal operation like sampling, the irregular structure of generic graphs causes many complications. The focus of this chapter is on the design of efficient sampling method which is necessary to reach highly accurate graph signal recovery.

In this chapter, we extend the theory of graph signal sampling by developing a fast and efficient algorithm for selecting the sampling set $\mathcal{S}$ of an arbitrary graph signal $\mathbf{x}$. The sampling theory deals with measuring a graph signal on a reduced set of nodes with conditions under which the signal has a stable reconstruction. Our goal is to select the minimum number of nodes in a way that it yields the reliable reconstruction of a signal.

The process of graph signal sampling is highly dependent on the structure of the graph signal. The smoothness factor, which is defined generally in terms of the signal's Fourier transform, is one of the main players in selecting the efficient sampling set of a graph signal. In a smooth graph signal, by sampling a node we are able to reconstruct its neighbours with high probability. Hence sampling two adjacent nodes may not be efficient. In a non-smooth graph signal, however, by sampling a node we cannot gain too much information about its neighbours. Another factor which affects the performance of the sampling algorithm is the degree of the nodes in the graph. When there is a high variation in the degree of different nodes and graph is relatively smooth, distributing the samples through the graph is more tricky. In one hand, sampling algorithm should be properly adjusted to avoid having a bulk of

samples in a highly connected part of the graph and in the other hand sampling the nodes with very small node degrees is not as useful as sampling the nodes with very large node degrees.

In order to reach the above mentioned goals considering the influence of smoothness and node degree variance we propose – as an alternative to random sampling of the nodes – a new AGSS algorithm to sample the nodes such that the recovery error is minimized. In this mechanism, we apply a tuning factor for the neighbouring nodes of sampled node r. The tuning factor should be adaptively adjusted based on the graph signal structure such as smoothness and node degree variance. We confirm the performance of proposed sampling algorithm by conducting illustrative numerical experiments on various random and real-world graphs.

The rest of the chapter is organized as follows. Section 5.2 introduces the preliminaries, and reviews the state-of-the-art sampling technique that we will compare our sampling algorithm with. Section 5.3 presents and discusses the proposed sampling method. Section 5.4 provides simulations on both real-world data-set and random graphs.

5.2 Preliminaries

To make it more accessible for the reader to follow, we again briefly review the concept of graph and the foundations of the proposed work. We consider the loopless weighted graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{W})$, where $\mathbf{W} \in \mathbb{R}^{N\times N}$ is a weighted adjacency matrix demonstrating a discrete version of the graph, $\mathcal{V} = \{1, \ldots, N\}$ is the node set and $\mathcal{E}$ is the edge set consisting of unordered node pairs (r, s) for which $W_{rs} \neq 0$. The edge weight W_{rs} between the two neighbouring nodes r and s is a quantitative illustration of the underlying relation between the nodes, i.e., a dependency or similarity. The support of the matrix $\mathbf{W}$ reflects the edge structure of the graph $\mathcal{G}$. A graph $\mathcal{G}$ is loopless if $W_{rr} = 0, \forall r \in \mathcal{V}$; and it is connected if there exists a path between any pair of the nodes. The degree matrix $\mathbf{D}$ is a diagonal matrix of size $N \times N$, whose r^{th} diagonal element D_{rr} is equal to the sum of the weights of all the edges connected to the node r, i.e., $D_{rr} = \sum_{s\in\mathcal{V}} W_{rs}$. The weights are normalized if $D_{rr} = 1, \forall r$. In addition to the above matrices, another essential matrix associated with a graph $\mathcal{G}$ is the graph Laplacian matrix $\mathbf{L}$, i.e., $\mathbf{L} = \mathbf{D} - \mathbf{W}$.

The δ_{rs} is the distance metric which shows the number of edges in a shortest path (also called a graph geodesic) connecting node r and s. The k-step neighbourhood $\mathcal{N}_r^k = \{s \in \mathcal{V} : \delta_{rs} = k\}$ of the node r is the set of all nodes which are at distance k from the node r. A connected graph $\mathcal{T}$ without cycles, which has all the nodes and a subset of edges of $\mathcal{G}$ is a spanning tree (ST) of a connected graph $\mathcal{G}$. If the total edge weight of $\mathcal{T}$ is maximum over

all possible STs of $\mathcal{G}$ then it is called a maximum spanning tree (MST) of the underlaying graph $\mathcal{G}$. In a case that the underlying graph is unweighted all of the possible STs are MST.

For a given graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{W})$, a graph signal $\mathbf{x} : \mathcal{V} \to \mathbb{R}^N$ is a mapping from the node set into the reals[1], i.e., each node $r \in \mathcal{V}$ is assigned a graph signal value $x_r \in \mathbb{R}$. We will stack the graph signal values into a vector $\mathbf{x} \in \mathbb{R}^N$, whose r^{th} entry is the graph signal value x_r at node $r \in \mathcal{V}$. It should be noted that the graph signal recovery problem arises, when we assume that only a few components x_r of the signal $\mathbf{x}$ are sampled. We consider the problem of recovering a smooth graph signal $\mathbf{x} = \{x_r, r = 1, 2, ..., N\}$ (true graph signal) from its noisy samples

$$y_l = x_{s_l} + n_l, \quad l \in \{1, 2, ..., M\}, \tag{5.1}$$

where

$$s_l \in \{1, 2, ..., N\} \quad \text{and} \quad s_l \neq s_{l'} \quad \text{for} \quad l \neq l', \tag{5.2}$$

and the set

$$\mathcal{S} = \{s_1, s_2, ..., s_M\}, \tag{5.3}$$

is, for $M << N$, a subset of the node set $\mathcal{V}$. We assume the number M of samples to be much smaller than the graph signal size N. By placing the observations y_l into the measurement vector $\mathbf{y} = \{y_l, l = 1, 2, ..., M\} \in \mathbb{R}^M$, we obtain a linear measurement model

$$\mathbf{y} = \mathbf{A}\mathbf{x} + \mathbf{n}, \tag{5.4}$$

where $\mathbf{n} = \{n_l,\ l = 1, 2, ..., M\}$, is the noise vector modeled as component-wise independent AWGN with zero-mean and variance σ^2 i.e., $n_l \sim \mathcal{N}(0, \sigma^2)$. The noise vector is to encompass the effects of measurement and modeling errors. The measurement matrix $\mathbf{A} \in \{1, 0\}^{M \times N}$ represents the sampling matrix.

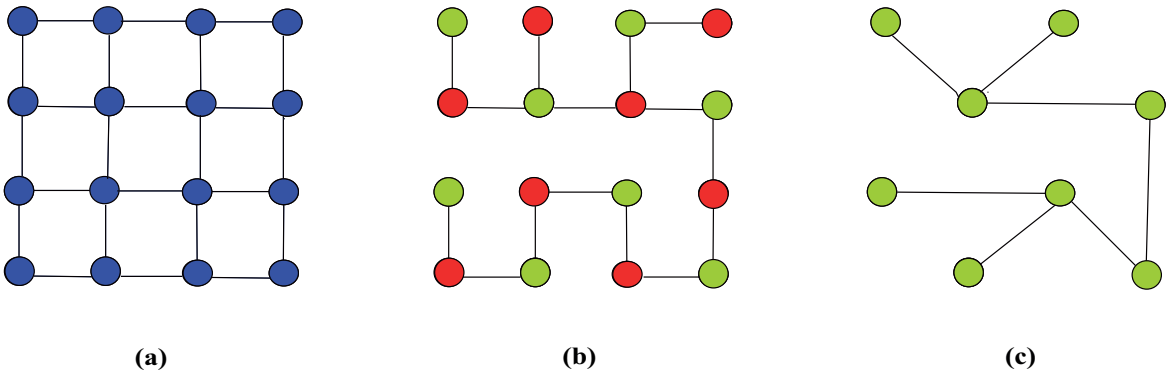
5.2.1 The MST-based sampling

In [115] the authors proposed a new sampling technique which approximates the underlying graph $\mathcal{G}$ using a spanning tree. Since the spanning tree $\mathcal{T}$ should be as similar as possible to the original graph $\mathcal{G}$, the graph multi-resolution is provided by approximating the original graph using a MST. The graph multi-resolution is defined by the structure of the tree $\mathcal{T}$, which is a special bipartite graph $\mathcal{G}$ [116].

In the first step, the sampling algorithm finds the MST using Prime or Kruskal algorithm [117]. Then through the MST of the graph, the sampling algorithm does the graph partition

[1]The extension to complex values is feasible but is not considered in this work.

and reduction. In this sampling method, the sampling set $\mathcal{S}$ is constructed using all the nodes with even distance from an arbitrary chosen root node r in the induced spanning tree $\mathcal{T}$. Finally, each node in $\mathcal{S}$ is connected to its grandparent nodes (which are also in $\mathcal{S}$) to make a connected tree. In order to clear out the procedure of the algorithm, an example of MST-based sampling is provided in Figure 5.1. Figure 5.1-(a) shows the original grid graph. The two independent subsets of MST-based sampling are labeled with red and green circles in Figure 5.1-(b) and the sampled graph Figure 5.1-(c) is provided by applying the connecting rules on the sampled nodes.



(a) **(b)** **(c)**

Figure 5.1. MST-based sampling on grid graph, (a) is the original grid graph, (b) the MST of the grid graph, and (c) is the sampled graph.

5.3 Adaptive graph signal sampling (AGSS)

The sampling process of a graph signal depends highly on the structure of the underlying graph signal. Generally, the nodes with larger value in the diagonal degree matrix $\mathbf{D}$ have more influence on the recovery of the graph signal compared to the nodes with smaller value in the diagonal of matrix $\mathbf{D}$, i.e., if $D_{rr} > D_{ss}$ then the r^{th} node participates more than s^{th} node in the recovery of the graph signal. A trivial way for the sampling of the graph signal is to first select the nodes with larger value in the degree matrix $\mathbf{D}$, i.e., to sample a node that the sum of the weights of all the edges connected to that node is higher. Applying this approach on a graph signal will lead to the aggregation of all samples in the certain areas with the high node degrees. Hence, the nodes located in the dense part of the graph will have a good recovery while the rest of the nodes will suffer from an unpleasing recovery performance. This drawback is more sever in smooth graph signals where the signal value does not vary too much among the neighbouring nodes. Hence, sampling the neighbouring nodes with similar signal values is not beneficial for the recovery algorithm. The sampled nodes should

be distributed over the underlying graph to attain a higher recovery performance. To achieve this goal we introduce a new parameter called tuning factor **q**. When the graph is smooth and there is a drastic difference between the values of the components of the degree matrix **D**, by increasing the tuning factor at first we observe an increase in the recovery performance, however, after a certain point further increment of tuning factor leads to a lower recovery performance. In this situation, if we continue to increment the value of the tuning factor this may lead to sample the nodes with very low node degrees which are not also much beneficial in recovery performance of smooth signal. Therefore, in smooth graph signal with high node degree variance the optimal value of tuning factor should be properly adjusted. Conversely, in non-smooth graph signal sampling a node does not give us too much information about its neighbours. Hence, in this case applying a lower tuning factor is a wiser option.

Considering the above mentioned justification and in order to have an efficient graph signal sampling, we present a greedy heuristic sampling algorithm called AGSS. We provide an adaptive sampling method for constructing the sampling set $\mathcal{S}$ and the corresponding measurement matrix **A**. The proposed sampling scheme is stated in Algorithm 4.

Algorithm 4: Adaptive graph signal sampling (AGSS)

Input: node set $\mathcal{V} = \{1, 2, ..., N\}$, K maximum walking step (MWS), $\mathbf{W}$ weighted adjacency matrix and M number of samples
Initialization: sampling set $\mathcal{S} = \emptyset$, the tuning vector $\mathbf{q} = [q_1, q_2, \ldots, q_K]$ and degree vector $\mathbf{d}$ where $d_r = D_{rr} = \sum_{s \in \mathcal{V}} W_{rs}$
for $m = 1$ **to** M
    find the highest-degree node $d_r \in \mathbf{d}$
    set $d_r = 0$
    add r to sampling set $\mathcal{S}$
    create $N \times 1$ zero-vector $\mathbf{x} = (0, 0, ..., 0)^T$
    set vector component $x_r = 1$
    define m^{th} row of sampling matrix: $A_m = \mathbf{x}^T$
    for $k = 1$ to K **do**
        for all node $s \in \mathcal{N}_r^k$ **do**
            set $d_s = \frac{d_s}{q_k}$ where $q_k > 0$
        end for all
    end for
end for
Output: sampling set $\mathcal{S}$ and measurement matrix $\mathbf{A}_{M \times N}$.

Our goal is to sample as few components x_r of the graph signal $\mathbf{x}$ as possible and still recover the graph signal with the highest achievable quality, by exploiting the weighted adjacency matrix $\mathbf{W}$ which describes the similarity of the values of the graph signal components. This

information is used, e.g., by Tikhonov regularization in the recovery process for the original graph signal from its under-sampled representation.

The AGSS algorithm starts by finding the node r in the node set $\mathcal{V}$ with maximum value in vector $\mathbf{d}$ which is constructed from the diagonal elements of the degree matrix $\mathbf{D}$. Then it adds node r to the sampling set $\mathcal{S}$ and sets its degree to $d_r = 0$. The tuning vector $\mathbf{q}$ is defined to prevent from aggregation of samples in the highly connected subgraphs of the original graph $\mathcal{G}$. In other words, after sampling the node r the algorithm changes the probability of sampling its neighbours in $\mathcal{N}_r^k$ by factor q_k. The tuning factor decreases/ increases the chance of sampling the neighbouring nodes in the next rounds of algorithm, i.e., $\forall s \in \mathcal{N}_r^k, d_s = d_s/q_k$. As an example in the smooth graph signal, the 1-step neighbouring nodes of r have higher similarity with it than the 2-step neighbours, so we penalize them with higher factor. Hence, the tuning vector $\mathbf{q}$ is chosen such that $q_i \leq q_j$ when $i > j$ to penalize the neighbouring node s with smaller distance from the selected node r (δ_{rs}) by higher factor. The algorithm continues by adding the node with maximum degree in updated vector $\mathbf{d}$ to the sample set $\mathcal{S}$. Considering $K > 2$ leads to alteration in degree of the nodes that are far away from the sampled node. This setting is just efficient for highly smooth graph signals. Hence, in this chapter we use MWS $K \leq 2$.

The proposed AGSS algorithm provides any given arbitrary sampling rate by setting the number of the samples M. However, the MST-based sampling is able to just provide some certain resolutions of the graph and is not able to sample the graph for the arbitrary sampling rate (M/N).

5.3.1 Illustrative example

In order to facilitate the understanding of the algorithm, we follow its procedure on a small unweighted toy example. Figure 5.2 (a) shows a graph with 15 nodes. Following the approach of Algorithm 4, degree vector $\mathbf{d}$ is equal to $\mathbf{d} = [5\ 4\ 4\ 3\ 6\ 4\ 3\ 3\ 2\ 2\ 2\ 3\ 2\ 2\ 1]$. Considering $K = 1$, $q_1 = 2$ and $M = 6$ the algorithm starts by sampling node 5 and updates the degree vector to $\mathbf{d} = [2.5\ 2\ 4\ 1.5\ 0\ 2\ 1.5\ 3\ 2\ 2\ 2\ 1.5\ 2\ 2\ 1]$. The algorithm continues by sampling node 3 with maximum value in updated degree vector and updates once again the degree vector to $\mathbf{d} = [1.25\ 2\ 0\ 1.5\ 0\ 2\ 1.5\ 1.5\ 1\ 1\ 2\ 1.5\ 2\ 2\ 1]$. By following this approach to sample

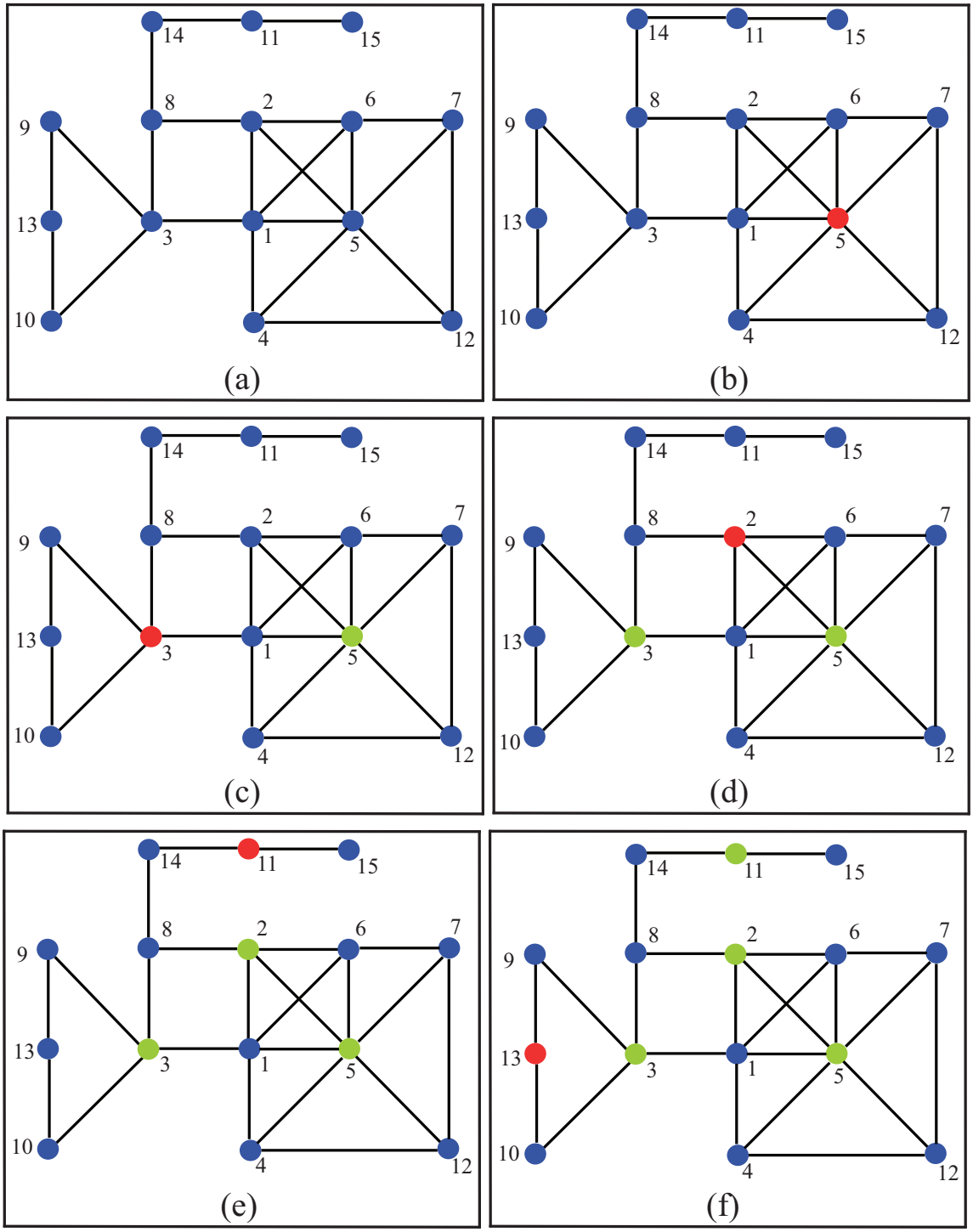


Figure 5.2. Toy example for AGSS algorithm

$M = 6$ nodes the sampling set is equal to $\mathcal{S} = \{5,\ 3,\ 2,\ 11,\ 13,\ 4\}$ and sampling matrix is

$$\mathbf{A} = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}, \tag{5.5}$$

The successive steps of sampling algorithm on investigate toy example is shown in Figure 5.2 (b) to Figure 5.2 (f), where the nodes sampled in the previous steps are in green and the sampled node in the current step is in red.

5.4 Numerical results

In this section, we present the results of the numerical experiments comparing the performance of the proposed sampling method with MST-based sampling of the graph. Using the GSPBox software [96], we generate three distinct loopless graphs, i.e., the RIR graph with size $N = 1000$, $\mathcal{E} = 3362$ edges and the node degree variance of $\mathrm{Var}(\mathbf{d}) = 0.8$, the Bunny graph with size $N = 2503$, $\mathcal{E} = 13726$ edges and the node degree variance of $\mathrm{Var}(\mathbf{d}) = 1.1$, as well as the Minnesota road graph with size $N = 2642$, $\mathcal{E} = 3304$ edges and the node degree variance of $\mathrm{Var}(\mathbf{d}) = 0.5$. All the graph signals take their values from the set $x_r \in \{-1, 1\}$.

We applied the proposed algorithm and the MST-based sampling technique to the mentioned graph signals along with the Amazon product rating graph (see Section 3.4.1 for more details) and compared their performance by employing the state-of-the-art recovery algorithm BGS [13] on the outcome of each of the scrutinized sampling methods, i.e., on every constructed sampling set $\mathcal{S}$ separately. For sufficient statistical significance of the results we ran the BGS recovery method for 500 times and each time with different noise realizations. The final result is averaged over the outcomes of the individual runs of the recovery scheme. It should be noted that for all of the simulations, we set the value of MWS $K = 1$.

In the first part of the numerical evaluations, we investigate the effect of the graph smoothness factor $S^{'}(x)$ defined in (3.5) on adjusting the tuning factor q_k. To do that, we consider two different arbitrary graph smoothness factors $S^{'}(x)$ for RIR, Bunny and Minnesota graphs.

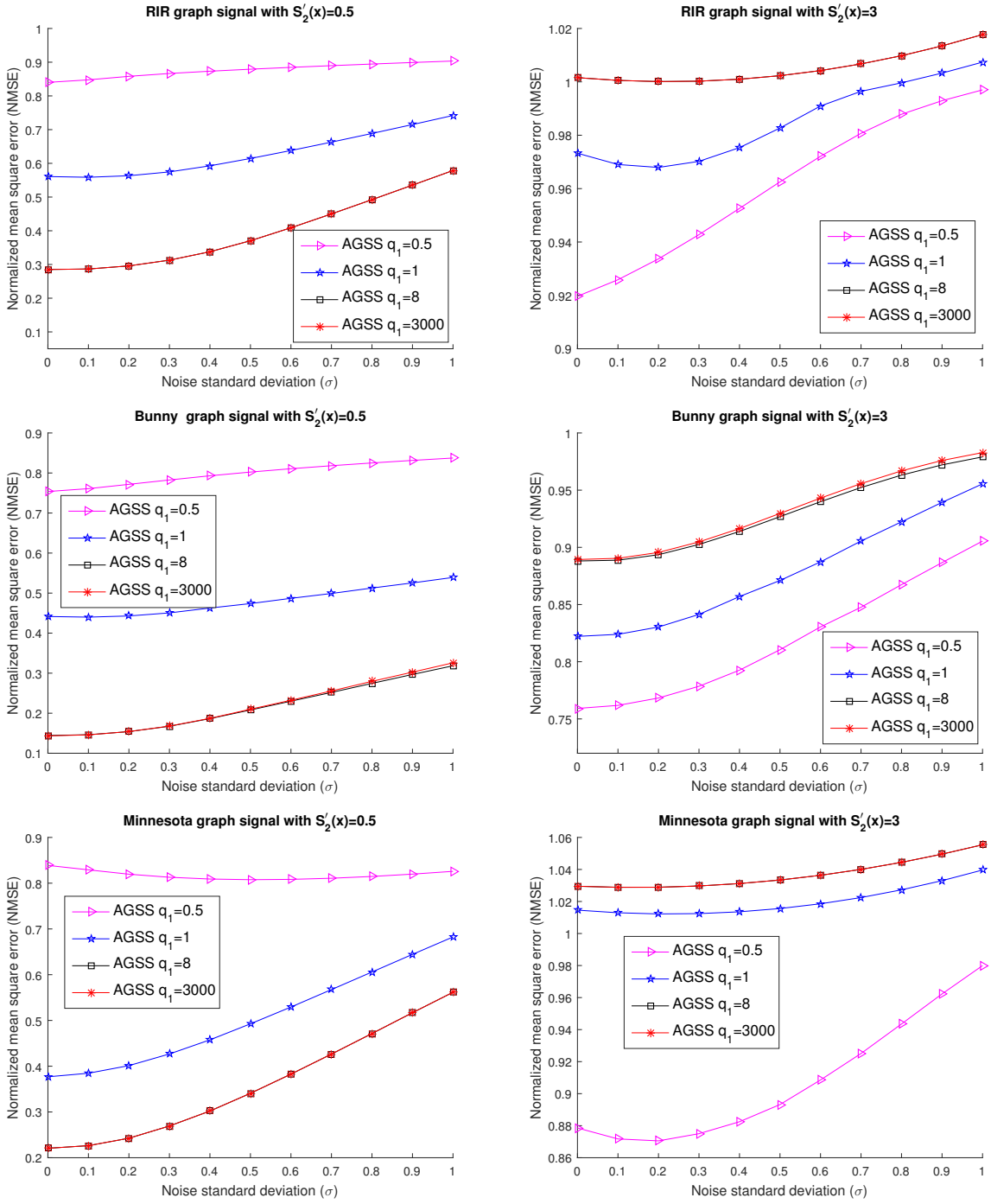


Figure 5.3. The relation between the smoothness and tuning factor.

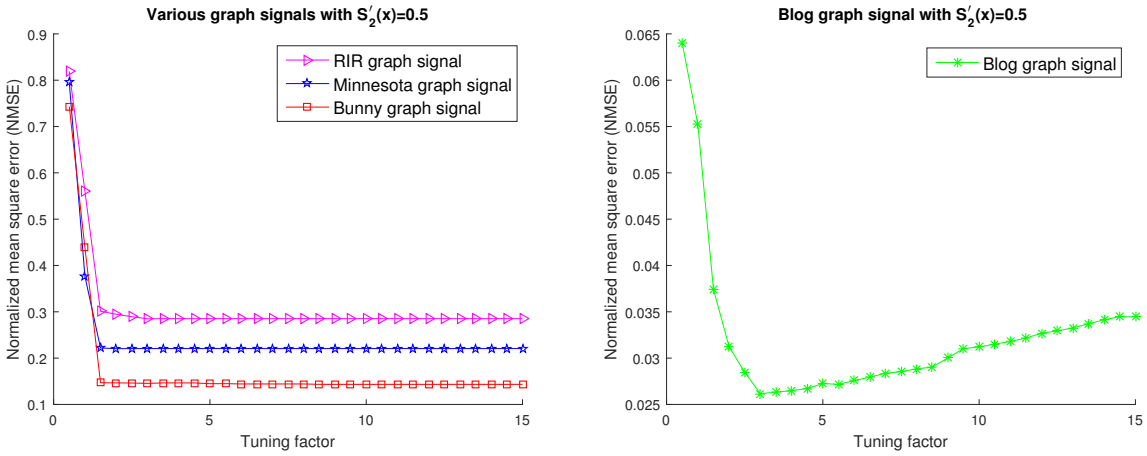


Figure 5.4. The relation between the node degree variance and tuning factor.

It should be noted that, the Amazon product rating data-set is a real-world data and it has a fix graph smoothness factor $S^{'}(x) = 0.54$.

We consider a relatively smooth signal with $S^{'}(x) = 0.5$ and a non-smooth signal with $S^{'}(x) = 3$ over RIR, Bunny and Minnesota graphs. The simulation results presented in Figure 5.3 illustrate the relation between the smoothness and the tuning factor, in terms of the NMSE of recovered signals with sampling rate $M/N = 0.25$. As we discussed in Section 5.3, it can be seen from Figure 5.3 (left) that applying a larger tuning factor q_k over the smooth graph signal leads to a higher recovery performance. In this case, the signal values of the neighbouring nodes are almost the same and the recovery algorithm does not benefit from sampling of the adjacent nodes. Hence, after sampling a node we should penalize its neighbours with a high tuning factor to prevent the algorithm to resample from one particular highly connected area of the graph, i.e., the high tuning factor aids the algorithm to avoid having a bulk of similar (uninformative) samples from the same area. In this case, sampling of the signal with tuning factor of $q_1 = 0.5$ leads to lowest recovery performance. On the contrary, in the non-smooth graph signal, the value of graph signals vary drastically among the adjacent nodes and it is more efficient and reasonable to sample all the nodes with higher degree. Hence, in the non-smooth graph signal running the AGSS algorithm with $q_1 = 0.5$ leads to a highest recovery performance, (see, Figure 5.3 (right)). This results confirm our arguments regarding to the influence of smoothness on the sampling of the graph signal.

In order to see the effect of the node degree variance on the optimal setting of tuning factor, we evaluate the effect of varying tuning factor over a real-world political blogs data-set [118] which has very high node degree variance. This data-set consists information about left-leaning and right-leaning political blogs. Blogs are represented by the nodes of the graph and nodes are connected by an edge when one blog refers to another one. The graph of the

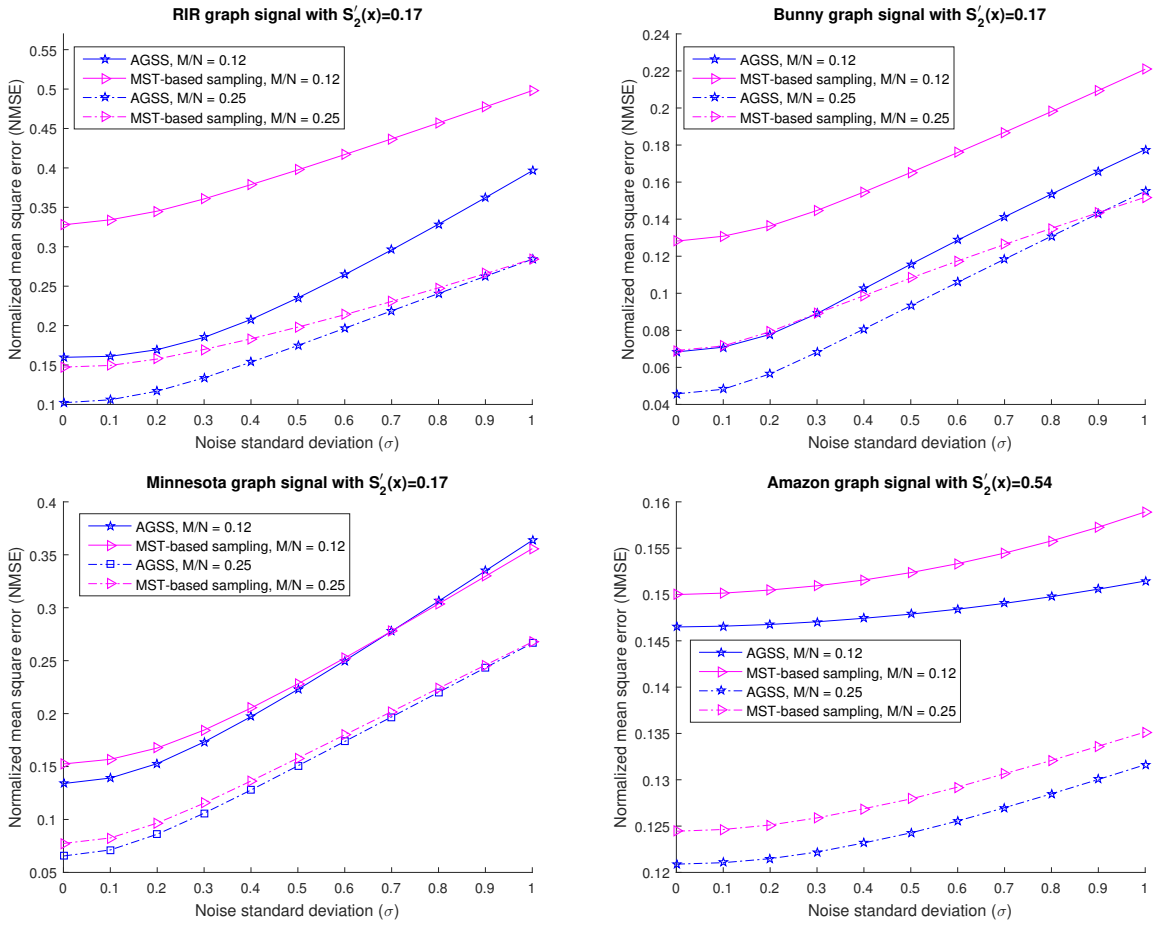


Figure 5.5. NMSE over the noise standard deviation σ, where the sampling rate $M/N \in \{0.12, 0.25\}$.

raw data contained 266 isolated nodes. We selected the largest connected subgraph $\mathcal{G}$ for our numerical experiments. In this graph there are $N = 1222$ nodes, $|E| = 16660$ edges and the node degree variance is $\text{Var}(\mathbf{d}) = 14682$. As it is shown in Figure 5.4, in RIR, Minnesota, and Bunny graphs with low node degree variance the incrementation of the tuning factor after certain point does not influence the NMSE of recovered signal. However in Blog graph signal with high node degree variance, increasing the tuning factor until it reaches to the optimal point decreases NMSE, where after that point the further increment of tuning factor increases the NMSE of recovered signal. This results confirm our arguments regarding to the influence of node degree variance on the sampling of graph signal.

In order to validate the effect of the proposed algorithm on the performance of the recovery algorithms, first by exploiting the compared sampling techniques we sample the nodes of each of the mentioned graphs to have an under-determined graph signal with the sampling rate $M/N \in \{0.12, 0.25\}$. Then, we use the BGS recovery algorithm to analyze the NMSE

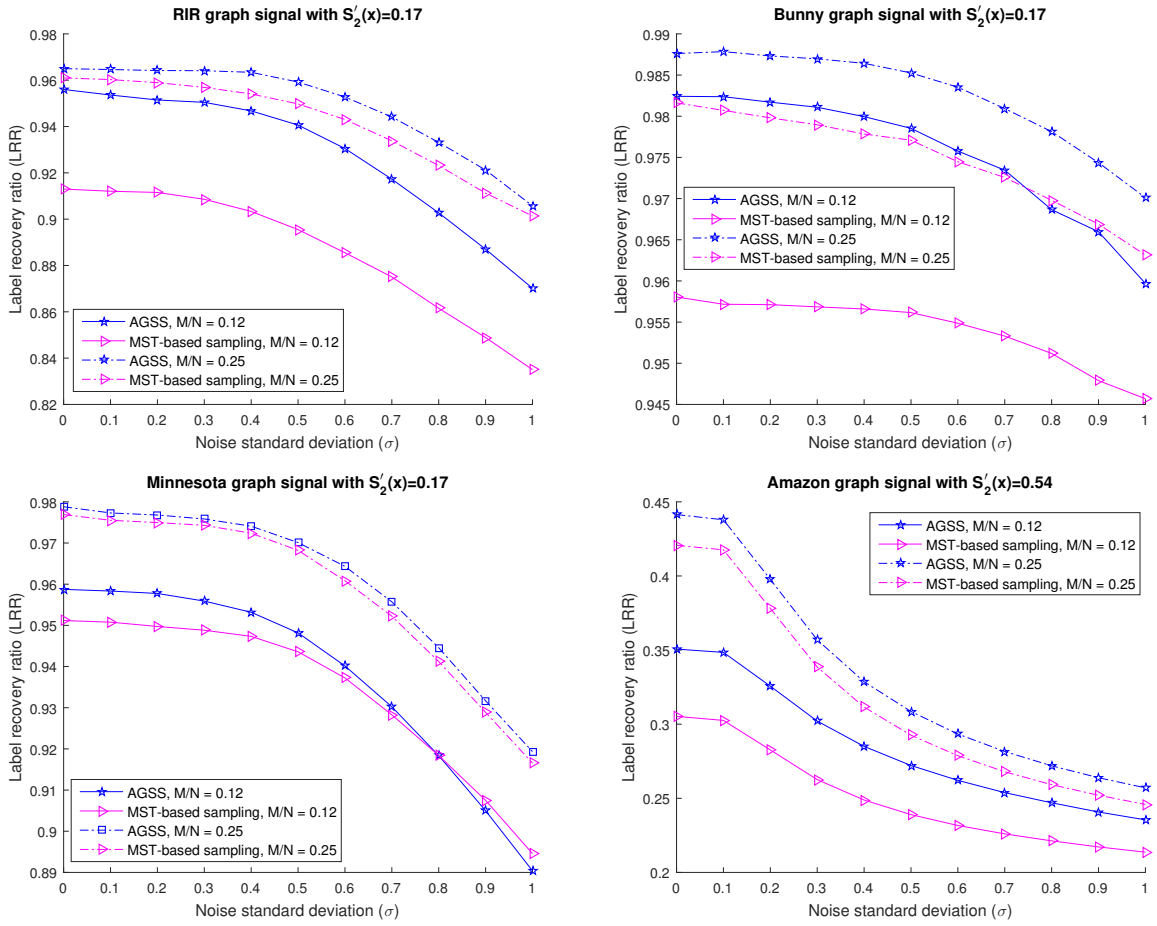


Figure 5.6. LRR over the noise standard deviation σ, where the sampling rate $M/N \in \{0.12, 0.25\}$.

and LRR over varying noise standard deviation (σ). The LRR a defined as the fraction of nodes $r \in \mathcal{V}$, for which the recovery error is $|x_r - \hat{x}_r| = 0$, i.e.,

$$a = \mid \{r \in \mathcal{V}, x_r = \hat{x}_r\} \mid /N, \tag{5.6}$$

where, x_r is the original signal value at node r, and $\hat{x}_r$ is the recovered signal value. To obtain the value of a, first we round the recovered signal value to the nearest signal value in the signal value set. Here, LRR exhibits the ratio of signal values which are perfectly recovered by the BGS recovery algorithm exploiting AGSS and MST-based sampling techniques.

For the rest of the simulations, we set the value $q_1 = 4$ to compare the performance of AGSS algorithm with the MST-based algorithm. Figures 5.5 and 5.6 compare the performance of AGSS sampling with MST-based sampling over varying noise standard deviation (σ) in the RIR, Bunny and Minnesota smooth graph signals with smoothness factor $S'(x) = 0.17$

and in rather smooth Amazon product rating graph with $S^{'}(x) = 0.54$. The simulation results show the performance of different sampling schemes in terms of NMSE (Figure 5.5) and LRR (Figure 5.6), where the sampling rate $M/N \in \{0.12, 0.25\}$. Simulation results confirm the superiority of AGSS sampling over the compared MST-based sampling algorithm particularly in low sampling rate regime, i.e., $M/N = 0.12$.

5.5 Conclusion

In this chapter, we have proposed a novel adaptive sampling algorithm for signals living on the arbitrary weighted graphs. The proposed scheme selects a sampling set based on the degree of the nodes. After sampling a node, depend on the graph structure and signal model, the probability of sampling its neighbours change. The value of tuning factor is adaptively adjustable based on the smoothness and structure of the graph, e.g., in the smooth graph signals high value of tuning factor prevents from aggregation of samples in the highly connected subgraphs. Simulation results show the superiority of proposed scheme over the existing state-of-the-art sampling method.

Chapter 6

Conclusion

A recent approach to deal with large-scale datasets occurring in big data applications such as genetics, image processing and social network analysis is the theory of graph signal processing. GSP can be viewed as a generalization of classical signal processing; the latter is obtained from GSP for the special case of a chain graph (representing the sequence of time instants). The usage of graph models within GSP entails efficient regularization algorithms that are well suited to deal with large volumes of high-speed data. Moreover, graphs allow to organize heterogeneous data by exploiting application specific notions of similarity, thereby addressing the variety of big data.

6.1 Summary of contributions

The first part of this dissertation is devoted to a new AMP-based method for recovering smooth graph signals based on a small number of noisy signal samples. The proposed schema in Chapter 3 can be regarded as an instance of DAMP framework for the special case of graph signals with small total variation. The presented GSDAMP recovery algorithm combines graph signal denoisers with the AMP framework. We proposed a novel method to adjust the regularization parameter based on the estimated noise variance in each iteration of GSDAMP framework. The effectiveness of the proposed recovery method is verified by numerical experiments via several sample graphs including a real-world data-set containing product ratings of a large Internet-based retail shop.

In Chapter 4 in this dissertation, we formulate the problem of recovering a smooth graph signal from under-determined noisy samples as a convex optimization problem which, in turn, amounts to solving a system of linear equations involving the graph Laplacian. An efficient recovery method for smooth graph signals is then obtained by applying a BGS method to this Laplacian system. Through extensive simulation experiment we show that, the proposed

iterative BGS recovery algorithm provides a tight approximation of optimal solution while it is highly efficient in terms of time complexity.

The final part of the dissertation complements the whole process by proposing a novel adaptive sampling algorithm for signals living on the arbitrary weighted graphs. The proposed scheme selects a sampling set based on the sum of the nodes edge weights. After sampling each node, the algorithm changes the degree value of its neighbours based on a predefined tuning factor. The value of tuning factor is adaptively adjustable based on the smoothness and structure of the graph. Simulation results show the superiority of proposed scheme over the existing state-of-the-art sampling method.

6.2 Open issues and outlook

Notwithstanding the precise modeling and analyze of the graph signal sampling and recovery in this dissertation, there are still some promising directions for future research and exploration which are listed below:

- This dissertation is based on the static graphs. Hence, our schemes are designed for such graphs. What about he non-static graphs such as time-variant or multilayer graphs? The possibility of extending the existing tools for the non-static graph signals could be considered in future.

- Throughout this dissertation we have assumed implicitly that the topology of the underlying graph is known. In many applications, however, the graph topology is not known a priori. Learning the graph topology from the signals themselves could be another research direction over this dissertation.

- As we mentioned during this dissertation, most works in graph signal processing consider smoothness in a way that it reflects small variations (low frequency) in the graph signal values. What about the mirror opposite of this definition? This definition is called heterophily and it is defined as the degree to which pairs of the nodes with connecting edge are different in certain attributes. Heterophily has also become an area of social network analysis. Hence, the question here is that does heterophily reflect the high variations (high-frequency) signal values? If so how can we use this prior information to our benefit?

- An important extension of this work in Chapter 6 is to formalize in a rigorous mathematical manner the relation between the chosen tuning vector and the underlying graph structure. This will be addressed in forthcoming works.

Appendix A

A.1 Estimators

MSE estimator:
The mean squared error (MSE) between a vector $\mathbf{a} \in \mathbb{C}^N$ and a vector $\mathbf{b} \in \mathbb{C}^N$ is

$$MSE(\mathbf{a}, \mathbf{b}) = \frac{1}{N}\|\mathbf{a} - \mathbf{b}\|_2^2 = \frac{1}{N}\sum_{n=1}^{N}|a_n - b_n|^2.$$

The MSE between two matrices $\mathbf{A} \in \mathbb{C}^{M\times N}$ and $\mathbf{B} \in \mathbb{C}^{M\times N}$ is defined as

$$MSE(\mathbf{A}, \mathbf{B}) = \frac{1}{MN}\|\mathbf{A} - \mathbf{B}\|_F^2.$$

NMSE estimator:
The NMSE between two vectors is defined as

$$NMSE(\mathbf{a}, \mathbf{b}) = \frac{\|\mathbf{a} - \mathbf{b}\|_2^2}{\|\mathbf{a}\|_2^2},$$

the NMSE between two matrices as

$$NMSE(\mathbf{A}, \mathbf{B}) = \frac{\|\mathbf{A} - \mathbf{B}\|_F^2}{\|\mathbf{A}\|_F^2}.$$

A.2 Norms

l_p-norm:
The l_p-norm of a vector $\mathbf{x} \in \mathbb{C}^N$ is defined as

$$\|\mathbf{x}\|_p = \left(\sum_{n=1}^{N} |x_n|^p \right)^{\frac{1}{p}}.$$

l_2-norm:
The l_2-norm of a vector $\mathbf{x} \in \mathbb{C}^N$ is defined as

$$\|\mathbf{x}\|_2 = \sqrt{\sum_{n=1}^{N} |x_n|^2}.$$

Frobenius norm:
The Frobenius norm of a matrix $\mathbf{A} \in \mathbb{C}^{M \times N}$ is defined as

$$\|\mathbf{A}\|_F = \sqrt{trace(A^H A)} = \sqrt{\sum_{m=1}^{M} \sum_{n=1}^{N} |A_{mn}|^2}.$$

A.3 Distributions

Gaussian distribution:
With mean μ and variance σ^2 Gaussian distribution evaluated at x is defined as

$$\mathcal{N}\left(x;\mu,\sigma^2\right) = \frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(-\frac{1}{2\sigma^2}(x-\mu)^2\right).$$

Uniform distribution:
For a finite set $\mathcal{S} = \{s_1, ..., s_N\}$ uniform distribution $\mathcal{U}[s_1, ..., s_N]$ is defined by the (generalized) probability density function

$$f_s(s) = \sum_{n=1}^{N}\frac{1}{N}\delta\left(s - s_n\right),$$

or

$$\mathbb{P}\{s = s_n\} = \frac{1}{N}\forall n.$$

Laplace distribution:
With mean μ and variance $2k^2$ Laplace distribution evaluated at x is defined as

$$\mathcal{L}\left(x;\mu,k\right) = \frac{1}{2k}\exp\left(-\frac{|x-\mu|}{k}\right).$$

Appendix B

B.1 Recovery performance with respect to LRR

As we mentioned in Section 3.4, incrementing the noise level harms the recovery performance. As shown in Figure B.1, for noise standard deviation $\sigma = 0.5$, the recovery performance

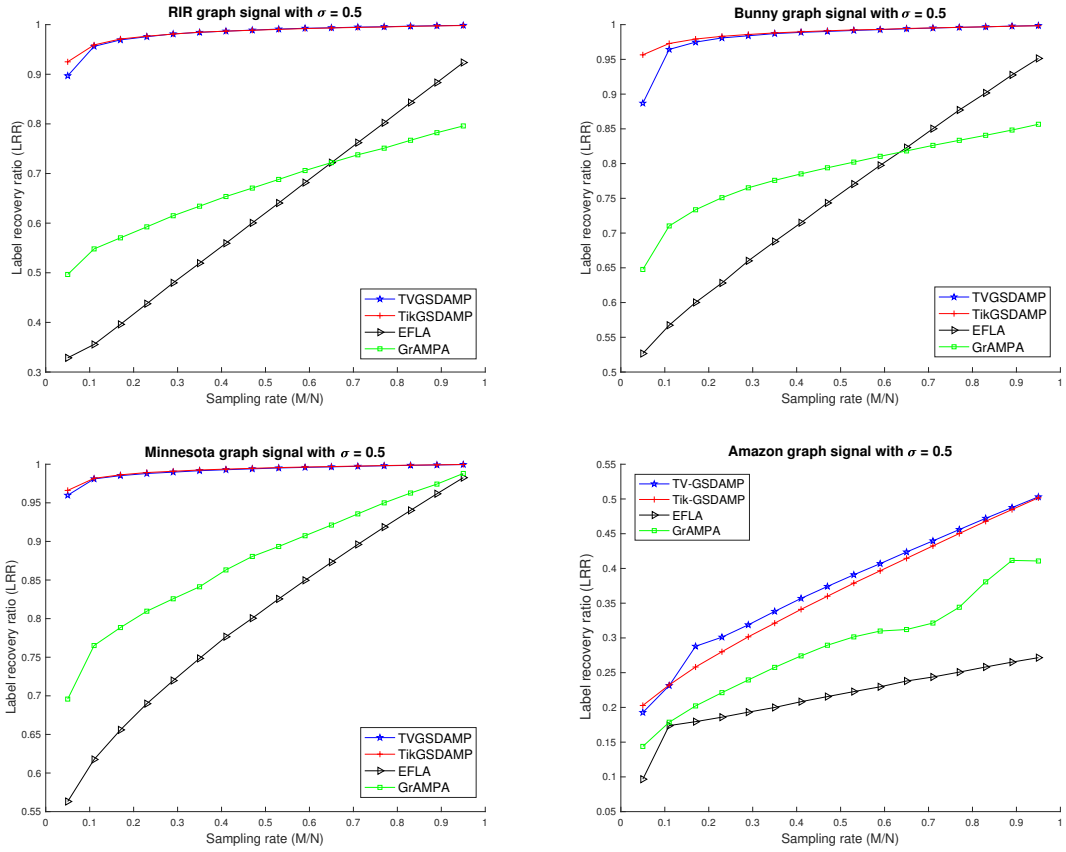


Figure B.1. LRR vs sampling rate M/N, where the noise standard deviation σ is set to 0.5.

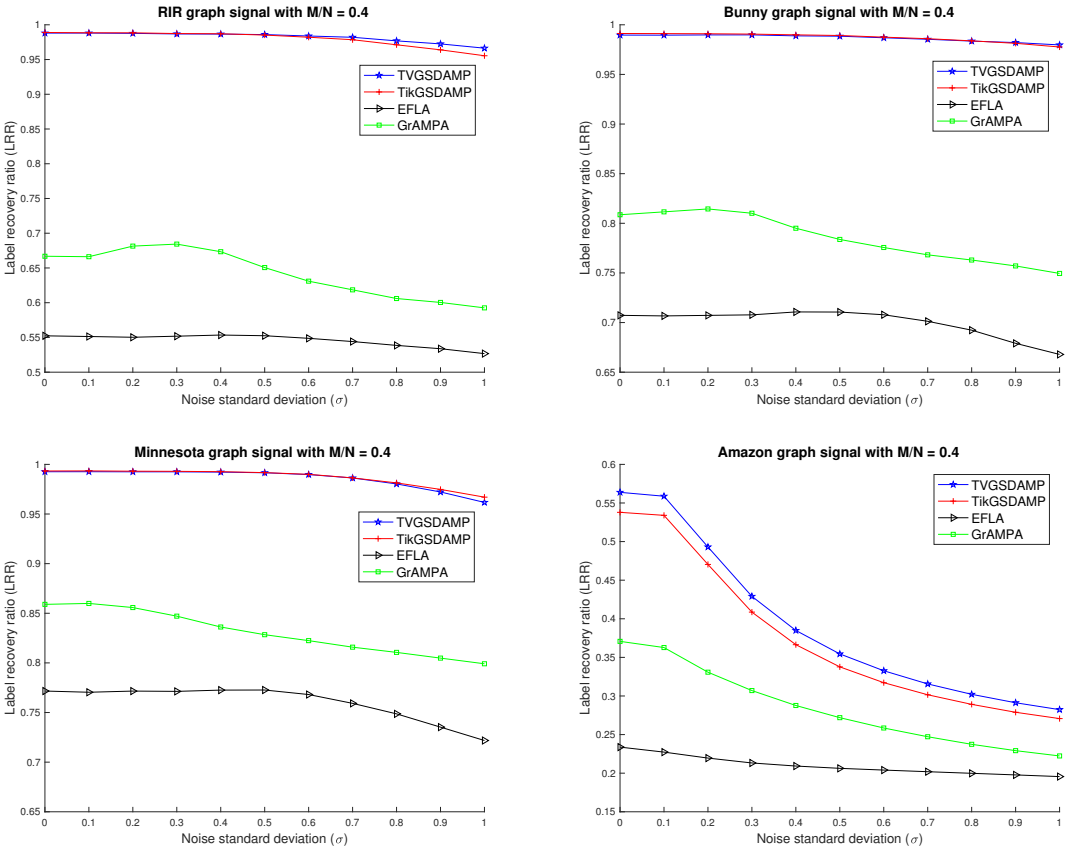


Figure B.2. LRR vs noise standard deviation σ, where the sampling rate M/N is set to 0.4.

decreases for all of the mentioned algorithms compared to the noiseless case, i.e., $\sigma = 0$. By increasing the noise standard deviation Figure B.2, the recovery slopes show that the TV-GSDAMP and Tik-GSDAMP algorithms are more robust to the noise compared to EFLA and GrAMPA algorithms.

Appendix C

C.1 List of Notations

Scalars, vectors and matrices

Symbol	Description
a	Random scalar
$\mathbf{a}$	Random vector
$\mathbf{A}$	Random matrix
a_n	n^{th} component of vector
$\mathbf{A} = (\mathbf{a}_1, \ldots, \mathbf{a}_N)$	columns of matrix
$(\mathbf{A})_{mn} = A_{mn}$	$(m, n)^{\text{th}}$ entry of a matrix
A_m	m^{th} row of a matrix
$\mathbf{a}^T, \mathbf{A}^T$	vector and matrix transpose
$\mathbf{A}^{-1}$	Inverse of matrix
$\lvert\mathbf{A}\rvert$	Determinant of a matrix
$\mathbf{I}_N$	$N \times N$ identity matrix
$\mathbf{0}_{M\times N}$	$M \times N$ all-zero matrix
$\mathbf{1}_{M\times N}$	$M \times N$ all-one matrix
$\mathbf{A} = diag(\mathbf{a})$	Diagonal matrix $\mathbf{A}$ with vector $\mathbf{a}$ entries
$trace(\mathbf{A}) = \sum_{n=1}^{n=N} A_{nn}$	Sum of diagonal entries of $N \times N$ matrix
$\lVert\mathbf{a}\rVert_p$	Vector p-norm ($p \geq 1$)
$\lVert\mathbf{A}\rVert_F$	Matrix Frobenius norm

Sets, probabilities and distributions

Variable	Description
$\mathcal{S}$	Set (in calligraphic font)
$\lvert\mathcal{S}\rvert$	Cardinality of a set (number of elements)
$\mathbf{a}_{\mathcal{S}}$	Vector with components indexed by set
$\mathbf{A}_{\mathcal{S}}$	Matrix with columns indexed by set
$(\cdot)^{(t)}$	Iteration index
$\mathbb{P}\{\cdot\}$	Probability of an event
$\mathbb{E}\{\cdot\}$	Expectation of a random quantity
$\mathrm{Var}\{\cdot\}$	Variance of a random quantity
$\mathrm{Var}(\cdot)$	Sample variance of a quantity
$\mathrm{Cov}\{\cdot\}$	Covariance of a random vector
$\mathrm{Cov}(\cdot)$	Sample covariance of a vector
$\mathcal{U}[\cdot]$	Uniform distribution
$\mathcal{N}(\mu, \Sigma)$	Multivariate Gaussian distribution with mean μ and (co-)variance Σ
$\mathbf{x} \sim \mathcal{N}(\mu, \sigma^2)$	$\mathbf{x}$ is Gaussian distributed with mean μ and variance σ^2
$\mathbf{x} \sim \mathcal{L}(\mu, k)$	$\mathbf{x}$ is Laplace distributed with mean μ and scale parameter k

C.2 List of Abbreviations

AGSS adaptive graph signal sampling
AMP approximate message passing
AWGN additive white Gaussian noise
BGS block Gauss-Seidel
BM3D block matching 3D
BP basis pursuit
BPDN basis pursuit denoising
CLT central limit theorem
CS compressed sensing
DAMP denoising-based approximate message passing
DCT discrete cosine transform
DFT discrete Fourier transform
DSPG discrete signal processing on graphs
DWT discrete wavelet transform
EFLA efficient fused lasso algorithm
GAMP generalized approximate message passing
GFT graph Fourier transform
GrAMPA GAMP for cosparse analysis
GS Gauss-Seidel
GSDAMP graph signal DAMP
GSP graph signal processing
IHT iterative hard thresholding
i.i.d. independent and identically distributed
IoT Internet of things
IST iterative soft thresholding
LASSO least absolute shrinkage and selection operator
LRR label recovery ratio
MAP maximum a-posteriori probability
MMSE minimum mean squared error
MRI magnetic resonance imaging
MSE mean squared error
MST maximum spanning tree
MWS maximum walking step
NLM non-local means
NMSE normalized mean squared error

RIR random irregular
SDD symmetric and diagonally dominant
SNIPE sparse non-informative parameter estimator
ST spanning tree
Tik-GSDAMP Tikhonov GSDAMP
TV-AMP total variation AMP
TV-GSDAMP total variation GSDAMP

C.3 List of Figures

C.4 List of Tables

Bibliography

[1] A. Sandryhaila and J. M. Moura, "Big data analysis with signal processing on graphs: Representation and processing of massive data sets with irregular structure," *IEEE Signal Processing Magazine*, vol. 31, pp. 80–90, sep 2014.

[2] S. Chen, R. Varma, A. Sandryhaila, and J. Kovačević, "Discrete signal processing on graphs: Sampling theory," *arXiv preprint arXiv:1503.05432*, 2015.

[3] A. J. Smola and R. Kondor, "Kernels and regularization on graphs," in *COLT*, vol. 2777, pp. 144–158, 2003.

[4] D. Zhou and B. Schölkopf, "A regularization framework for learning from graph data," in *ICML workshop on statistical relational learning and Its connections to other fields*, vol. 15, pp. 67–68, 2004.

[5] J. Kovacevic and M. Puschel, "Algebraic signal processing theory: Sampling for infinite and finite 1-D space," *IEEE Transactions on Signal Processing*, vol. 58, no. 1, pp. 242–257, 2010.

[6] S. J. Orfanidis, *Introduction to signal processing*. Prentice-Hall, Inc., 1995.

[7] S. Chen, R. Varma, A. Singh, and J. Kovacevic, "Signal recovery on graphs: Random versus experimentally designed sampling," in *International Conference on Sampling Theory and Applications (SampTA)*, pp. 337–341, 2015.

[8] X.-G. Xia, "On bandlimited signals with fractional fourier transform," *IEEE Signal Processing Letters*, vol. 3, no. 3, pp. 72–74, 1996.

[9] X. Zhu and M. Rabbat, "Graph spectral compressed sensing for sensor networks," *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 2865–2868, 2012.

[10] J. Kang, H. Jung, H.-N. Lee, and K. Kim, "Bernoulli-Gaussian approximate message-passing algorithm for compressed sensing with 1D-finite-difference sparsity," *arXiv preprint arXiv:1408.3930*, 2014.

[11] C. A. Metzler, A. Maleki, and R. G. Baraniuk, "From denoising to compressed sensing," *IEEE Transactions on Information Theory*, vol. 62, no. 9, pp. 5117–5144, 2016.

[12] G. B. Eslamlou, A. Jung, N. Goertz, and M. Fereydooni, "Graph signal recovery from incomplete and noisy information using approximate message passing," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6170–6174, 2016.

[13] G. B. Eslamlou, A. Jung, and N. Goertz, "Smooth graph signal recovery via efficient Laplacian solvers," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5915–5919, 2017.

[14] G. B. Eslamlou and N. Goertz, "Binary graph-signal recovery from noisy samples," in *The Signal Processing with Adaptive Sparse Structured Representations (SPARS)*, 2017.

[15] G. B. Eslamlou, M. Fereydooni, and N. Goertz, "Adaptive sampling of arbitrary graph signals," submitted to *IEEE Signal Processing Letters*, 2018.

[16] A. Sandryhaila and J. M. F. Moura, "Discrete signal processing on graphs," *IEEE Transactions on Signal Processing*, vol. 61, pp. 1644–1656, apr 2013.

[17] C. Zhang, D. Florêncio, and P. A. Chou, "Graph signal processing–a probabilistic framework," *Microsoft Res., Redmond, WA, USA, Tech. Rep. MSR-TR-2015-31*, 2015.

[18] X. Zhu and M. Rabbat, "Approximating signals supported on graphs," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3921–3924, 2012.

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

[20] D. I. Shuman, M. J. Faraji, and P. Vandergheynst, "A multiscale pyramid transform for graph signals," *IEEE Transactions on Signal Processing*, vol. 64, no. 8, pp. 2119–2134, 2016.

[21] D. I. Shuman, P. Vandergheynst, and P. Frossard, "Chebyshev polynomial approximation for distributed signal processing," in *International Conference on Distributed Computing in Sensor Systems and Workshops (DCOSS)*, pp. 1–8, 2011.

[22] A. Gadde, A. Anis, and A. Ortega, "Active semi-supervised learning using sampling theory for graph signals," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 492–501, ACM, 2014.

[23] A. Anis, A. Gadde, and A. Ortega, "Towards a sampling theorem for signals on arbitrary graphs," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3864–3868, May 2014.

[24] G. Camps-Valls, T. V. B. Marsheva, and D. Zhou, "Semi-supervised graph-based hyperspectral image classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 45, no. 10, pp. 3044–3054, 2007.

[25] S. Chen, A. Sandryhaila, and J. Kovačević, "Distributed algorithm for graph signal inpainting," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3731–3735, 2015.

[26] N. Tremblay and P. Borgnat, "Graph wavelets for multiscale community mining," *IEEE Transactions on Signal Processing*, vol. 62, no. 20, pp. 5227–5239, 2014.

[27] F. R. Chung, *Spectral graph theory*. No. 92, American Mathematical Soc., 1997.

[28] T. Bıyıkoglu, J. Leydold, and P. F. Stadler, "Laplacian eigenvectors of graphs," *Lecture notes in mathematics*, vol. 1915, 2007.

[29] R. K. Ando and T. Zhang, "Learning on graph with Laplacian regularization," *Advances in neural information processing systems*, vol. 19, p. 25, 2007.

[30] O. E. Livne and A. Brandt, "Lean algebraic multigrid (LAMG): Fast graph Laplacian linear solver," *SIAM Journal on Scientific Computing*, vol. 34, pp. B499–B522, jan 2012.

[31] F. Chung, "Laplacians and the cheeger inequality for directed graphs," *Annals of Combinatorics*.

[32] A. Sandryhaila and J. Moura, "Discrete signal processing on graphs: Frequency analysis," *Signal Processing, IEEE Transactions on*, vol. 62, pp. 3042–3054, June 2014.

[33] A. Sandryhaila and J. M. F. Moura, "Discrete signal processing on graphs: Graph filters," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6163–6166, 2013.

[34] D. I. Shuman, B. Ricaud, and P. Vandergheynst, "Vertex-frequency analysis on graphs," *Applied and Computational Harmonic Analysis*, vol. 40, no. 2, pp. 260–291, 2016.

[35] S. Chen, A. Sandryhaila, J. M. F. Moura, and J. Kovacevic, "Signal denoising on graphs via graph filtering," *IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, pp. 872–876, dec 2014.

[36] X. Zhu and M. Rabbat, "Graph spectral compressed sensing for sensor networks," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 2865–2868, March 2012.

[37] D. L. Donoho, "Compressed sensing," *IEEE Transactions on information theory*, vol. 52, no. 4, pp. 1289–1306, 2006.

[38] E. J. Candes, J. K. Romberg, and T. Tao, "Stable signal recovery from incomplete and inaccurate measurements," *Communications on pure and applied mathematics*, vol. 59, no. 8, pp. 1207–1223, 2006.

[39] E. J. Candès, J. Romberg, and T. Tao, "Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information," *IEEE Transactions on information theory*, vol. 52, no. 2, pp. 489–509, 2006.

[40] Q. Li, Y. Han, and J. Dang, "Image decomposing for inpainting using compressed sensing in DCT domain," *Frontiers of Computer Science*, vol. 8, no. 6, pp. 905–915, 2014.

[41] M. Lustig, D. Donoho, and J. M. Pauly, "Sparse MRI: The application of compressed sensing for rapid MR imaging," *Magnetic resonance in medicine*, vol. 58, no. 6, pp. 1182–1195, 2007.

[42] M. Lustig, D. L. Donoho, J. M. Santos, and J. M. Pauly, "Compressed sensing MRI," *IEEE signal processing magazine*, vol. 25, no. 2, pp. 72–82, 2008.

[43] Y. Zhang, B. S. Peterson, G. Ji, and Z. Dong, "Energy preserved sampling for compressed sensing MRI," *Computational and mathematical methods in medicine*, vol. 2014, 2014.

[44] R. Fergus, A. Torralba, and W. T. Freeman, "Random lens imaging," *Technical report, MIT*, 2006.

[45] M. F. Duarte, M. A. Davenport, D. Takbar, J. N. Laska, T. Sun, K. F. Kelly, and R. G. Baraniuk, "Single-pixel imaging via compressive sampling," *IEEE signal processing magazine*, vol. 25, no. 2, pp. 83–91, 2008.

[46] D. Panknin, *Compressed Sensing and Machine Learning*. PhD thesis, TU Berlin, 2012.

[47] C. M. Bishop, *Pattern recognition and machine learning*. springer, 2006.

[48] J. Bobin, J.-L. Starck, and R. Ottensamer, "Compressed sensing in astronomy," *IEEE Journal of Selected Topics in Signal Processing*, vol. 2, no. 5, pp. 718–726, 2008.

[49] F. Li, T. J. Cornwell, and F. de Hoog, "The application of compressive sampling to radio astronomy i: Deconvolution," *arXiv preprint arXiv:1106.1711*, 2011.

[50] A. Massa, P. Rocca, and G. Oliveri, "Compressive sensing in electromagnetics-a review," *IEEE Antennas and Propagation Magazine*, vol. 57, no. 1, pp. 224–238, 2015.

[51] W. Dai, M. A. Sheikh, O. Milenkovic, and R. G. Baraniuk, "Compressive sensing DNA microarrays," *EURASIP journal on bioinformatics and systems biology*, vol. 2009, no. 1, p. 162824, 2008.

[52] R. Dorfman, "The detection of defective members of large populations," *The Annals of Mathematical Statistics*, vol. 14, no. 4, pp. 436–440, 1943.

[53] H. F. Schepker and A. Dekorsy, "Sparse multi-user detection for CDMA transmission using greedy algorithms," in *8th International Symposium on Wireless Communication Systems (ISWCS)*, pp. 291–295, 2011.

[54] H. F. Schepker, C. Bockelmann, and A. Dekorsy, "Coping with CDMA asynchronicity in compressive sensing multi-user detection," in *IEEE 77th Vehicular Technology Conference (VTC Spring)*, pp. 1–5, 2013.

[55] B. Shim and B. Song, "Multiuser detection via compressive sensing," *IEEE Communications Letters*, vol. 16, no. 7, pp. 972–974, 2012.

[56] C. Bockelmann, H. F. Schepker, and A. Dekorsy, "Compressive sensing based multi-user detection for machine-to-machine communication," *Transactions on Emerging Telecommunications Technologies*, vol. 24, no. 4, pp. 389–400, 2013.

[57] L. Anitori, M. Otten, W. Van Rossum, A. Maleki, and R. Baraniuk, "Compressive CFAR radar detection," in *IEEE Radar Conference (RADAR)*, pp. 0320–0325, 2012.

[58] M. A. Herman and T. Strohmer, "High-resolution radar via compressed sensing," *IEEE transactions on signal processing*, vol. 57, no. 6, pp. 2275–2284, 2009.

[59] M. Elad and M. Aharon, "Image denoising via sparse and redundant representations over learned dictionaries," *IEEE Transactions on Image processing*, vol. 15, no. 12, pp. 3736–3745, 2006.

[60] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 267–288, 1996.

[61] T. Blumensath and M. E. Davies, "Iterative hard thresholding for compressed sensing," *Applied and computational harmonic analysis*, vol. 27, no. 3, pp. 265–274, 2009.

[62] K. Bredies and D. A. Lorenz, "Linear convergence of iterative soft-thresholding," *Journal of Fourier Analysis and Applications*, vol. 14, no. 5, pp. 813–837, 2008.

[63] I. Daubechies, M. Defrise, and C. De Mol, "An iterative thresholding algorithm for linear inverse problems with a sparsity constraint," *Communications on pure and applied mathematics*, vol. 57, no. 11, pp. 1413–1457, 2004.

[64] T. Blumensath and M. E. Davies, "Normalized iterative hard thresholding: Guaranteed stability and performance," *IEEE Journal of selected topics in signal processing*, vol. 4, no. 2, pp. 298–309, 2010.

[65] T. Blumensath, "Accelerated iterative hard thresholding," *Signal Processing*, vol. 92, no. 3, pp. 752–756, 2012.

[66] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM journal on imaging sciences*, vol. 2, no. 1, pp. 183–202, 2009.

[67] D. L. Donoho, "De-noising by soft-thresholding," *IEEE transactions on information theory*, vol. 41, no. 3, pp. 613–627, 1995.

[68] R. Tibshirani, M. Saunders, S. Rosset, J. Zhu, and K. Knight, "Sparsity and smoothness via the fused lasso," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 67, no. 1, pp. 91–108, 2005.

[69] D. L. Donoho, A. Maleki, and A. Montanari, "Message-passing algorithms for compressed sensing," *Proceedings of the National Academy of Sciences*, vol. 106, no. 45, pp. 18914–18919, 2009.

[70] D. L. Donoho, A. Maleki, and A. Montanari, "Message passing algorithms for compressed sensing: I. motivation and construction," in *IEEE Information Theory Workshop (ITW)*, pp. 1–5, 2010.

[71] M. A. Maleki, *Approximate message passing algorithms for compressed sensing*. PhD thesis, Stanford University, 2011.

[72] M. R. Andersen, *Sparse inference using approximate message passing*. PhD thesis, Copenhagen: Technical University of Denmark, 2014.

[73] S. Rangan, "Generalized approximate message passing for estimation with random linear mixing," in *IEEE International Symposium on Information Theory Proceedings (ISIT)*, pp. 2168–2172, 2011.

[74] S. Rangan, A. K. Fletcher, V. K. Goyal, and P. Schniter, "Hybrid generalized approximate message passing with applications to structured sparsity," in *IEEE International Symposium on Information Theory Proceedings (ISIT)*, pp. 1236–1240, 2012.

[75] M. Borgerding, P. Schniter, J. Vila, and S. Rangan, "Generalized approximate message passing for cosparse analysis compressive sensing," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3756–3760, apr 2015.

[76] J. Liu, L. Yuan, and J. Ye, "An efficient algorithm for a class of fused lasso problems," in *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 323–332, ACM, 2010.

[77] M. Fornasier and H. Rauhut, "Iterative thresholding algorithms," *Applied and Computational Harmonic Analysis*, vol. 25, no. 2, pp. 187–208, 2008.

[78] K. K. Herrity, A. C. Gilbert, and J. A. Tropp, "Sparse approximation via iterative thresholding," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 624–627, 2006.

[79] J. S. Yedidia, "Message-passing algorithms for inference and optimization," *Journal of Statistical Physics*, vol. 145, no. 4, pp. 860–890, 2011.

[80] E. Van Den Berg and M. P. Friedlander, "Probing the pareto frontier for basis pursuit solutions," *SIAM Journal on Scientific Computing*, vol. 31, no. 2, pp. 890–912, 2008.

[81] A. Maleki and A. Montanari, "Analysis of approximate message passing algorithm," in *44th Annual Conference on Information Sciences and Systems (CISS)*, pp. 1–7, 2010.

[82] D. L. Donoho, A. Maleki, and A. Montanari, "How to design message passing algorithms for compressed sensing," *preprint*, 2011.

[83] H.-A. Loeliger, "An introduction to factor graphs," *IEEE Signal Processing Magazine*, vol. 21, no. 1, pp. 28–41, 2004.

[84] H.-A. Loeliger, J. Dauwels, J. Hu, S. Korl, L. Ping, and F. R. Kschischang, "The factor graph approach to model-based signal processing," *Proceedings of the IEEE*, vol. 95, no. 6, pp. 1295–1322, 2007.

[85] A. Buades, B. Coll, and J.-M. Morel, "A non-local algorithm for image denoising," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 2, pp. 60–65 vol. 2, June 2005.

[86] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "Image denoising by sparse 3-D transform-domain collaborative filtering," *IEEE Transactions on Image Processing*, vol. 16, pp. 2080–2095, Aug 2007.

[87] M. Bayati and A. Montanari, "The dynamics of message passing on dense graphs, with applications to compressed sensing," *IEEE Transactions on Information Theory*, vol. 57, no. 2, pp. 764–785, 2011.

[88] D. Guo and C.-C. Wang, "Random sparse linear systems observed via arbitrary channels: A decoupling principle," in *IEEE International Symposium on Information Theory (ISIT)*, pp. 946–950, 2007.

[89] D. Guo and C.-C. Wang, "Asymptotic mean-square optimality of belief propagation for sparse linear systems," in *IEEE Information Theory Workshop (ITW)*, pp. 194–198, 2006.

[90] H. Ishwaran and J. S. Rao, "Spike and slab variable selection: Frequentist and Bayesian strategies," *Annals of statistics*, pp. 730–773, 2005.

[91] J. Ziniel, P. Schniter, and P. Sederberg, "Binary linear classification and feature selection via generalized approximate message passing," in *48th Annual Conference on Information Sciences and Systems (CISS)*, pp. 1–6, 2014.

[92] Y. Nesterov, *Introductory lectures on convex optimization: A basic course*, vol. 87. Springer Science & Business Media, 2013.

[93] S. Osher, M. Burger, D. Goldfarb, J. Xu, and W. Yin, "An iterative regularization method for total variation-based image restoration," *Multiscale Modeling & Simulation*, vol. 4, no. 2, pp. 460–489, 2005.

[94] D. P. Bertsekas, *Nonlinear programming*. Athena Scientific, 1999.

[95] P. Getreuer, "Rudin-Osher-Fatemi total variation denoising using split Bregman," *Image Processing On Line*, vol. 2, pp. 74–95, 2012.

[96] N. Perraudin, J. Paratte, D. Shuman, V. Kalofolias, P. Vandergheynst, and D. K. Hammond, "GSPBOX: A toolbox for signal processing on graphs," *arXiv preprint arXiv:1408.5781*, 2014.

[97] S. Ramani, T. Blu, and M. Unser, "Monte-Carlo Sure: A black-box optimization of regularization parameters for general denoising algorithms," *IEEE Transactions on Image Processing*, vol. 17, pp. 1540–1554, sep 2008.

[98] D. Gleich, "MatlabBGL. A Matlab graph library," *Institute for Computational and Mathematical Engineering, Stanford University*, 2008.

[99] J. Leskovec, L. A. Adamic, and B. A. Huberman, "The dynamics of viral marketing," *ACM Transactions on the Web*, vol. 1, p. 5, may 2007.

[100] G. H. Golub, P. C. Hansen, and D. P. O'Leary, "Tikhonov regularization and total least squares," *SIAM Journal on Matrix Analysis and Applications*, vol. 21, pp. 185–194, jan 1999.

[101] J. Liu, S. Ji, J. Ye, *et al.*, "SLEP: Sparse learning with efficient projections," *Arizona State University*, vol. 6, no. 491, p. 7, 2009.

[102] M. Borgerding and P. Schniter, *Generalized approximate message passing for analysis compressive sensing*. http://www2.ece.ohio-state.edu/ schniter/GrAMPA/download.html.

[103] D. L. Donoho, I. Johnstone, and A. Montanari, "Accurate prediction of phase transitions in compressed sensing via a connection to minimax denoising," *IEEE transactions on information theory*, vol. 59, no. 6, pp. 3396–3433, 2013.

[104] P. L. Combettes and J.-C. Pesquet, "A douglas–rachford splitting approach to nonsmooth convex variational signal recovery," *IEEE Journal of Selected Topics in Signal Processing*, vol. 1, no. 4, pp. 564–574, 2007.

[105] S. Becker, J. Bobin, and E. J. Candès, "Nesta: A fast and accurate first-order method for sparse recovery," *SIAM Journal on Imaging Sciences*, vol. 4, no. 1, pp. 1–39, 2011.

[106] J. Wright, A. Ganesh, S. Rao, Y. Peng, and Y. Ma, "Robust principal component analysis: Exact recovery of corrupted low-rank matrices via convex optimization," in *Advances in neural information processing systems*, pp. 2080–2088, 2009.

[107] Y. Saad, *Iterative methods for sparse linear systems*. Society for Industrial and Applied Mathematics, 2 ed., 2003.

[108] W. J. Stewart, *Probability, Markov chains, queues, and simulation: The mathematical basis of performance modeling*. Princeton University Press, 2009.

[109] R. Varga, *Matrix iterative analysis*. Englewood Cliffs, New Jersey: Prentice Hall, 1 ed., 1962.

[110] N. K. Vishnoi, "$Lx = b$ Laplacian solvers and their algorithmic applications," *Foundations and Trends in Theoretical Computer Science*, vol. 8, no. 1–2, pp. 1–141, 2012.

[111] K. Gremban, *Combinatorial preconditioners for sparse, symmetric, diagonally dominant linear systems*. PhD thesis, Carnegie Mellon University, Pittsburgh, Oct. 1996.

[112] K. D. Gremban, G. L. Miller, and M. Zagha, "Performance evaluation of a new parallel preconditioner," in *Proceedings of 9th International Parallel Processing Symposium*, pp. 65–69, Apr. 1995.

[113] G. T. Gilbert, "Positive definite matrices and Sylvester's criterion," *The American Mathematical Monthly*, vol. 98, no. 1, pp. 44–46, 1991.

[114] G. H. Golub and C. F. Van Loan, *Matrix computations*. Johns Hopkins University Press, 3rd edition ed., 1996.

[115] H. Q. Nguyen and M. N. Do, "Downsampling of signals on graphs via maximum spanning trees.," *IEEE Trans. Signal Processing*, vol. 63, no. 1, pp. 182–191, 2015.

[116] D. B. West *et al.*, *Introduction to graph theory*, vol. 2. Prentice hall Upper Saddle River, 2001.

[117] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms, Third Edition*. The MIT Press, 3rd ed., 2009.

[118] L. A. Adamic and N. Glance, "The political blogosphere and the 2004 US election: divided they blog," in *Proceedings of the 3rd international workshop on Link discovery*, pp. 36–43, ACM, 2005.